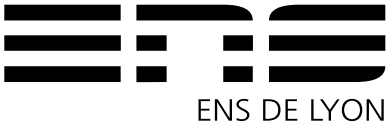
Numerical simulation and rare events algorithms for the study of extreme fluctuations of the drag force acting on an obstacle immersed in a turbulent flow

Thibault Lestang

Numéro National de Thèse : 2018LYSEN049

THÈSE de DOCTORAT DE L'UNIVERSITÉ DE LYON

opérée par

l'École Normale Supérieure de Lyon

École Doctorale N°52

École Doctorale de Physique et Astrophysique de Lyon (PHAST)

Spécialité de doctorat : Physique

Soutenue publiquement le 25/09/2018, par :

Thibault LESTANG

Numerical simulation and rare events algorithms for the study of extreme fluctuations of the drag force acting on an object immersed in a turbulent flow

Simulation numérique et algorithmes d'échantillonnage d'événements rares pour l'étude des fluctuations extrêmes de la force de traînée sur un obstacle immergé dans un écoulement turbulent

---

Devant le jury composé de :

DUBRULLE, Bérengère	Directrice de recherche	SPEC, CNRS	Rapportrice
NOULLEZ, Alain	Directeur de recherche	OCA, CNRS	Rapporteur
SIMONNET, Eric	Chargé de recherche	INPHYNI, CNRS	Examinateur
SCHNEIDER, Kai	Professeur	I2M, Aix-Marseille Université	Examinateur
PUMIR, Alain	Directeur de recherche	ENS de Lyon, CNRS	Examinateur
LANOTTE, Alessandra S.	Chargée de recherche	ISAC, CNR	Examinatrice
BOUCHET, Freddy	Directeur de recherche	ENS de Lyon, CNRS	Directeur
LÉVÊQUE, Emmanuel	Directeur de recherche	Ecole Centrale de Lyon, CNRS	Co-encadrant

NUMERICAL SIMULATION AND RARE EVENTS ALGORITHMS FOR THE STUDY OF EXTREME EVENTS IN TURBULENT FLOWS

THIBAULT LESTANG

Freddy Bouchet & Emmanuel Lévêque

Application to extreme fluctuations of the drag force on an obstacle

Laboratoire de Physique de l'ENS de Lyon & LMFA, Ecole Centrale Lyon

Lyon, September 2018

à Marité

ABSTRACT

This thesis discusses the numerical simulation of extreme fluctuations of the drag force acting on an object immersed in a turbulent medium. Because such fluctuations are rare events, they are particularly difficult to investigate by means of direct sampling. Indeed, such approach requires to simulate the dynamics over extremely long durations. In this work an alternative route is introduced, based on rare events algorithms. The underlying idea of such algorithms is to modify the sampling statistics so as to favour rare trajectories of the dynamical system of interest. These techniques recently led to impressive results for relatively simple dynamics. However, it is not clear yet if such algorithms are useful for complex deterministic dynamics, such as turbulent flows. This thesis focuses on the study of both the dynamics and statistics of extreme fluctuations of the drag experienced by a square cylinder mounted in a two-dimensional channel flow. This simple framework allows for very long simulations of the dynamics, thus leading to the sampling of a large number of events with an amplitude large enough so as they can be considered extreme. Subsequently, the application of two different rare events algorithms is presented and discussed. In the first case, a drastic reduction of the computational cost required to sample configurations resulting in extreme fluctuations is achieved. Furthermore, several difficulties related to the flow dynamics are highlighted, paving the way to novel approaches specifically designed to turbulent flows.

RÉSUMÉ

Cette thèse porte sur l'étude numérique des fluctuations extrêmes de la force de traînée exercée par un écoulement turbulent sur un corps immergé. Ce type d'événement, très rare, est difficile à caractériser par le biais d'un échantillonage direct, puisqu'il est alors nécessaire de simuler l'écoulement sur des durées extrêmement longues.

Cette thèse propose une approche différente, basée sur l'application d'algorithmes d'échantillonnage d'événements rares. L'objectif de ces algorithmes, issus de la physique statistique, est de modifier la statistique d'échantillonnage des trajectoires d'un système dynamique, de manière à favoriser l'occurrence d'événements rares. Si ces techniques ont été appliquées avec succès dans le cas de dynamiques relativement simples, l'intérêt de ces algorithmes n'est à ce jour pas clair pour des dynamiques déterministes extrêmement complexes, comme c'est le cas pour les écoulement turbulents.

Cette thèse présente tout d'abord une étude de la dynamique et de la statistique associée aux fluctuations extrêmes de la force de traînée sur un obstacle carré fixe immergé dans un écoulement turbulent à deux dimensions. Ce cadre simplifié permet de simuler la dynamique sur des durées très longues, permettant d'échantillonner un grand nombre de fluctuations dont l'amplitude est assez élevée pour être qualifiée d'extrême. Dans un second temps, l'application de deux algorithmes d'échantillonage est présentée et discutée. Dans un premier cas, il est illustré qu'une réduction significative du temps de calcul d'extrêmes peut être obtenue. En outre, des difficultés liées à la dynamique de l'écoulement sont mises en lumière, ouvrant la voie au développement de nouveaux algorithmes spécifiques aux écoulements turbulents.

Feelings don't dictate outcomes.

— Gwen Jorgensen [72]

REMERCIEMENTS

La rédaction d'un manuscrit de thèse est un travail de longue haleine. Arrivé en bout de course, les yeux rivés sur la ligne d'arrivée, plus grand chose ne compte que d'en terminer. En en oublierait presque d'avoir une pensée pour ses parents. N'exagérons rien. Des parents qui m'ont encouragé, assisté, écouté, accompagné. Souvent loin de leurs mondes, mais toujours là où je voulais aller. Je pense ensuite aux camarades "Lyonnais". Ils étaient en première ligne, que l'on célèbre ou que l'on se plaigne. Je pense regretter toujours ce SMS libérateur, arrivant au terme d'une longue journée à n'y rien comprendre, annonciateur d'une soirée en bonne compagnie: «Pinte Douce ?» —De laquelle on ressort en ayant tout compris, ou pas, mais certainement plus léger!.

Une thèse représente beaucoup d'énergie injectée dans une seule chose. Il y a alors de très bon moments, et de très mauvais. Au Laboratoire de Physique ou au LMFA, j'ai bénéficié d'un environnement d'une qualité que j'estime exceptionnelle. Pendant presque 4 années, j'y ai côtoyé, partagé, fraternisé avec des gens, certes brillants, mais avant tout passionnés. Toujours prêts à partager leur énergie, donner un coup de main, une suggestion. Mais aussi en recevoir, l'esprit grand ouvert. Je remercie particulièrement Corentin Herbert, qui ne mesure peut-être pas totalement l'impact qu'ont eu sa disponibilité, son calme et sa gentillesse sur mon travail et mon moral. Merci aussi à Francesco Ragone pour sa simplicité, son courage et, bien sûr, son amitié. Sans oublier Takahiro Nemoto et Charles-Edouard Bréhier, avec qui j'ai toujours un grand plaisir à discuter. Remerciements spéciaux à Alessandro De Rosis, prophète du Lattice Boltzmann, qui m'a bien formé.

Il fallait bien un paragraphe à part pour remercier Freddy et Emmanuel. J'ai reçu de leur part une bienveillance manifeste, tantôt sous forme de conseils, de critiques ou tout simplement d'une présence attentive. Il me semble avoir mûri au cours de cette thèse. Sur le plan scientifique bien sûr, mais pas que. Si le temps y est sans doute pour quelque chose, Emmanuel et Freddy y sont aussi.

Je ne peux pas conclure ces remerciements sans rédiger quelques mots pour Les Sacrées Tignasses. Quand on se lance dans une entreprise de longue haleine, ardue, le soutien indéfectible de bons copains-copines est un atout précieux. Alors que dire de celui d'une telle bande de macaques ! Ces amis de toujours qui, même sans comprendre grand chose au projet lui même, ont bien compris l'importance

qu'il avait pour moi. Merci aussi aux Orcéens restés se battre contre le RER B ou exilés au pieds des Alpes, j'ai toujours aimé aller vous voir ou vous recevoir. Mention spéciale à Mathilde, qui a relu courageusement les chapitres 1 et 2 de cette thèse.

CONTENTS

ACRONYMS

PDF	Probability Density Function
i.i.d.	Independent Identically Distributed
OU	Ornstein–Ulhenbeck
AMS	Adaptive Multilevel Splitting
TAMS	Trajectory Adaptive Multilevel Splitting
SCGF	Scaled Cumulant Generating Function
LBM	Lattice Boltzmann Method
LBE	Lattice Boltzmann Equation
LGCA	Lattice Gas Cellular Automata
LBGK	Lattice Bhatnagar–Gross–Krook
GKTL	Giardina–Kurchan–Tailleur–Lecomte
DNS	Direct Numerical Simulation
MD	Molecular Dynamics
CFD	Computational Fluid Dynamics

INTRODUCTION

A striking property of turbulent flows is the random occurrence of strong coherent structures, emerging from the apparent disorder of the flow. Such structures, for instance vortices, are linked to intense fluctuations of the velocity gradients and pressure fields. Considering extreme fluctuations, the typical return period of these events is very long compared with the typical timescale of the turbulent fluctuations. In this sense, such extreme fluctuations are *rare events*.

In spite of their scarcity, extreme fluctuations in turbulent flows can have dramatic consequences on the structure of the flow itself, as well as on immersed objects. Examples include the fluctuations of turbulence intensity in wall-bounded flows near the transition to turbulence [63, 119, 138] or rare transitions between attractors in turbulent flows, for instance in zonal jet dynamics [12].

In addition, the formation of strong dynamical structures in the vicinity of an immersed object can have an important mechanical impact, such as extreme fluctuations of the drag and lift forces. An example is the impact of extreme wind loads on tall structures, such as buildings [91] or wind turbines [74, 87, 88]. In addition to the potential wreckage of the system—as illustrated in figure 0.1—extreme wind gusts can be held responsible for premature fatigue of the structure [87]. Other examples include vehicle aerodynamics [23, 67] or marine biology [36].

Since the drawings of Da Vinci in the early 1500s, the physics of turbulent flows remains poorly understood. Although the governing equation, the Navier-Stokes equations, can be derived from elementary conservation laws, they encode complex dynamics characterised by

Figure 0.1: Collapsed wind turbine in Bouin, France. It is attributed to extreme winds that occurred during the Carmen storm that hit western France in early January 2018. The turbine had been in place for 15 years. It is worth noting that it resisted several similar and even stronger storms, such as Xynthia in February 2010.

interactions across all scales. To this date, there exist no definitive universal theory capable of predicting the statistics of fluctuations in turbulent flows. The characterisation of the dynamics related to rare events is therefore a step towards a more complete understanding of the physics of turbulent flows.

NUMERICAL SIMULATIONS OF TURBULENT FLOWS

Numerical experiments

Because of the extreme mathematical complexity of the equations describing fluid flows, analytical investigation of turbulence with pencil and paper is out of the question. On the basis of a suitable mathematical model, an alternative approach to experiments is numerical simulation. With the advent of modern computers and High Performance Computing, Computational Fluid Dynamics (CFD) has become a major tool in the study of turbulent flows, for both fundamental research and engineering problems, in both academic and industrial contexts [46].

The large computational cost of simulating turbulence

However, the computational approach in hindered by the *tremendous* computational cost associated with the numerical integration of the Navier-Stokes equations. As a matter of fact, a dimensional analysis shows that the amount of grid points N^3 required to resolve the small scales in three dimensional turbulent flows grows as a power aw of the Reynolds number: $N^3 \sim Re^{9/4}$. Numerical integration of the Navier-Stokes equations, without any additional approximations, is referred to as Direct Numerical Simulation (DNS). To fix ideas, the largest DNS to date has been published in 2015 [178]. It solves the three dimensional, incompressible Navier-Stokes equations in a periodic box with a resolution of 8.192^3 spatial mesh points.

When it comes to industrial or geological flows, DNS are out of the question. Indeed, for the Reynolds numbers encountered in such applications, the computing resources required by a DNS exceed the capacity of the most powerful computers available. As a consequence, practical applications must rely on computational approaches based on approximations, such as Large Eddy Simulation (LES) [146] or Reynolds Averaged Navier-Stokes (RANS) methods.

The need for rare event methods

In this context, the numerical simulation of rare events in turbulent flows appears like a daunting task. Indeed, the common approach relies on the simulation of the flow over very long durations, in

order to obtain a representative sample of extreme events [104, 147, 150]. In view of the previous discussion, this direct method involves tremendous computational burden. Moreover, it is limited to simple flows which numerical simulation require low computational effort. In most cases however, rare events simply cannot be accessed.

There is thus a need for computational techniques decicated to the simulation of extreme events in turbulence, in order to bypass the requirement for long integration times. In this thesis we address the numerical sampling of extremes in turbulent flows, based on computational methods originating from statistical physics. Such methods are applied *on top* of the numerical simulation of the flow and alter the statistics of extremes, so that they are sampled with a higher probability;*as opposed* to a direct simulation of the flow over long durations.

RARE EVENT SAMPLING

Systems at equilibrium

In computational statistical mechanics, equilibrium systems are commonly simulated by means of Monte Carlo methods [10]. For systems in static equilibrium, the fluctuating behaviour is simulated by sampling states according to the stationary distribution. This distribution has a Boltzmann-like form and can be sampled using classical Markov Chain Monte Carlo strategies such as the *Metropolis-Hastings* algorithm [34]. For systems in dynamic equilibrium, rare events sampling is commonly achieved by means of *Path Sampling*. In this approach, rare transitions between different regions of phase space are sampled using equilibrium Monte Carlo methods in trajectory space. Path Sampling techniques emerged with the development of Transition Path Sampling in 2001 [11]. Subsequently, a large body of methods has been developed, such as Transition Interface Sampling [44] or Milestoning[77]. Path Sampling methods are primarily applied to the simulation of rare events in biomolecular systems [45]. Importantly, this approach requires that the dynamics are reversible in time, with a Boltzmann phase space stationary distribution [2, 45].

Rare event sampling in turbulent flows

By contrast, the dynamics of non-equilibrium systems is *not* reversible, and the stationary distribution is not known. It is for instance the case for turbulent flows. Consequently, the sampling must be performed at the level of the trajectories. To this date, the computational investigation of rare events in turbulence have been primarily based on simplified hydrodynamical models with stochastic forcing [64, 65,

69, 71]. An example is the random-forced Burgers equation [8]. In this framework, it is possible to derive an *action* for trajectories, see chapter 1, section 1.2.1. It plays a role analogous to the energy for trajectories in phase space. As a result, it allows for a Monte Carlo sampling of the path measure [111, 112]. A second approach is based on the instanton method [65]. It also relies on the knowledge of an action describing the landscape of the path measure. In the limit of weak stochastic forcing, trajectories connecting two states in phase space are expected to concentrate on the most probable path, referred to as the *minimum action path*. This path can be computed by numerically minimising the action [13, 32, 69, 101].

Methods based on the knowledge of the action, whether they perform Monte Carlo sampling or minimisation of the action, are limited to stochastic dynamics. However, most turbulence problems are fully deterministic. In addition, even in situations where a small noise is present, it is expected to play little role in the physical mechanisms responsible for extreme events.

As a consequence, more general methods are required in order perform computational studies of extremes in turbulent flows, and especially flows that are of interest for the engineer.

Rare event algorithms that work for non-equilibrium dynamics

In this thesis we address a different route to rare event sampling in turbulent flows. We consider the application of rare event algorithms that are not limited to stochastic dynamics and that have been proven relevant in chaotic deterministic systems [177]. More importantly, there are applicable to non-equilibrium, irreversible dynamics. Most of these algorithms rely on the simulation of a population of copies of the system. Along the simulation, copies are replicated according to how well they perform with respect to the realisation of a rare event. In this way, the ensemble of trajectories is enriched in trajectories corresponding to extreme events. In this thesis we focus on two specific algorithms: the Adaptive Multilevel Splitting [27, 141] and the Giardina–Kurchan–Tailleur–Lecomte algorithm [57, 120]. Although both methods are relatively recent, the corresponding algorithms are rooted in older ideas such as Diffusion Monte Carlo methods [89], go-with-the-winners algorithms [68, 167] or splitting algorithms [61]. As an illustration, figure 0.2 depicts the procedure corresponding to one iteration of the Adaptive Multilevel Splitting algorithm.

In this work we address the applicability of both the GKTL and AMS algorithms to the numerical simulation of rare events in turbulent flows. Although both algorithms are very general, to this date they have never been applied to the numerical simulation of the Navier-

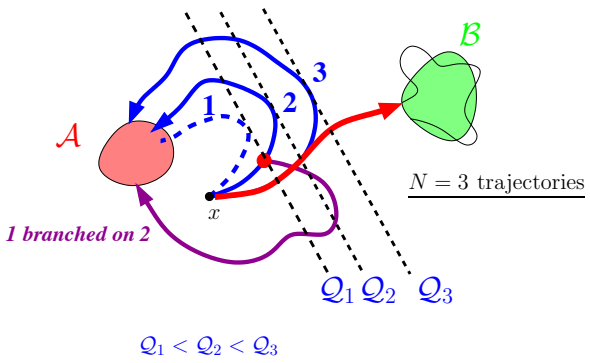


Figure 0.2: Illustration of an iteration of the AMS algorithm. In this example we consider two regions of phase space $\mathcal{A}$ and $\mathcal{B}$ for which transitions to one another are very rare. In this sketch, three trajectories are computed from the initial condition x located within the basin of attraction of $\mathcal{A}$. They are computed until they either reach $\mathcal{B}$ or fall to $\mathcal{A}$. The levels $\{\mathcal{Q}\}_{1\leq i\leq 3}$ correspond to the maximum value of a *score function* over the corresponding trajectory, that quantifies how close each trajectory got from $\mathcal{B}$. Having the lowest level among the ensemble, trajectory 1 (dashed) is discarded from the ensemble of trajectories. Trajectory 2 is then randomly selected and copied until it reaches the maximum of 1, identified by a red dot. From there, dynamics are integrated until it either reach $\mathcal{A}$ or $\mathcal{B}$. In typical applications this procedure is iterated many times until all trajectories reach $\mathcal{B}$.

Stokes equations. Their relevance and practicality for very complex dynamics such as turbulent flows is an open question.

A SIMPLE FRAMEWORK FOR A FIRST STUDY

Together with [12], the work described in this thesis is the first attempt at sampling rare events in turbulent flows without relying on the definition and minimisation of an action in trajectory space. In addition, the flows considered in this work are fully deterministic. The objective of the present work is to assess the relevance of rare event algorithms to mitigate the computational cost associated with the numerical simulation of extreme events in turbulence.

Two-dimensional turbulent flow past a square cylinder

As a first application of the AMS and GKTL algorithms, we consider a simple geometry in which a square obstacle is embedded into a two-dimensional channel flow. An example is given in figure 0.3. While the relevance of two-dimensional turbulence is questionable for practical applications, it provides a convenient framework for the study of extremes, as well as first tests with rare events algorithms. Indeed, the simulation of two-dimensional flows is much less demanding than

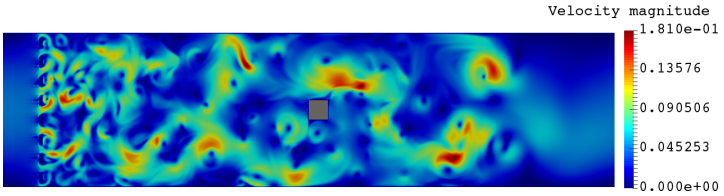


Figure 0.3: Snapshot of the velocity field surrounding a square obstacle. The flow goes from left to right and upstream turbulence is generated by means of a grid.

their three-dimensional counterparts. In this simplified framework, extremes *can* be investigated on the basis of a very long simulation of the flow.

Extreme fluctuations of the drag force

More precisely, we focus on extreme fluctuations of the longitudinal force acting on the obstacle, referred to as the *drag* force. The study of drag fluctuations on immersed obstacles is relevant to numerous contexts, such as ground vehicle aerodynamics [23, 67], astrophysics [82] or civil engineering [91]. While the characterisation of the statistics and dynamics of fluctuating drags have been addressed in previous studies [7, 82, 162], the study of *extreme* events have been left aside. Consequently, in the first part of this thesis we present a study of the extreme drag fluctuations acting on a square cylinder, based on a very long simulation of the flow. Extreme drag fluctuations are identified and locally re-simulated in order to compute the physical quantities, as well as visualise the corresponding flow fields. From the sampling of roughly one hundred events, we are able to distinguish the common properties of rare events of similar extreme amplitudes. Furthermore, we propose a classification of extremes according to two dynamical scenarios.

In a second part of the thesis the GKTL and AMS are applied to the sampling of extreme drag fluctuations on a square cylinder. Thanks to the relative simplicity of the flow, the results obtained from the algorithms can be compared to the direct sampling based on a very long simulation. Using the GKTL algorithm we compute the *large deviation rate function* describing the tail statistics of the time-averaged drag. We show that the GKTL algorithm yields an accurate estimate for a much lower computational cost than direct sampling.

By contrast, we find that the AMS algorithm does not lead to a significant computational gain. This result highlights the fact that the application of the AMS to chaotic deterministic dynamics, such as turbulent flows, is not straightforward. The reasons for this difficulties are illustrated and perspectives of adaptation of the algorithm are discussed.

The Lattice Boltzmann Method

Throughout this work, numerical simulations of turbulent flows are based on the Lattice Boltzmann Method (LBM) [92, 155]. Since its first appearance in the late 1990s, the LBM is attracting ever-growing interest in numerous communities, as it offers a very simple algorithmic procedure applicable to a wide variety of problems in fluid mechanics, including turbulence. In this thesis we make use of a novel formulation of the LBM, called the *central-moments based* lattice Boltzmann scheme [144]. It is very well suited for the simulation of turbulent flows.

Plan of the thesis

In chapter 1, general aspects of the statistics of rare events are presented. Basic results from large deviation theory are also introduced.

In chapter 2 we discuss two two-dimensional geometries in which a square obstacle is immersed in a turbulent flow.

On the basis of these flows, extreme drag fluctuations are sampled from the integration of the dynamics over very long durations. Chapter 3 presents the characterisation of both the dynamics and statistics corresponding to these extreme events.

In chapter 4 the GKTL algorithm is introduced. Its connection with *large deviation theory* is highlighted. Moreover, its practical implementation for complex, chaotic deterministic dynamics is discussed.

The application of the GKTL algorithm to the sampling of extreme drag fluctuations is presented in chapter 5

Finally, in chapter 7 we address the numerical computation of *return times* for rare events, *i.e* the typical timescale of occurence of the fluctuations. The estimation of return times—or return periods—is discussed based on either long timeseries of the observable of interest, or rare events algorithms such as the AMS or GKTL algorithms.

1

RARE EVENTS, THE POISSON PROCESS AND THEORETICAL APPROACHES

In many physical systems, the mean state and the typical fluctuations about this state, usually studied in statistical physics, are not the only quantities of interest. Indeed, fluctuations far away from the mean state, although they are usually very rare, can play a crucial part in the macroscopic behaviour of the system. For instance, they can drive the system to a new metastable state, possibly with radically different properties [90]. Such transitions arise in a wide variety of situations, such as Josephson junctions [93], quantum oscillators [38], turbulent flows [15], magneto-hydrodynamics dynamos [9], diffusion-controlled chemical reactions [25], protein folding [124], climate dynamics [126]. Even if the system returns to its original state after undergoing the large fluctuation, the impact of this event may be so large that it is worth being studied on its own. One may think for instance about heat waves [136] and tropical cyclones, rogue waves in the ocean [39], strong dissipative events in turbulent flows [179], shocks in financial markets [43]. Here, we are concerned with the study of such atypical fluctuations starting from the equations (deterministic or stochastic) that govern the system's dynamics. This approach is different from and complementary to the purely statistical methods which try to extract the best possible information about the distribution of rare events from an existing timeseries, such as, for instance, extreme value statistics [48, 55, 107].

1.1 THE PHENOMENOLOGY OF RARE EVENTS

In this thesis we discuss rare events in dynamical systems. Typically, we consider extreme fluctuations of a given observable A of the dynamics beyond a threshold a. This situation is depicted in figure 1.1 for the example of an Ornstein–Ulhenbeck (OU) process. In this particular example, the dynamics is stochastic and one-dimensional. The observable A is chosen as the state variable itself $A(x) = x$. The values of A appear to randomly fluctuate around an average value, $\langle A \rangle = 0$. The typical timescale of these fluctuations is the *correlation time* of the process, wich is denoted by τ_c in the following. It can be thought of as the time it takes for the evolution of the process to become statistically independent of its previous values at earlier times. Furthermore, the evolution of A displays typical fluctuations around the average, having a typical timescale of occurrence close to the correlation time. By contrast, the process also displays fluctuations to values far away

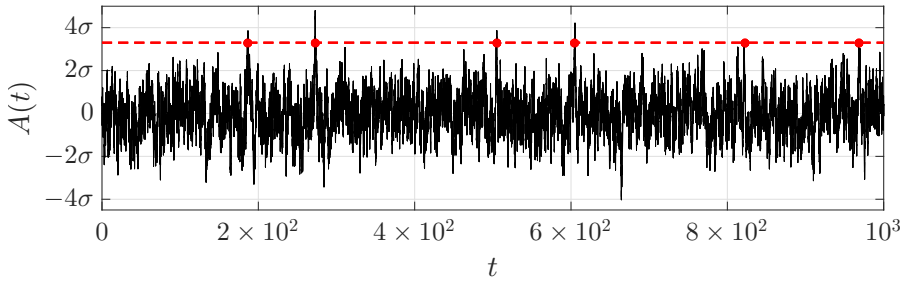


Figure 1.1: An example of a random process associated to events reaching a given threshold. **(a)**: Sample timeseries (black line), generated from an Ornstein–Uhlenbeck process (1.4); $\sigma = 1/\sqrt{2}$ is the standard deviation. We are interested in fluctuations which reach a prescribed threshold $a = 3.6\sigma$ (dashed red line). These events are identified by the red dots.

from the average, identified by the red dots in figure 1.1. The typical timescale of occurrence of these fluctuations is much larger than the correlation time. In the example of figure 1.1 only 6 events above the level denoted by the dashed red line are sampled in a timeseries spanning 10^3 correlation times. These rare fluctuations are located in the tails of the Probability Density Function (PDF) describing the statistics of the values of A.

In the following we refer to the typical timescale of occurrence of a fluctuation $A \geq a$ as its *return time*, denoted by $r(a)$. More precisely, we define the waiting time

$$\tau(a,t) = \min\{\tau \geq t \,|A(\tau) > a\} - t. \tag{1.1}$$

Then the return time $r(a)$ for the threshold a is defined as

$$r(a) = \mathbb{E}[\tau(a,t)]. \tag{1.2}$$

The return time therefore corresponds to the average duration one has to wait in order to observe a given fluctuation $A \geq a$.

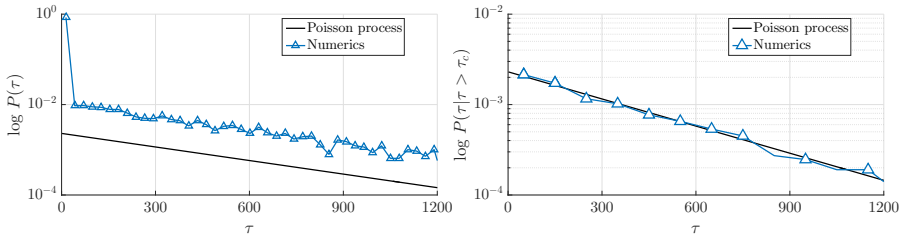
1.1.1 *The Poisson approximation for rare events*

In practice, we must distinguish two kinds of events $A(t) \geq a$. On the one hand, there are correlated events corresponding to fluctuations around the threshold a, on a timescale of the order of the correlation time. On the other hand, there are successive events such as those depicted in figure 1.1, which can be considered as statistically independent events. Because the return time of rare events is much larger than the correlation time of the process, we therefore consider the fluctuations around the threshold a as the same event. In other words,

we neglect the temporal strucutre of rare fluctuations $A(t) \geq a$ and consider them as one-time events.

The random occurrence of statistically independent one-time rare events can be modelled by a Poisson process [37, 43, 102]. A Poisson process is described by a single parameter $\lambda(a)$ which quantifies the typical rate at which events $A(t) \geq a$ occur. Let τ be the intermediate time interval between two consecutive events. For a Poisson process, the distribution of intermediate times is:

$$P(\tau) = \lambda(a) \exp(-\lambda(a)\tau). \tag{1.3}$$



(a) (b)

Figure 1.2: PDF of waiting times between two consecutive fluctuations of amplitude $a = 2.5$, estimated from a timeseries of length $T_d = 10^6$ of the Ornstein–Uhlenbeck process (1.4) (blue triangles), and assuming that the events follow a Poisson process with rate $\lambda(a) = 1/r(a)$, $P(\tau) = \lambda(a)e^{-\lambda(a)\tau}$ (black solid line), where $r(a)$ is computed from the timeseries. The correlation time of the Ornstein–Uhlenbeck process is $\tau_c = 1$. **(a)** Taking all intervals into account, including those corresponding to oscillations around the threshold. **(b)** Discarding small intervals ($\tau < \tau_c$) linked to oscillations around the threshold.

Figure 1.2a shows the Probability Density Function (PDF) of the time interval between two occurrences of an event $x(t) \geq a$, drawn from a sample timeseries $\{x(t)\}_{0 \leq t \leq T}$ generated with an Ornstein–Uhlenbeck process defined as

$$\dot{x}(t) = -x(t) + \sqrt{2}\eta(t), \tag{1.4}$$

where η is a Gaussian white noise. One can see that most of the contributions are indeed small intervals of the order of the correlation time. Discarding all the time intervals below the correlation time, one obtains the PDF displayed in Fig. 1.2b, which coincides with the exponential distribution corresponding to a Poisson point process. The rate of the process corresponds to the return time of the fluctuation: $r(a) = 1/\lambda(a)$.

1.1.2 *How to access the return time numerically ?*

Although the statistics of rare events can be described by a simple Poisson process with a simgle parameter $\lambda(a)$, the numerical estimation of this parameter is a difficult task. Because rare events occur over times much larger than the correlation time, the dynamical equations describing the evolution of the system must be integrated over a very large number of correlation times in order to sample a statistically significant number of events. In many applications, the numerical simulation of a few correlation times is a scientific and technological challenge in itself. An example is the simulation of turbulent flows. Therefore, there is a need for computational methods improving the sampling of rare events. More precisely, statistical algorithms which, coupled with the simulation of the dynamics, sample extreme fluctuations with integration times accessible to state-of-the-art numerical simulations.

1.2 LARGE DEVIATION THEORY

The theoretical framework which has been developed over the last decades in statistical physics to tackle the investigation of rare events is that of *large deviation theory* [35, 42, 49, 164, 174]. Large deviation theory is concerned with the asymptotic decay of the probability of rare events as a function of a small parameter ϵ. Let us consider a PDF $\mathcal{P}_\epsilon$. For instance it could be the PDF of the sample mean S_N over N Independent Identically Distributed (i.i.d.) variables $\{X_n\}_{1\leq n\leq N}$: $S_N = \frac{1}{N}\sum_{n=1}^{N} X_n$. In this case $\epsilon = 1/N$. A PDF $\mathcal{P}_\epsilon$ is said to verify a *large deviation principle* if there exist a so-called *rate function* I so that

$$\lim_{\epsilon\to 0} \epsilon \ln \mathcal{P}_\epsilon(a) = I(a). \tag{1.5}$$

The large deviation principle is commonly written as

$$\mathcal{P}_\epsilon(a) \underset{\epsilon\to 0}{\asymp} e^{-\frac{I(a)}{\epsilon}}, \tag{1.6}$$

where the symbol $f_\epsilon \underset{\epsilon\to 0}{\asymp} g_\epsilon$ denotes logarithmic equivalence as ϵ goes to 0: $\ln(f_\epsilon) \underset{\epsilon\to 0}{\sim} \ln(g_\epsilon)$. As $I(a)$ can be shown to be convex, the large deviation principle states that the probability of observing a fluctuation a away from the typical value $a^\star$, for which $I(a^\star) = \min_a I(a)$, decays exponentially with ϵ. The rate of the decay is $I(a)$.

The term *large deviation theory* can actually refer to different theories, depending on the nature of the parameter ϵ. For instance, the Donsker-Varadhan theory of large deviations describes the decay of the probabilities of the fluctuation of the integral of a process $x(t)$, in

the limit of large integration times. The corresponding large-deviation principle reads:

$$P\left(\frac{1}{T}\int_0^T f(t)\mathrm{d}t = a\right) \underset{T\to\infty}{\asymp} e^{-TI(a)}. \tag{1.7}$$

Another example is Freidlin-Wentzell theory of large deviations. It deals with the probabilities of trajectories in stochastic dynamical systems in the limit of weak noise. It leads to large deviation results that describe the exponential decay of the probability of trajectories away from the most probable one.

In the following we briefly discuss Freidlin-Wentzell theory. We illustrate that in stochastic systems with weak noise, the dynamics of rare events can often be predicted. Then, we present the main ideas of the Donsker-Varadhan theory of large deviations. We explain that it can be understood as a generalisation of the central limit theorem that goes beyond typical fluctuations.

1.2.1 *Large deviations for trajectories in stochastic dynamics*

Freidlin–Wentzell theory deals with dynamical systems perturbed by a stochastic term of weak amplitude. The stochastic differential equation that describes such a system writes

$$\dot{x} = b(x) + \sqrt{2\epsilon}\sigma(x)\eta(t), \tag{1.8}$$

where the system $x(t)$ has dimension N, $b(x)$ is a M-dimensional vector field, $\sigma(x)$ is a $N \times M$ matrix and $\eta(t) = (\eta^m(t))_{1\leq m\leq M}$ is a M dimensional white noise term

$$\eta^m(t)\eta^{m'}(t') = \delta_{mm'}\delta(t-t').$$

We denote by $\mathcal{P}_\epsilon(x, T|x_0, 0)$ the probability to find the system in state x at time $t = T$ given the initial condition x_0 at time $t = 0$. At the core of Freidlin–Wentzell theory is the derivation of a large deviation principle, not only for the transition probability $\mathcal{P}_\epsilon(x, T|x_0, 0)$ but also for the probability $\mathcal{P}_\epsilon[x(t)]$ to observe any trajectory $x(t)$ for the system in the interval $[0, T]$. The probability density $\mathcal{P}_\epsilon[x(t)]$ is thus a *functional*: it depends on the whole trajectory $x(t)$ which is itself a function of time.

For the sake of simplicity, we consider the case where x is a scalar and $\sigma(x) = 1$. In this case the probability $\mathcal{P}_\epsilon[x(t)]$ verifies a large deviation principle as follows:

$$\mathcal{P}_\epsilon[x] \asymp e^{-\frac{1}{\epsilon}\mathcal{A}[x]}. \tag{1.9}$$

The functional $\mathcal{A}[x]$ is called the path *action* and writes

$$\mathcal{A}[x] = \frac{1}{4}\int_0^T (\dot{x} - b(x))^2 \mathrm{d}t \tag{1.10}$$

One can see that the action is always positive. It is consistent with the fact that it is a large deviation functional, analogous to the large deviation rate function $I(a)$ in (1.5). More importantly, the minimum of the action is zero and is always reached for the deterministic path of equation $\dot{x} = b(x)$.

So far, we did not specify the initial conditions for the path $x(t)$. The large deviation result (1.9) is valid for *any* path in the interval $[0, T]$. In this section, we are concerned with the transition probability $\mathcal{P}_\epsilon(x_T, T|x_0, 0)$ to reach a final state x_T starting from x_0. We thus have to restrict ourselves to the paths that satisfy the two constraints $\{x(T) = x_T, x(0) = x_0\}$. The large deviation result (1.9) is still valid for this particular class of paths.

The probability $\mathcal{P}_\epsilon(x_T, T|x_0, 0)$ can actually be written as a path integral

$$\mathcal{P}(x_T, T|x_0, 0) = \int_{\{x(T)=x_T, x(0)=x_0\}} \mathcal{D}[x]\, e^{-\frac{1}{4\epsilon}\int_0^T \mathrm{d}t(\dot{x}(t)-b(x(t)))^2}, \tag{1.11}$$

where the integral runs over all trajectories starting from the initial condition x_0 at $t = 0$ and reaching x_T at $t = T$. Equation (1.11) indicates that the most probable trajectories with prescribed initial and final states are minimizers of the action with prescribed initial and final points. The optimal action is denoted

$$A(x_0, x_T, T) = \min\{\mathcal{A}[x]|x(0) = x_0, x(T) = x_T\}. \tag{1.12}$$

Following the large deviation principle (1.9), the probability of transition paths from x_0 to x_T concentrates around the action minimizer for $\epsilon \to 0$. Such a path is called an *instanton*.

1.2.1.1 *Fluctuation paths*

When the initial point x_0 belongs to an attractor of the deterministic dynamics, it is expected that the action $A(x_0, x_T, T)$ decreases with time. The action minima starting from an attractor and having an infinite duration will thus play an important role. Those action minimizers starting from an attractor and with an infinite duration are called *fluctuation paths*. They are defined as

$$A(x_0, X) = \min\left\{\mathcal{A}[x] \lim_{T\to\infty} x(-T) = x_0, x(0) = X\right\} \tag{1.13}$$

Freidlin-Wentzell theory therefore suggests that, in the limit of low noise, the probability of trajectories of such fluctuations concentrate on the fluctuation paths. As a consequence, in the limit of low noise, the dynamics leading to extreme fluctuations is predictable, as it corresponds to the minimisation of (1.13). One of the original motivations of this work is to assess the validity of this result result for complex chaotic dynamics such as turbulent flows. Indeed, from a modelling perspective, turbulent flows can be viewed as a stochastic dynamics with a weak noise, *i.e.* small scale turbulent fluctuations. Indeed, it is not clear if the behaviour of the large scale motions can be well described by a determinisic dynamics perturbed by a stochastic noise, resulting from small scale turbulent fluctuations.

1.2.2 *Independent Identically Distributed variables and the Donsker-Varadhan theory of large deviations*

We now turn to a different kind of large deviation theory which plays a central role in this thesis. It describes the exponential decay of the probability of fluctuations away from the average in the limit where the average is computed over a large number of realisations for random variables, or over long durations for random processes. In particular we will illustrate that it generalises the central limit theorem.

1.2.2.1 *Large deviation principle for a sum of i.i.d. variables*

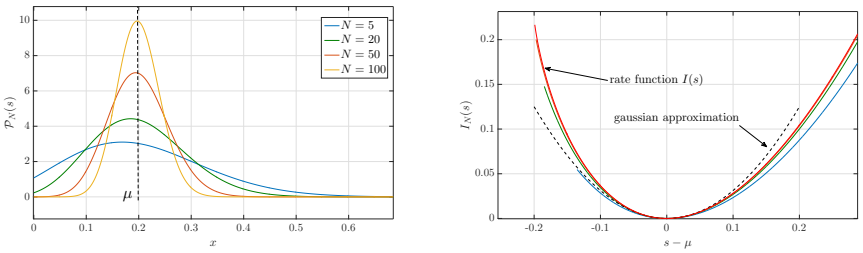
Let X be a random variable described by a PDF $\mathcal{P}$. Furthermore we note $\mu = \mathbb{E}[X]$ and $\sigma^2 = \mathbb{E}[X^2] - \mu^2$. In the following we consider the sample mean over N realisations of X, defined as

$$S_N = \frac{1}{N} \sum_{n=1}^{N} x_n. \tag{1.14}$$

We know from the law of large numbers that $S_N \underset{N\to\infty}{\to} \mu$. Furthermore, the central limit theorem states that the PDF of S_N converges in law towards a Gaussian PDF with mean μ and standard deviation $\sigma/\sqrt{N}$. Actually, it can be proved that the sample mean S_N verifies a large deviation principle[1]:

$$\mathcal{P}(S_N = s) \underset{N\to\infty}{\asymp} e^{-NI(s)} \tag{1.15}$$

1 This result is actually the first result of large deviation theory. It dates back to the 1930s and is due to the Swedish mathematician Harald Cramèr who first applied this result in actuarial science and insurance mathematics. Accordingly, it is known as Cramèr's theorem.



(a) Probability density function (b) Rate function

Figure 1.3: Illustration of the large deviation principle for the sample mean over N Bernoulli variables with parameter μ. **(a)** Probability Density Function $\mathcal{P}(S_N = s)$. One can see that the distributions are more and more concentrated around the mean value μ of the Bernoulli process as N is increased. In addition, the typical width of the PDF decreases like $1/\sqrt{N}$. **(b)** Estimates of the rate function for finite N corresponding to the PDFs displayed in figure 1.3a. As N is increased, the estimates converge towards the rate function $I(s)$. In addition, the Gaussian approximation (1.20) is only accurate over a $1/\sqrt{N}$ range around the mean value μ, *i.e.* the minimum of the rate function. I thank Eric Woillez for these figures.

As a matter of fact, this result contains both the law of large numbers and the central limit theorem. Following (1.15), the expectation value μ reads:

$$\mu = \int \mathrm{d}s \, s\mathcal{P}(s) \approx \int \mathrm{d}s \, s e^{-NI(s)}. \tag{1.16}$$

As N is increased, a saddle point approximation therefore yields $\mu = I(s^\star)$ with $s^\star$ the minimizer of $I(s)$. This is illustrated in figure 1.3b. As N grows, the distribution of S_N therefore concentrates around the average value of μ. This corresponds to the law of large numbers and is illustrated in figure 1.3b. Let us now consider the reduced random variable $Z_N = (S_N - \mu)/\sigma\sqrt{N}$. A linear change of variable in the large deviation principle (1.15) leads to

$$\mathcal{P}(Z_N = z) = \frac{\sigma}{\sqrt{N}} C\left(N, \mu + \frac{\sigma}{\sqrt{N}} z\right) e^{-NI(\mu + \frac{z\sigma}{\sqrt{N}})}. \tag{1.17}$$

where C is a prefactor. Considering small deviations from μ, a series expansion at second order yields:

$$I(a) \approx I(\mu) + I'(\mu) + \frac{I''(\mu)}{2}(a - \mu)^2 \tag{1.18}$$

Recalling that $I(\mu) = I'(\mu) = 0$, one gets

$$\mathcal{P}(Z_N = z) \approx \frac{\sigma}{\sqrt{N}} C\left(N, \mu\right) e^{-\frac{1}{2} I''(\mu) z^2 \sigma^2}. \tag{1.19}$$

Because of the normalisation constraint $\int \mathcal{P}(z)\mathrm{d}z = 1$, the prefactor $\frac{1}{\sqrt{N}}C\left(N,\mu\right)$ must converge to the normalisation constant, with $I''(\mu) = 1/\sigma^2$. As a result,

$$\lim_{N\to\infty} \mathcal{P}(Z_N = z) = \sqrt{\frac{1}{2\pi}} e^{-\frac{1}{2}z^2} \tag{1.20}$$

which corresponds to the central limit theorem. We stress here that the large deviation principle (1.15) goes beyond the central limit theorem. Indeed, it states that, for a fixed s, the probability decays exponentially as N is increased. In addition, the rate of decay is $I(s)$. Figure 1.3b displays the quantity $I_N = -\frac{1}{N}\ln(\mathcal{P}(s)/\mathcal{P}(\mu))$ for the sum of N Bernoulli variables, for different N. It illustrates that the different curves converge towards the rate function I as N is increased. In addition, the Gaussian approximation is also displayed, computed using (1.20). It illustrates that $\mathcal{P}(s)$ is not Gaussian. Indeed, the central limit theorem is only valid for fluctuations amplitude up to $1/\sqrt{N}$. A fluctuation further away from μ is always possible, but with exponentially small probability. Such atypical fluctuations are called *rare events*.

1.2.2.2 *Large-time large deviations (Donsker-Varadhan)*

In the following we give a heuristic justification that the existence of a large deviation principle for the sum of i.i.d. random variables can be extended to the time integral over continuous-time processes. Consider a random process $f(t)$ with a correlation time τ_c. In the following we consider time-integrals such as

$$\int_0^T f(t)\mathrm{d}t. \tag{1.21}$$

In addition, we assume that T is large enough so that the integral can be split in a sequence of N integrals over a duration $\Delta T \gg \tau_c$:

$$\int_0^T f(t)\mathrm{d}t = \sum_{i=1}^{T/\Delta T} \int_{(i-1)\Delta T}^{i\Delta T} f(t)\mathrm{d}t. \tag{1.22}$$

Under the assumption $\Delta T \gg \tau_c$, equation (1.22) can be considered as a sum over i.i.d. variables. As a result, we write the following large deviation principle, based on the large deviation principle for i.i.d. variables discussed in the previous section:

$$P\left(\frac{1}{T}\int_0^T f(t)\mathrm{d}t = a\right) \underset{T\to\infty}{\asymp} e^{-TI(a)}. \tag{1.23}$$

This result is often referred to as *large-time large deviations*, or Donsker-Varadhan large deviations. It plays in important role in this manuscript as it is the cornerstone of the GKTL algorithm, that allows to numerically sample extreme fluctuations of large-time averaged observables.

Part I

THE PHENOMENOLOGY OF EXTREME DRAG FLUCTUATIONS

Even though they are very rare, extreme events are worth being studied on their own. First of all, because they often have a big impact on the dynamics. Furthermore, extreme events often feature interesting physical mechanisms, which understanding can provide new insights on the general behaviour of the system under study. In the following two chapters, we describe both the statistics and the dynamics of extreme fluctuations of the drag force acting on a square cylinder mounted in a two-dimensional channel. In chapter 2 we describe several simple flow configurations for which the numerical investigation of extreme events can be achieved by means of very long simulations. In chapter 3, we investigate the physics of extreme fluctuations of the drag acting on the obstacle, as well as the time-averaged drag.

2

TEST FLOWS

In this chapter, we describe the choice and design of flow geometries that will serve as test cases throughout this thesis. First of all, the corresponding dynamics will be simulated over very long durations in order to investigate the statistics and dynamics of extreme fluctuations of the drag force acting on an obstacle immersed in the flow. Second, we will apply rare events algorithms on the basis of these test flows, in order to assess the applicability of such approaches to rare event sampling in turbulent flows. Rare event algorithms are usually not very restrictive about the properties of the dynamics on which they are applied. In addition, as far as most algorithms are concerned, dynamics is often seen as a black box. As a consequence there are, *a priori*, no restrictions over the choice of the flow dynamics.

In practice however, we would like to keep it simple. Clearly, testing rare event algorithms on complex, three-dimensional turbulent flows such as the flow past a turbine or a truck appears like an unreasonable first step. Indeed, the numerical simulation of such flows involves tremendous computational resources. This is especially true when considering the simulation of rare events, as we aim at computing the flow dynamics over long times, in order to sample extreme fluctuations directly. In addition to the study of extreme events, the integration of the dynamics over long durations will provide a reference for the results of the algorithms to compare to. Rare events algorithms involved in this work rely on the evolution and replication of a population of clones of the flow. In practice, it requires the storage and manipulation of information about the state of the clones in memory, for instance the velocity and pressure fields. Therefore, we would like to work on a test flow that can be simulated using a rather coarse grid, in order to limit the volume of data and to be able to use a large number of copies of the flow.

In summary, a suitable test flow must verify the following requirements :

- It must feature fully developed turbulence
- Simulation of long time series, say 1 million of characteristic times, must be performed in a wallclock time of the order of the week.
- Required spatial resolution must be low enough to mitigate memory storage requirement, as well as computing time.

Natural candidates are two-dimensional flows. Nowadays, most simulations of two-dimensional turbulence can be performed on a laptop, as they involve much less computations and data storage than their 3D counterparts. In addition, numerical simulations of two-dimensional flows are easier to implement, analyse and visualise. However, the physics of 2D turbulence differs from 3D turbulence. As a result, two-dimensional flows are often regarded as unrealistic and irrelevant to industrial purposes. Nonetheless, 2D turbulence embed the key features that we believe make the application of rare event algorithms to turbulence a challenging and worthy problem: long-range spatio-temporal correlations that result in the chaotic occurrence of flow structures, accountable for intense fluctuations of the macroscopic fields such as the pressure or the velocity. This work is motivated by the argument that, if it succeeds in establishing a proof-of-principle for rare event algorithms on the basis of 2D flows, extension to 3D turbulence should be straightforward, and only hindered by difficulties linked to the numerical simulation of complex three-dimensional turbulent flows, and not by the rare event algorithms themselves.

If the restriction to two-dimensional flows is a already a big step, there still remain a great number of possible choices for the test flow itself. In CFD, two iconic flows are the Lid-Driven Cavity Flow and the flow in a channel, often referred to as the Hagen-Poiseuille flow in the laminar regime. These two different geometries are classical benchmarks on which numerical methods for flow simulation are commonly tested. As a consequence, extensive literature and corresponding experimental and numerical results are available. The Lid-Driven Cavity flow has the advantage of being ruled by the most simple possible boundary conditions, namely no-slip boundaries on walls aligned with the grid, except for a constant imposed velocity for the lid. However, such flow has been found to be exceptionally stable, meaning that if a turbulent regime is ever to be reached, it is only for very high Reynolds numbers. By contrast, the flow in a channel, *in presence of an obstacle at its centre*, is known to reach turbulence for moderate Reynolds number, thus involving moderate computational resources for its numerical simulation.

In the following we discuss two geometries. Both are two-dimensional channel flows with square cylinders at a fixed position along the axis of the channel. The first test flow has periodic boundaries at the inlet and outlet, and is forced through a force density following the channel axis. The second test flow is imposed a parabolic velocity profile at the inlet and an open boundary at the outlet. Turbulence is generated by a grid, that is an array of short walls and holes positioned next to the inlet. These two flows verify the criteria listed above.

In this chapter we describe the numerical simulation of these two test cases. The simulation are based on the Lattice Boltzmann Method [92, 155], a numerical method that solves the Navier-Stokes Equations. Note that this method somewhat differs from conventional Computational Fluid Dynamics approaches. Indeed, it does not rely on the discretisation of the Navier-Stokes Equations, but instead on the simulation of the dynamics of an ensemble of fictitious particles evolving along a discrete lattice according to the *Lattice Boltzmann Equation*. This equation originates from the discretisation of the Boltzmann Equation, that provides a statistical description of the physics of dilute gases. The Lattice Boltzmann Method is discussed in section 2.1, which focuses on the practical aspects of the simulation of fluid flows using the LBM. The discussion of the theoretical foundations and justification of the method is left aside, and key references are given for interested readers. Sections 2.2 and 2.3 discuss the numerical simulation as well as the properties of both test flows, respectively. In each case, the statistics of the drag acting on the obstacles are described on the basis of a very long timeseries, resulting from a simulation of the flow over a long duration.

2.1 THE LATTICE BOLTZMANN METHOD

In this section we give a very brief presentation of the Lattice Boltzmann Method. The method is further discussed in appendix B. The Lattice Boltzmann Method (LBM) offers a computationally efficient particle-based alternative to conventional *continuum*-based approaches to simulate fluid dynamics [155]. The fluid is considered at a kinetic level intermediate between the microscopic and the macroscopic. More precisely, the fluid is viewed as populations of particles that collide, redistribute and propagate along the different links of a discrete lattice. The complexity of the flow emerges from the repeated application of simple rules of collision and streaming of these populations at each lattice node. The flow variables such as velocity and pressure are obtained by averaging locally over the populations of particles moving in the different directions (Fig. 2.1). This obviously refers to a kinetic description of fluid dynamics and rigorous connections can be established with the Boltzmann equation [31], under the so-called LBGK approximation [129].

Formally, the LBM scheme reads as

$$f_i(\mathbf{x} + \mathbf{c}_i \Delta t, t + \Delta t) = f_i(\mathbf{x}, t) + \Omega_i(\mathbf{x}, t) \qquad \text{for } i = 0, \ ...8 \qquad (2.1)$$

where $f_i(\mathbf{x}, t)$ represent the amount of mass (per unit volume) carried by the particles moving (with speed $\mathbf{c}_i$) in the i^{th}-direction at position $\mathbf{x}$ and time t. Additionally, $\Omega_i(\mathbf{x}, t)$ is the *collision operator*, that can encore many properties of the fluid, such as the viscosity.

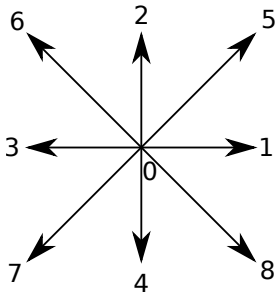


Figure 2.1: Illustration of a particular node of the lattice. The velocity space describing the motion of particles is descretised according to 9 velocities $\{\mathbf{c}_i\}_{0\leq i\leq 8}$. The velocity $\mathbf{c}_0$ correspond to particles at rest. This particular discretisqtion is referred to as D2Q9: two spatial dimensions and 9 velocities.

In its simplest form, the collision operator acts like a relaxation towards equilibrium with a single timescale τ, referred to as the Lattice Bhatnagar–Gross–Krook (LBGK) operator [129]:

$$\Omega_i(\mathbf{x},t) = -\frac{f_i(\mathbf{x},t) - f_i^{eq}(\mathbf{x},t)}{\tau}\Delta t \qquad \text{for } i = 0,\ ...8, \tag{2.2}$$

where the $\{f_i^{eq}(\mathbf{x},t)\}_{0\leq i\leq 8}$ denote the values of the populations corresponding at thermodynamic equilibrium.

The macroscopic quantities are recovered locally by summing the contributions

$$\rho = \sum_i f_i \tag{2.3}$$

$$\rho\mathbf{u} = \sum_i f_i\mathbf{c}_i \tag{2.4}$$

$$\mathbf{S} = -\frac{1}{2\tau c_s^2\rho}\sum_i \mathbf{c}_i\mathbf{c}_i(f_i - f_i^{eq}) \tag{2.5}$$

$$\tag{2.6}$$

Where ρ, $\mathbf{u}$ and $\mathbf{S}$ are the density, velocity and strain-rate on the lattice node, respectively. The pressure is intrinsically given by $p = c_s^2\rho$ in a weak-compressibility approximation, where c_s may be interpreted as the speed of sound. The relaxation parameter τ is related to the kinematic viscosity of the fluid: $\tau = \Delta t/2 + \nu/c_s^2$.

Since its emergence in the early 90s [154, 155], the LBM evolved into many variants [92], depending on the choice of the collision operator. In this thesis, we make use of a specific LBM scheme referred to as the central-moments based LBM [143, 144]. This particular LBM formulation shows exceptional stability for highly turbulent flows.

2.2 TEST FLOW (1): CHANNEL FLOW WITH PERIODIC BOUNDARY CONDITIONS

In this section we describe the first of the two test flows on which this project is based, referred to as test flow (1). Its numerical simulation is carried out with the Lattice Boltzmann Method, introduced in the previous section. Test flow (1) consists in the flow in a two-dimensional channel with periodic boundary conditions along the channel axis. The flow is forced through a constant, homogeneous force density acting on the whole domain.

In a first version, a single square cylinder is introduced at the centre of the channel. This first version is referred to as test flow (0). Due to the periodicity along the flow direction, turbulence is triggered by the interaction of the obstacle with its own wake. In section 2.2.1 we describe the flow in more detail and present results for the computation of the drag acting on the obstacle. On the basis of a long simulation of the flow, we observe that the autocorrelation function of the drag has a zero valued integral, indicating that the corresponding PDF does not verify a large-deviation principle. We then explain this pathological behaviour through the conservation of momentum across the computational domain.

In section 2.2.2 we present a modified version of test flow (0), referred to as test flow (1). In this version, two identical square cylinders are introduced in the flow, along the channel axis line. In this setup, the PDF describing the statistics of the drag on a *single* square does verify a large-deviation principle. We first characterise the flow by discussing numerical parameters, as well as the corresponding Reynolds numbers and average velocity profile along the channel axis. Then, we illustrate the statistics of the drag acting on the obstacles, on the basis of a very long simulation of the flow. We show that, for both obstacles, the autocorrelation of the drag displays an exponential decay. Furthermore, both drag timeseries display similar PDF, for which both negative an positive tails are well described by an exponential PDF.

2.2.1 *Test flow (0): a single obstacle, pathological large deviation behaviour*

In this section we discuss a first version of test flow (1), in which a single obstacle is introduced in the channel. In the following we refer to this flow as test flow (0). The corresponding geometry is sketched in figure 2.2. It consists in a two-dimensional channel with periodic boundaries at the inlet and the outlet. In concrete terms, fluid particles flowing through the outlet reappear at the inlet, such as the flow in a torus. The fluid is set in motion by means of a constant and homogeneous force density acting over the whole domain. The walls of the channel correspond to no-slip boundaries, at which both components of the velocity field vanish. Last but not least, a square

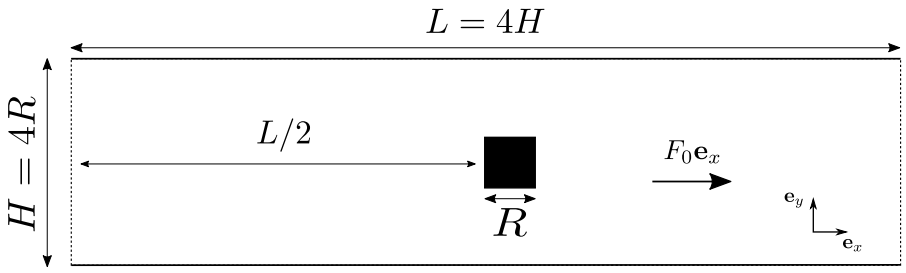


Figure 2.2: Sketch of the computational domain for test flow (0) with a single obstacle. The computational domain is periodic along the channel axis and the flow is forced through a constant, homogeneous force density of amplitude F_0. No-slip boundary conditions are imposed on both the surface of the obstacle and the top and bottom walls of the channel.

cylinder with no-splip boundaries is introduced at the centre of the computational domain.

In the following we discuss the choice of the numerical parameters for the numerical simulation of test flow (0) using the Lattice Boltzmann Method. Namely, the lattice spacing δx, the force density amplitude F_0 and the relaxation parameter τ.

2.2.1.1 *Choice of numerical parameters*

The choice of numerical parameters follows three guidelines:

1. The resulting flow must have a high enough Reynolds number so that the flow exhibits developed turbulence.
2. Strong stability of the simulation, as it will further be used to sample extreme events.
3. Small computational burden for the simulation of the dynamics. Typically, we expect the wallclock time for the simulation of one characteristic time to be of the order of the second.

To begin with, space is discretised into a regular, isotropic lattice. Denoting by R the diameter of the square cylinder, the lattice spacing is set to $\delta x = R/16$. Following dimensions indicated in figure 2.2, the computational domain therefore consists in a grid of 65×193 lattice nodes.

As mentioned in section 2.1, the LBM solves the incompressible Navier-Stokes Equations in the weakly compressible limit. As a result, we choose the intensity of the force density F_0 so that the typical velocity remains small with respect to the speed of sound in the fluid. More precisely, we denote by $Ma(\mathbf{x}, t)$ the local, instantaneous Mach number Ma, denoting the ratio between the fluid velocity and the speed of sound. Therefore, we choose F_0 so that $\overline{Ma(\mathbf{x})} \ll 1$ everywhere in the computational domain, where $\overline{Ma(\mathbf{x})}$ denotes the

time-average of $Ma(\mathbf{x})$. In the absence of obstacle in the flow, the well-known Hagen-Poiseuille equation [97] connects the pressure gradient along the channel as a function of the flow velocity along the axis of the channel. This relation does not hold in the presence of an obstacle, as its interaction with the flow alters the momentum balance. In this case, the average flow velocity in the channel can be directly related to the forcing amplitude by integrating the momentum balance equation over the whole computational domain, and averaging over time. As a result, we set $F_0 = 1.2 \times 10^{-6}$, leading to $\overline{Ma} \approx 4 \times 10^{-2}$ at the inlet, at the centre of the channel. This value has been estimated by measuring the longitudinal velocity over a duration $10^4 T_0$, where $T_0 = R/U_0$ with R the diameter of the cylinder and U_0 the estimated average velocity of the upstream flow. We find $U_0 = 0.02$. Furthermore, we checked that this estimate is representative of the order of magnitude of the average Mach number across the whole computational domain.

With F_0 and δx fixed, the remaining parameter is the relaxation parameter τ, determining the Reynolds number of the flow. As mentioned in section 2.1, this parameter is linked to the kinematic viscosity of the fluid. In lattice units, *i.e.*, $\Delta x = \Delta t = 1$, $\nu = \frac{1}{3}(\tau - \frac{1}{2})$. However, this parameter is critical with respect to the stability of the simulation. Indeed, the Lattice Boltzmann Method is known to become highly unstable as $\tau \to 1/2$. In order to set τ, we therefore gradually decreased τ from $\tau = 0.51$—a value high enough so that stability makes no doubt— until either the corresponding Reynolds number was considered high enough, or the stability limit was reached. In the following we set $\tau = 0.50005$, leading to $\nu = 1.67 \times 10^{-5}$ in lattice units. For this value, the flow features developed turbulence and typical Mach number fluctuations $\mathbf{E}[Ma^2] \approx 10^{-2} \ll 1$. A simulation of the flow over a duration $10^4 T_0$ showed no sign of instability.

On the basis of the present geometry, illustrated in figure 2.2, we define a Reynolds number based on the dimension of the obstacle:

$$Re_R = \frac{U_0 R}{\nu} \approx 2 \times 10^4.$$

By construction, the Reynolds number based on the cross section of the channel is $Re_H \approx 4 Re_R = 8 \times 10^4$.

We define the *turnover time* τ_0 as the typical timescale of advection by the flow. It is defined based on the diameter of the obstacle and the mean flow velocity as follows:

$$\tau_0 = \frac{R}{U_0}. \tag{2.7}$$

In addition, the simulation timestep Δt is

$$\Delta t = 1.4 \times 10^{-3} \frac{R}{U_0}.$$

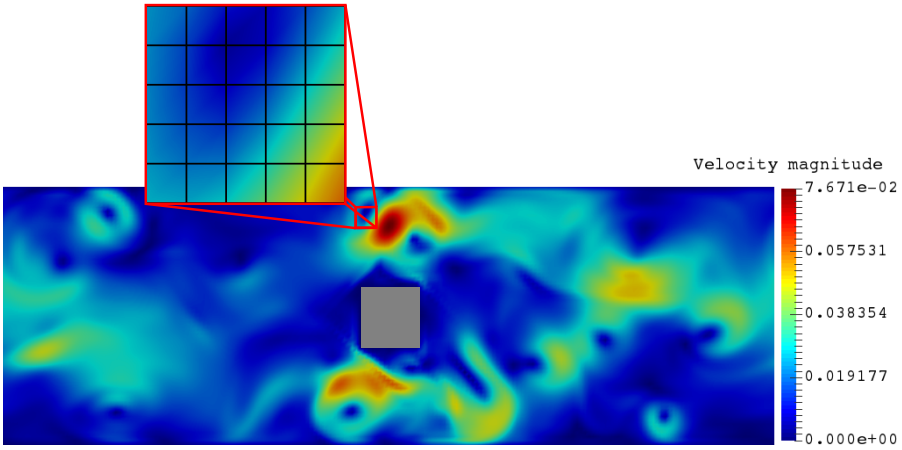


Figure 2.3: Velocity field in the computational domain for test flow (o). Denoting by R the diameter of the square cylinder, the lattice spacing is $\delta x = R/16$. The forcing amplitude is set to $F_0 = 1.2 \times 10^{-6}$, leading to a mean flow velocity $U_0 = 0.0017$. Finally, the relaxation pulsation w is set to $\omega = 1.9998$. The Reynolds number based on the mean flow velocity and the diameter of the square is $Re_R = 2500$. The zoom on the upper part of the figure displays the underlying lattice on which space is discretised and mesoscopic populations are advected following the Lattice Boltzmann Method, described in section 2.1.

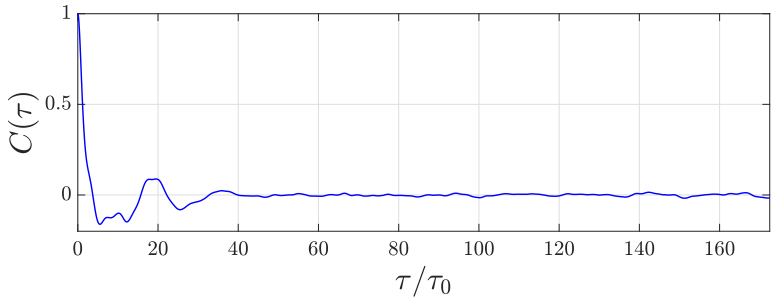
A typical velocity field is illustrated in figure 2.3, along with the underlying lattice over which space is discretised.

2.2.1.2 *The drag autocorrelation has a zero-valued integral*

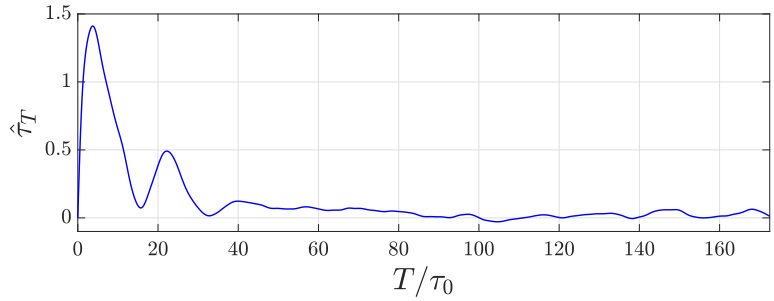
One of the main objective of this thesis is to establish a proof-of-principle for the application of a large-deviation algorithm to the computation of the statistics of rare fluctuations of the drag acting on an obstacle. In this section we make an important observation: the large-time integral of the autocorrelation function of the drag in test flow (o) is zero. We then relate this observation to the large-deviation properties of the large-time averaged drag and explain that the PDF of the drag does not verify a large deviation principle in this *specific* case. Please refer to chapters 1 and 4 for a discussion of the objectives of this work and large-deviation rate functions, respectively.

In the following we note $f_d(t)$ the drag acting on the square cylinder at a given time t. Let $C(\tau)$ be the *autocorrelation function* of the drag defined as

$$C(\tau) = \frac{\mathbb{E}[f_d(t) f_d(t+\tau)] - E[f_d]^2}{\sigma^2}, \tag{2.8}$$



(a) Autocorrelation of the drag force



(b) Finite time integral of the autocorrelation

Figure 2.4: **(a)** Autocorrelation function of the drag force f_d acting on the square obstacle immersed in the flow, as pictured in figure 2.3. The unit of time is the *turnover time* $\tau_0 = R/U_0$. On can see that the autocorrelation does not decay towards 0 but instead oscillates before vanishing at larger times. **(b)** Finite time-integral $\hat{\tau}_T$ as a function of the integration time, as defined in (2.9).

where σ is the *standard deviation* defined as $\sigma^2 = \mathbb{E}[f_d(t)^2] - E[f_d]^2$. Furthermore, let us define $\hat{\tau}_T$ as the time integral of the autocorrelation function:

$$\hat{\tau}_T = \frac{1}{\sigma^2} \int_0^T \left(\mathbb{E}[f_d(t) f_d(t+\tau)] - E[f_d]^2 \right) \mathrm{d}t. \tag{2.9}$$

This integral is expected to have a finite limit $\hat{\tau} = \lim_{t\to\infty} \hat{\tau}_t$. Indeed, the autocorrelation is expected to decay rapidly towards zero due to the effect of large-scale turbulent fluctuations. Therefore there exists a mixing timescale, or *correlation time* τ_c, from which $C(\tau) \approx 0, \forall \tau \geq \tau_c$. This properties are discussed further in this section and in chapter 3. The autocorrelation $C(\tau)$ is illustrated in figure 2.4a. It has been estimated from a long simulation of the flow over $10^5 T_0$, resulting in a timeseries for the drag acting on the obstacle. It shows that the autocorrelation displays oscillations before actually vanishing. In addition, figure 2.4b illustrates the convergence of the integral $\hat{\tau}_t$ in (2.9) as the integration time increases. Surprisingly, it suggests that $\hat{\tau} = \lim_{T\to\infty} \hat{\tau}_T = 0$. In the following we illustrate that his property leads to a pathological large deviation behaviour.

2.2.1.3 *The link with large deviations*

The large deviation principle for the time-averaged drag $F_T = \int_0^T f_d(t)\mathrm{d}t$ writes:

$$\mathcal{P}(F_T = a) \underset{T\to\infty}{\asymp} e^{-TI(a)}, \tag{2.10}$$

where $I(a)$ denotes the *rate function*. See chapter 1 for a discussion of Large Deviation Theory. Considering small fluctuations around the average $\mathbb{E}[F_T] = \mu$, we expand the rate function as follows:

$$I(a) \approx \frac{I''(\mu)}{2}(a-\mu)^2, \tag{2.11}$$

where we used that $I(\mu) = I'(\mu) = 0$. The large deviation principle (2.10) turns to

$$\mathcal{P}(F_T = a) \underset{T\to\infty}{\asymp} \exp\left[-T\frac{(a-\mu)^2}{2}I''(\mu)\right], \tag{2.12}$$

illustrating the Gaussian behaviour for small fluctuations around the average. The variance of F_T is therefore linked to the curvature of the rate function as follows

$$\sigma_T^2 \underset{T\to\infty}{\sim} \frac{1}{TI''(\mu)}. \tag{2.13}$$

In the following we connect the curvature $I''(\mu)$ to the time-integral of the autocorrelation $\hat{\tau}$. The variance σ_T can be written as

$$\begin{aligned}\mathbb{E}[(F_T - \mathbb{E}(F_T))^2] &= \mathbb{E}\left[\left(\frac{1}{T}\int_0^T (f_d(t) - \mathbb{E}(f_d)\mathrm{d}t\right)^2\right] \\ &= \mathbb{E}\left[\frac{1}{T^2}\int_0^T \mathrm{d}t_1 \int_0^T \mathrm{d}t_2 (f_d(t_1) - \mathbb{E}[f_d])(f_d(t_2) - \mathbb{E}[f_d])\right]\end{aligned} \tag{2.14}$$

With the change of variable $(t_1, t_2) \to (t, t+\tau)$, (2.14) leads to

$$T\sigma_T = 2\int_0^T \mathbb{E}[f_d(t+\tau)f_d(t)] - \mathbb{E}[f_d]^2\mathrm{d}\tau = \frac{\hat{\tau}_T\sigma^2}{2}. \tag{2.15}$$

Therefore, assuming that $\lim_{T\to\infty}\hat{\tau}_T = 0$, equation (2.13) leads to

$$I''(\mu) = +\infty \tag{2.16}$$

2.2.1.4 *Momentum balance in a channel with periodic boundaries*

We now explain the observations made in the previous section, based on the conservation of momentum in the computational domain. We show that, with periodic boundaries at the inlet/outlet of the channel, the total drag acting on the obstacle can be written as a linear

function of the first derivative of the velocity average over the whole computational domain. We then show that this entails that the integral of the drag autocorrelation function $\hat{\tau}$ vanishes at large times.

We start by recalling the Navier-Stokes Equations for an incompressible flow:

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} = \nu \Delta \mathbf{u} + F_0 \mathbf{e}_x - \nabla P \tag{2.17a}$$

$$\nabla \cdot \mathbf{u} = 0 \tag{2.17b}$$

$$\mathbf{u}(\mathbf{x}) = 0, \quad \forall \mathbf{x} \in \partial\mathcal{D} \tag{2.17c}$$

where $\mathbf{u}$ denotes the velocity field, ν the kinematic viscosity and P the pressure field. The solid boundaries are denoted by $\partial\mathcal{D}$ and consist of the top and bottom walls of the channel as well as the surface of the obstacle. See figure 2.2 for a sketch of the computational domain. Let $\mathbf{U}(t)$ be the velocity integrated over the whole computational domain, which we denote as $\mathcal{C}_d$ in the following:

$$\mathbf{U}(t) = \int_{\mathcal{C}_d} \mathbf{U}(\mathbf{r}, t) \mathrm{d}\mathbf{r}$$

We now write an equation for the evolution of $\mathbf{U}$ by integrating (2.17a) over the whole computational domain. To begin with, we write the integral of the viscous and pressure terms as a surface integral over the boundaries:

$$\int_{\mathcal{C}_d} (\nu \Delta \mathbf{u} - \nabla P) \mathrm{d}\mathbf{r} = \nu \int_{\partial\mathcal{D}} \nabla \mathbf{u} \cdot \mathbf{n} - \int_{\partial\mathcal{D}} P \cdot \mathbf{n}$$

where $\mathbf{n}$ is a unit vector tangential to $\partial\mathcal{D}$. This integral represents the total force applied by the fluid on the boundaries. It can therefore be split into the contribution of the force applied on the channel walls and the force applied on the obstacle. We denote by $\mathbf{F}_{walls}$ and $\mathbf{F}_{obs}$ these two contributions, respectively. As a result:

$$\mathbf{F}_{walls} + \mathbf{F}_{obs} = \int_{\mathcal{C}_d} (\nu \Delta \mathbf{u} - \nabla P) \mathrm{d}\mathbf{r} = \nu \int_{\partial\mathcal{D}} \nabla \mathbf{u} \cdot \mathbf{n} - \int_{\partial\mathcal{D}} P \cdot \mathbf{n}$$

Unless specified otherwise, the force density is constant and homogeneous across the whole domain. Its volume integral therefore simply yields:

$$\int_{\mathcal{C}_d} F_0 \mathbf{e}_x \mathrm{d}\mathbf{r} = F_0 V$$

where $V = H \times L$ denotes the total volume of the computational domain, see figure 2.2. Eventually, the integration of equation (2.17) over the computational domain $\mathcal{C}_d$ leads to

$$\frac{\mathrm{d}\mathbf{U}}{\mathrm{d}t} = -\mathbf{F}_{walls} - \mathbf{F}_{obs} + F_0 V \mathbf{e}_x \tag{2.18}$$

which projection on the channel axis yields

$$\frac{\mathrm{d}\mathbf{U_x}}{\mathrm{d}t} = -\mathbf{F}_{walls} \cdot \mathbf{e}_x - f_d + F_0 V \tag{2.19}$$

The contribution $\mathbf{F}_{walls} \cdot \mathbf{e}_x$ results from viscous friction at the top and bottom walls of the channel. In contrast, the drag f_d acting on the square is largely dominated by the pressure difference between the front and the back of the obstacle. Based on the relatively high Reynolds number, $Re_R = 2500$, we make the approximation:

$$\mathbf{F}_{walls} \cdot \mathbf{e}_x \ll f_d$$

The relative contribution of pressure effects versus viscous effects to the overall drag is discussed in chapter 3. Under this approximation, the drag f_d can be written as

$$f_d = F_0 V - \frac{\mathrm{d}U_x}{\mathrm{d}t} \tag{2.20}$$

We now show that (2.20) entails that the large-time limit of the integral of the drag autocorrelation is zero. Note that, through time-averaging (2.20), one gets

$$\mathbb{E}[f_d] = \lim_{T\to\infty} \frac{1}{T} \int_0^T f_d(t)\mathrm{d}t = F_0 V.$$

The integral of the autocorrelation function $C(\tau)$ therefore reads

$$\begin{aligned} \hat{\tau}_T &= \tfrac{1}{\sigma^2} \textstyle\int_0^T \left(\mathbb{E}[f_d(t) f_d(t+\tau)] - (F_0 V)^2\right) \mathrm{d}t \\ &= -F_0 V \mathbb{E} \textstyle\int_0^T \dot{U}_x(t)\mathrm{d}t - F_0 V \mathbb{E} \int_0^T \dot{U}_x(0)\mathrm{d}t + \mathbb{E} \int_0^T \dot{U}_x(t)\dot{U}_x(0)\mathrm{d}t \end{aligned} \tag{2.21}$$

The first two terms are actually zero, invoking statistical stationarity of the flow:

$$\mathbb{E} \int_0^T \dot{U}_x(t)\mathrm{d}t = \mathbb{E}[U(T) - U(0)] = 0$$

and

$$\mathbb{E} \int_0^T \dot{U}_x(0)\mathrm{d}t = \mathbb{E}[\dot{U}_x(0)] = \lim_{T\to\infty} \frac{1}{T} \int_0^T \dot{U}_x(t)\mathrm{d}t = 0$$

The third term corresponds to the temporal correlation of the longitudinal velocity with its derivative. It reads

$$\mathbb{E} \int_0^T \dot{U}_x(t)\dot{U}_x(0)\mathrm{d}t = \mathbb{E}[U_x(T)\dot{U}_x(0)] - \mathbb{E}[U_x(0)\dot{U}_x(0)] \underset{T\to\infty}{\to} 0 \tag{2.22}$$

This last limit entails

$$\lim_{T\to\infty} \hat{\tau}_T = 0 \tag{2.23}$$

2.2.1.5 *Conclusion: a pathological case*

As far as large-deviation theory is concerned, the drag acting on a single obstacle immersed in a channel flow with periodic boundaries is a pathological case. Indeed, we showed in this section that the integral of the autocorrelation function has a zero value. In section 2.2.1.2 we showed that in this case, the PDF describing the statistics of the drag acting on the obstacle displays a pathological large deviation behaviour. We stress that the calculations presented in this section rely on two ingredients. First, the periodic boundaries at the inlet/outlet of the channel lead to equation (2.19), which states that the interaction of the flow with the boundaries is solely balanced by the forcing term. Second, we neglected the interaction with the top and bottom walls of the channel compared to the interaction with the obstacle. The reason for that is twofold. First, the Reynolds number is high, indicating that pressure effects dominate viscous effects. Second, we assume that the obstacle is *bluff*. It means that its drag is dominated by the pressure difference between its upstream section and its downstream section. In contrast, this approximation is not justified for *streamlined* bodies, for which the pressure difference is very small, even at relatively high Reynolds numbers, and the drag has an important viscous contribution with respect to the pressure contribution. See the introduction of chapter 3 for a discussion of bluff and streamlined bodies.

As a result, the geometry discussed in this section, illustrated in figure 2.2, is clearly not appropriate for the test of a large-deviation algorithm. A workaround is to introduce a second obstacle in the channel. In this case, the reasoning presented in this section holds true for the system consisting of the *two* obstacle. As a consequence, the PDF for the drag acting on an *individual obstacle* will verify a large-deviation principle. This configuration is presented in the next section.

2.2.2 *Test flow (1): flow in a periodic channel past a tandem of square cylinders*

In the previous section we showed that the flow past a single square cylinder in a periodic channel is a very special case in which the PDF of the drag acting on the obstacle does not verify a large-deviation principle. In this section we describe a simple modified version of the flow presented in the previous section, in which *two* obstacles are introduced in the channel. In this way, the PDF of the drag for a

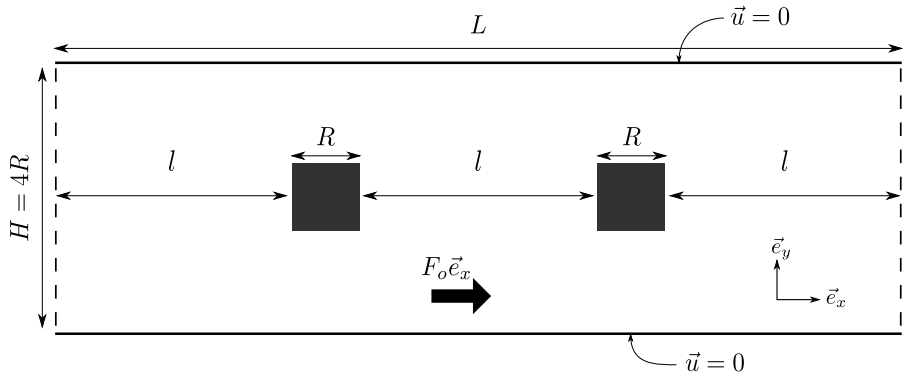


Figure 2.5: Sketch of the computational domain for test flow (1). The computational domain is periodic along the channel axis and the flow is forced through a constant, homogeneous force density of amplitude F_0. No-slip boundary conditions are imposed on both the surface of the obstacles and the walls of the channel.

single obstacle verifies a large-deviation principle. Throughout this manuscript, this flow will be referred to as *test flow (1)*.

2.2.2.1 *Description of test flow (1)*

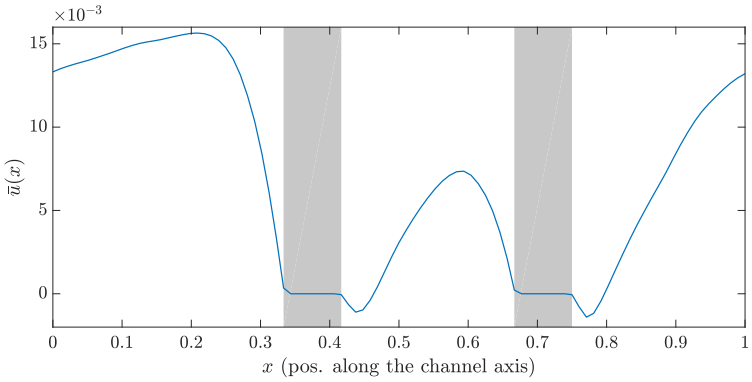
The corresponding geometry is illustrated in figure 2.5. The computational domain is periodic along the channel axis and the flow is forced through a constant, homogeneous force density $F_0\mathbf{e}_x$ acting across the whole domain, where $\mathbf{e}_x$ is a unit vector which direction is parallel to the channel axis. Note that the two square cylinders have the same diameter R.

The numerical parameters for test flow (1) are kept the same as for the flow presented in the previous section. That is $\delta x = R/16$, $F_0 = 1.2 \times 10^{-6}$ and $\tau = 0.50005$. Following figure 2.5, the computational domain consists in a grid of 129×513 grid points. Figure 2.6a illustrates the average longitudinal velocity profile along the channel axis. It results from the time-average over a simulation of the flow in stationary regime over a duration $500\tau_0$, in which the longitudinal velocity was measured along the channel axis.

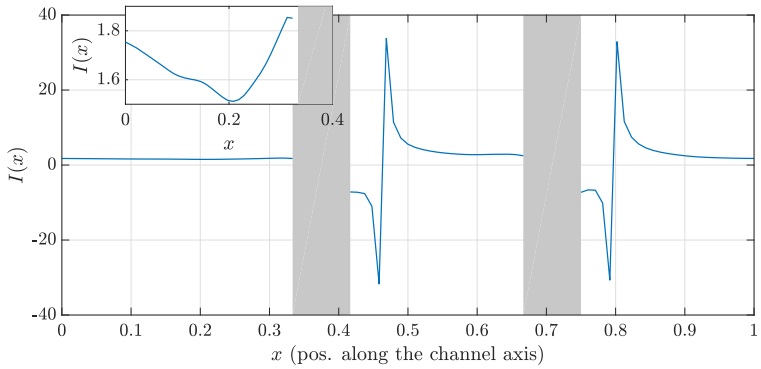
In addition, figure 2.6b displays the *turbulence intensity* profile along the channel axis. The turbulence intensity I is defined as the ratio between the root mean square fluctuations of the velocity and the average velocity. That is:

$$I(\mathbf{x}) = \frac{\sqrt{\overline{(u_x(\mathbf{x}) - \overline{u_x(\mathbf{x})})^2 + u_y(\mathbf{x})^2}}}{\overline{u(\mathbf{x})}} \tag{2.24}$$

Where $\overline{u_i(\mathbf{x})} = \lim_{T\to\infty} \frac{1}{T}\int_0^T u_i(\mathbf{x}, t)\mathrm{d}t$ denotes the average velocity at position $\mathbf{x}$, and u refers to the velocity magnitude $u(\mathbf{x}) = \sqrt{u_x^2(\mathbf{x}) + u_y^2(\mathbf{x})}$. We define the mean flow velocity U_0 as the max-

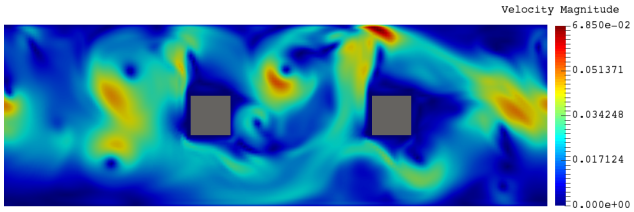


(a) Average velocity profile along the channel axis

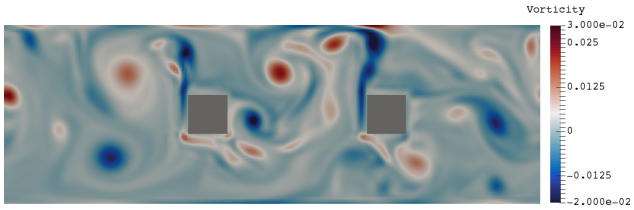


(b) Average turbulence intensity profile along the channel axis

Figure 2.6: Average longitudinal velocity $\overline{u_x}$ and turbulent intensity I profiles along the centre of the channel for test flow (1). The velocity was measured and averaged in stationary regime over $500\tau_0$. The x axis is the position along the centre of the channel with respect to the inlet. The unit of length is the length L of the computational domain.



(a) Velocity magnitude



(b) Vorticity

Figure 2.7: Typical velocity and vorticity fields for test flow (1)

imum average velocity along the channel axis. From 2.6a, it reads $U_0 \approx 1.6 \times 10^{-2}$ in lattice units. The Reynolds number based on the diameter of the square R is therefore

$$Re_R = \frac{U_0 R}{\nu} \approx 16000$$

The corresponding velocity and vorticity fields are illustrated in figure 2.7.

Finally, the simulation timestep is

$$\Delta t = 10^{-3} \frac{R}{U_0} \tag{2.25}$$

The numerical simulation is implemented with the help of the `pipeLBM` C++ library developed for this thesis project. See appendix A for more details about the `pipeLBM` library. Simulation of one eddy turn-over time τ_0 is achieved in roughly 1.5 seconds on a single Intel Core i7-4800MQ CPU processor core with a 2.70 GHz clock rate.

2.2.2.2 *Statistics for the drag acting on the square cylinders*

We now describe the statistical properties of the drag acting on both square cylinders embedded in the channel. This description is based on a very long simulation of the flow over $T_{tot} = 4 \times 10^6 \tau_0$, that results in a drag timeseries for both obstacles: $\{f_d^{(1)}\}_{0 \leq t \leq T_{tot}}$ and $\{f_d^{(2)}\}_{0 \leq t \leq T_{tot}}$ where the superscripts (1) and (2) denote the upstream obstacle and downstream obstacle, respectively.

Figure 2.8 displays the resulting estimate of the PDF for the drag fluctuations for both the upstream and downstream obstacles. It illustrates that both PDF are strongly non-Gaussian. Furthermore, they look very similar for positive fluctuations. Remarkably, extreme positive

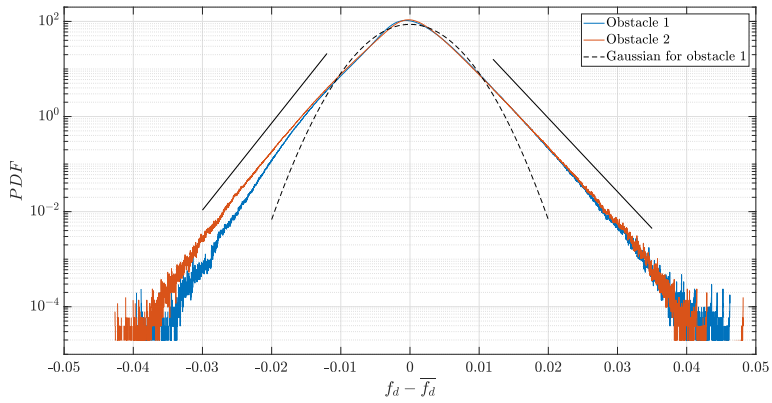


Figure 2.8: Estimate of the Probability Density Function for the fluctuations of the drag acting on both upstream and downstream square cylinders embedded in test flow (1), illustrated in figures 2.5 and 2.7. Both estimates have been computed over a timeseries spanning $T_{tot} = 4 \times 10^6 \tau_0$. Note that this figure describes the PDF of the *fluctuations* $f_d - \overline{f_d}$ where $\overline{f_d}$ is the average computed over the whole timeseries.

fluctuations seem to be well described by an exponential PDF. The measured variance in both cases lead to very similar values, indicating that both obstacles feature typical drag fluctuations with similar amplitude. Extreme negative fluctuations for the downstream square seem to be described by an exponential PDF as well, although it is less clear for the upstream obstacle, for which extreme negative fluctuations are less likely. In addition, both PDF are skewed toward positive fluctuations with a skewness coefficient $\gamma_1^{(1)} = 0.14$ and $\gamma_1^{(2)} = 0.035$.

Furthermore, we find that the average drag on the upstream square is greater than on the downstream square. Note that this cannot be seen on figure 2.8 as it shows the PDF of fluctuations around the average value. This difference is significant: the average drag acting the the downstream obstacle is 35% lower than the one acting on the upstream square. Such a difference is explained by the fact that the downstream obstacle sits in the wake of the upstream one, which corresponds to a region of lower pressure.

Figure 2.9 displays the estimate for the correlation function $C(\tau)$ based on the timeseries for both obstacles. It shows that in both cases the drag decorrelates exponentially in time, a property that will be useful to describe extremes of the averaged drag in chapter 3. Based on figure 2.9, we define the *correlation time* τ_c as the zero crossing time of the autocorrelation function. We find $\tau_c \approx 4\tau_0$.

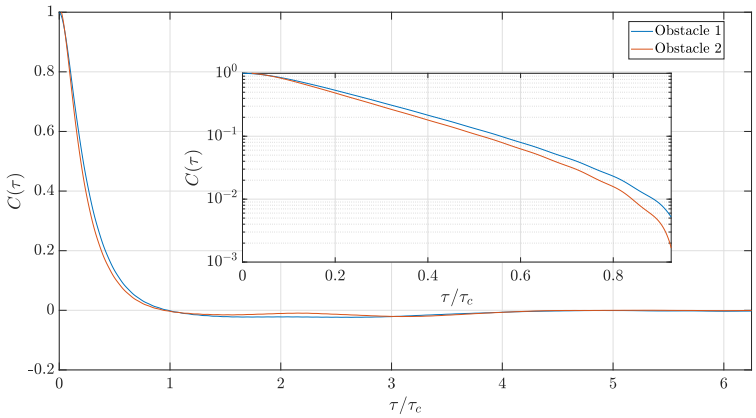


Figure 2.9: Estimate of the correlation function $C(\tau)$ for both drag acting the squares cylinders embedded in test flow (1), illustrated in figures 2.5 and 2.7. The autocorrelation is defined as $C(\tau) = (\mathbb{E}[f_d(t)f_d(t+\tau)] - \mathbb{E}^2[f_d(t)])/\sigma^2$ where $\sigma^2 = \mathbb{E}[f_d^2] - \mathbb{E}[f_d]^2$ is the standard deviation. The *correlation time* τ_c is defined as the zero-crossing time of the autocorrelation. Both autocorrelation functions have been computed over a timeseries spanning $T_{tot} = 10^6\tau_c$.

2.3 TEST FLOW (2): GRID-GENERATED TURBULENCE PAST A SQUARE CYLINDER IN A CHANNEL

In the previous section, we described test flow (1), designed to serve as a test case for the investigation of rare drag fluctuations and the use of rare event algorithms. In test flow (1), turbulence is triggered through the interaction of the square cylinders with their own wake. This geometry is very practical with respect to its numerical simulation. Indeed, it consists in flat walls modelled by no-slip boundaries and periodic inflow and outflow. However, test flow (1) is far from any realistic industrial or environmental configurations.

In this section we describe a second test-case for the application of rare events algorithms to the simulation of extreme drag fluctuations. Similarly to test flow (1), it consists of the two-dimensional flow past a square cylinder in a channel. This time however, there is no periodicity along the channel direction. A steady parabolic velocity profile is imposed at the inlet, and the channel is open at the outlet. Turbulence is generated by means of a grid following the inlet of the channel. Throughout this manuscript, we refer to this flow as test flow (2).

In section 2.3.1 we describe test flow (2) in more details. We first describe the boundary conditions at the inlet and outlet, as well as numerical parameters. Then, flow parameters such as Reynolds numbers, mean flow velocity and turbulence intensity are discussed. In section 2.3.2, the statistics of the drag acting on the square cylinder are described by means of a very long simulation of the flow. We

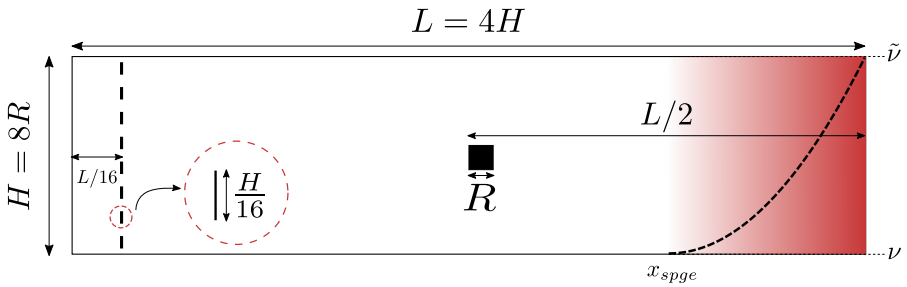


Figure 2.10: Sketch of the computational domain for test flow (2). A steady, Poiseuille velocity profile is imposed at the inlet (left-hand side of the domain). It is followed by a grid consisting of small no-slip boundaries which longitudinal extent is one grid spacing δx. The upstream side of the obstacle is positioned at one half of the channel length. At the outlet (right-hand side of the domain), an open boundary is implemented, based on a linear interpolation of the longitudinal velocity along the channel axis. Additionally, a sponge layer is introduced in the vicinity of the outlet in order to damp spurious density waves emitted at the outlet boundary [168]. In this region the viscosity of the fluid is artificially increased from its original value ν to a higher value $\tilde{\nu} \approx 10^3 \nu$. It is represented by the red area.

illustrate that the autocorrelation function and PDF have properties similar to test flow (1). The autocorrelation displays an exponential decay and both tails of the PDF are well described by an exponential PDF. In addition, we highlight the asymmetry of the PDF and show that it is related to the interaction between the flow and the obstacle. More precisely, we compute the PDF for the drag acting on the surface of the obstacle, *without actually introducing the obstacle*. We illustrate that it results in a symmetric, algebraic PDF.

2.3.1 *Description of test flow (2)*

The computational domain for test flow (2) is sketched in figure 2.10. Similarly to test flow (1), it consists in a two-dimensional channel in which a square obstacle is introduced. However, the flow is not periodic along the channel axis. Instead, a parabolic velocity profile $\mathbf{u}_{in}(y)$ is imposed at the inlet:

$$\mathbf{u}_{in}(y) = -\frac{4u_0}{H}(y - H)y\mathbf{e}_x \tag{2.26}$$

where $\mathbf{e}_x$ is a unit vector aligned with the channel axis. Therefore, the velocity u_0 is imposed at the centre of the channel.

At the outlet, an open boundary condition is implemented. That is, we impose the velocity and pressure at the outlet in a way that is consistent with the incoming flow from the bulk of the computational domain. More precisely, let x denote the distance from the inlet, located at $x = 0$. Furthermore, let us denote by $\mathbf{u}_{out}(y)$ the velocity field at

the outlet, corresponding to the right-hand side of 2.10, located in $x = L$. The velocity field at the outlet is computed by means of a first order linear extrapolation based on the nearest neighbours and second nearest neighbours lattice nodes at $L - \delta x$ and $L - 2\delta x$:

$$\mathbf{u}_{out}(y) = 2\mathbf{u}(x = L - \delta x, y) - \mathbf{u}(x = L - 2\delta x, y). \tag{2.27}$$

where δx is the distance between two adjacent lattice nodes. The value of this distance is discussed below. The extrapolated velocity is then imposed using finite-difference regularised boundary conditions [100]. Please refer to section 2.1 for a discussion of regularised boundary conditions. Furthermore, see the chapter 5 of [92] and references therein for a discussion of open boundaries in the LBM.

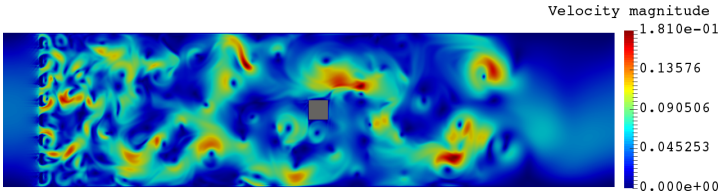
Because the LBM recovers the incompressible Navier-Stokes equations in the weakly compressible limit, open boundaries are known to be responsible for the reflection of density waves generated following the initial timesteps [84, 148]. The persistence of these spurious density waves in the domain can significantly alter the accuracy of the simulation and generate numerical instabilities. A common way to mitigate the reflection of such density waves at an open boundary is to introduce a *sponge layer* [168, 169]. It is a region in the vicinity of the boundary where the viscosity is artificially increased, so as to damp the waves. In practice, the relaxation pulsation is decreased quadratically to a fraction of its value from a given distance $L - x_{spge}$ of the outlet [168]. We denote by ω_{spge} the relaxation pulsation inside the sponge layer. It is defined as

$$w_{spge}(x) = \left(1 - 0.999\frac{(x - x_{spge})^2}{(L - x_{spge})^2}\right)\omega, \quad x_{spge} \leq x \leq L. \tag{2.28}$$

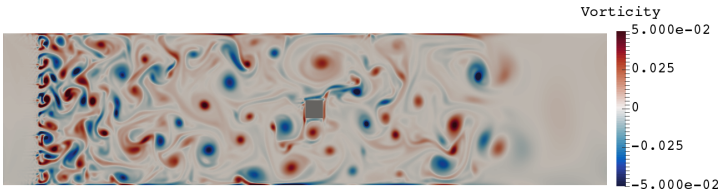
2.3.1.1 *Numerical parameters*

We now briefly discuss the choice of the numerical parameters for test flow (2),*i.e.* the lattice spacing δx, the inlet velocity U_0 and the relaxation parameter τ. The choice of numerical parameters for test flow (2) follows the same guidelines as for test flow (1), described on page 26. Similarly to test flow (1), space is discretised on a regular, isotropic lattice. The lattice spacing is set to $\delta x = R/16$, where R denotes the diameter of the square cylinder. Following figure 2.10, the computational domain consists in a grid of 129×513 points.

The characteristic inlet velocity u_0 is chosen small enough so that $Ma \ll 1$, where the inlet Mach number $Ma = u_0/c_s$ is the ratio between the inlet velocity and the speed of sound. As mentioned in section 2.2.1.1, the reason for this choice is twofold. First of all, the incompressible limit is achieved for $Ma \ll 1$. Second, the LBM can be shown to recover the incompressible Navier-Stokes equations with



(a) Velocity magnitude



(b) Vorticity

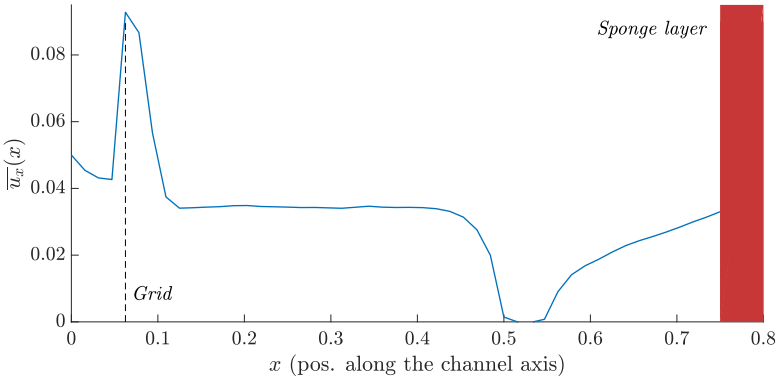
Figure 2.11: Snapshots for the velocity and vorticity fields in the computational domain for test flow (2). The Reynolds number based on the mean upstream flow velocity and the diameter of the cylinder is $Re \approx 1700$. The flow is computed using the central-moments based Lattice Boltzmann Method [143, 144]. Denoting by R the diameter of the square, the lattice spacing is $\delta x = R/16$. At the inlet a steady parabolic profile is imposed with characteristic velocity $u_0 = 0.05$. Finally the relaxation parameter ω is set to 1.996. Please refer to section 2.1 for more details about the Lattice Boltzmann Method.

errors scaling like Ma^2. In the following, we set $U_0 = 5 \times 10^{-2}$ in lattice units.

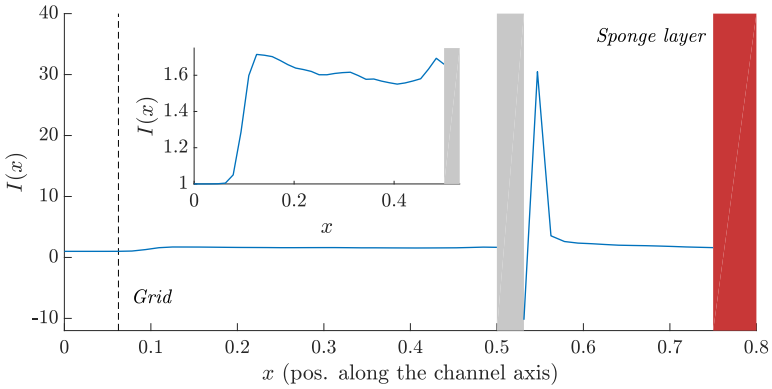
Lastly, the relaxation parameter is set to $\tau = 0.501$. As pointed out in section 2.1, values of τ too close to $1/2$ can cause numerical instabilities. More precisely, recall that $\tau = 1/2 + \frac{\nu}{c_s^2 \Delta t}$, and $\Delta x = \sqrt{3} c_s \Delta t$. With Δx and c_s fixed, the value of τ directly sets the value of the kinematic viscosity of the fluid. Therefore, with Δx and U_0 fixed, we set τ so that the flow displays developed turbulence without instabilities. In a way similar to section 2.2.2, we progressively decreased τ until the flow displayed a satisfactory turbulence intensity. Figure 2.11 illustrate the corresponding velocity and vorticity fields in the computational domain.

The Reynolds number for test flow (2) can be defined in several ways, depending on which typical length and velocity it is based on. To be consistent with test flow (1), we define the Reynolds number based on the diameter of the square and the mean flow velocity *past the grid*.

$$Re = \frac{U_0 R}{\nu} \approx 1700 \tag{2.29}$$



(a) Average velocity profile along the channel axis



(b) Average turbulence intensity profile along the channel axis

Figure 2.12: Average velocity $\overline{u_x}$ and turbulent intensity I profiles along the centre of the channel for test flow (2). The velocity was measured and averaged in stationary regime over $450\tau_0$. The x axis is the position along the centre of the channel with respect to the inlet. The unit of length is the length L of the computational domain. The red shaded area denotes the start of the sponge layer, that extends until $x = 1$. In this region the viscosity is gradually increased in order to damp density waves reflected at the outlet boundary. The grey shaded area denotes the obstacle.

where U_0 is the average velocity along the channel axis, at equal distance of the grid and the upstream side of the obstacle. We find $U_0 \approx 0.036$.

In addition we explicit the simulation timestep Δt in units of the turnover time $\tau_0 = R/U_0$:

$$\Delta t = 2.2 \times 10^{-4} \frac{R}{U_0} \tag{2.30}$$

The numerical simulation is implemented with the help of the `pipeLBM` C++ library developed for this thesis project. See appendix A for more details about the `pipeLBM` library. Simulation of one eddy turn-over time τ_0 is achieved in roughly 16 seconds on a single Intel Core i7-4800MQ CPU processor core with a 2.70 GHz clock rate. Figure 2.11 illustrates typical realisations of both velocity and vorticity fields in the computational domain.

Figure 2.12a illustrates the average velocity profile along the channel axis. It results from the time-average over a simulation of the flow in stationary regime over a duration $450\tau_0$, in which the longitudinal velocity was measured at different locations along the channel axis. In addition, figure 2.12b displays the *turbulence intensity* profile along the channel axis, which represents the relative importance of velocity fluctuations with respect to the mean flow velocity. Its definition is given in equation (2.24) on page 34.

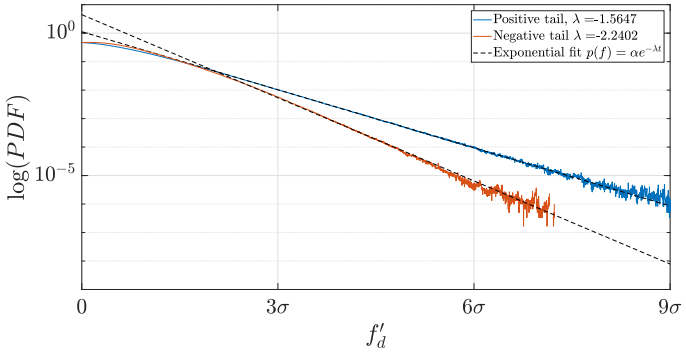
2.3.2 *Statistics for the drag on the obstacle*

In the previous section we described the numerical simulation of test flow (2), which geometry is illustrated in figure 2.10, as well as several aspects of the flow itself. We now turn to a description of the statistics for the fluctuations of the drag f_d acting on the square cylinder embedded in test flow (2). This description is based on a very long simulation of the flow over $T_{tot} = 4.5 \times 10^6 \tau_0$, leading to a long timeseries $\{f_d(t)\}_{0 \leq t \leq T_{tot}}$ of the drag acting on the obstacle.

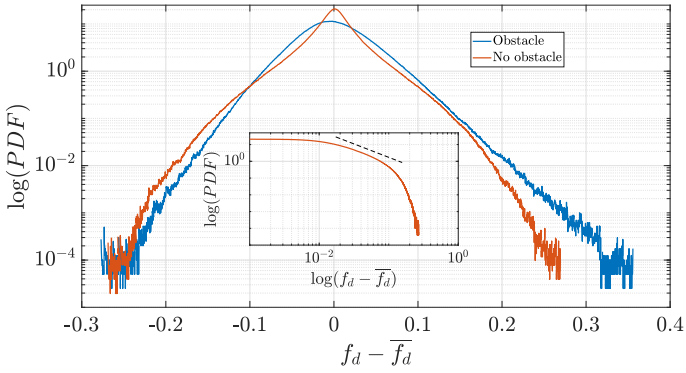
The corresponding estimate of the PDF is shown in figure 2.13. It is found to be asymmetric and skewed towards positive fluctuations. Additionally, figure 2.13a clearly illustrates that both negative and positive tails are remarkably well described by an exponential PDF. Recall that similar observations were made for test flow (1) in section 2.2.2.2.

2.3.2.1 *The influence of the flow-obstacle interaction on the statistics of the drag*

In order to further investigate the properties of the PDF describing the drag fluctuations, we perform the following experiment. We perform a similar simulation of test flow (2) over the same duration T_{tot}, however without embedding the obstacle in the channel. We therefore



(a)



(b)

Figure 2.13: Estimate of the Probability Density Function describing the statistics of the fluctuations of the drag f_d acting on the obstacle embedded in test flow (2). This estimate has been computed from a very long simulation of the flow resulting in a drag timeseries spanning $4.5 \times 10^6 \tau_0$. **(a)** Illustration of the asymmetry of the distribution, which is skewed towards positive fluctuations. It compares both negative and positive tails of the PDF, where the negative tail is represented for positive values by means of a symmetry operation with respect to the Y-axis. Furthermore, it shows that both tails are very well described by an exponential PDF. The quantity represented along the X-axis is the reduced-centred drag fluctuation: $f'_d = (f_d - \overline{f_d})/\sigma$ where $\overline{f_d}$ and σ are the average drag and standard deviations, respectively, computed over the whole timeseries. **(b)** Comparison of the PDF describing the fluctuations of the drag with the PDF describing the drag acting on a square cylinder that does not interact with the flow. One can see that the statistics are dramatically modified. The PDF in the case of the virtual obstacle is symmetric and its tails are well described by a power-law PDF.

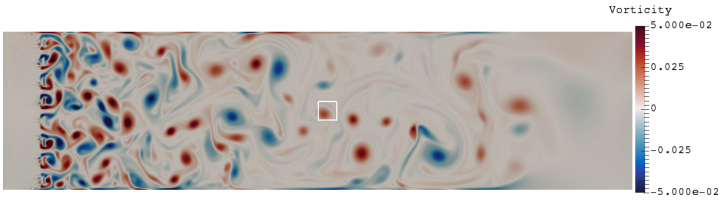


Figure 2.14: Numerical simulation of test flow (2) in which the obstacle does not interact with the surrounding flow. In this case the drag is simply computed over the surface represented by the white square. We stress that in this case no conditions are imposed on the velocity on the surface. Figure 2.13b shows that the statistics for this quantity are very different from the statistics describing the drag on a square that actually interacts with the flow, *i.e.* with no-slip boundary conditions for the velocity on the surface.

compute the drag acting on the surface of a *virtual* obstacle, that does not interact with the surrounding flow. This experiment is illustrated in figure 2.14.

The resulting estimate of the PDF is shown in figure 2.13b. It is compared with the estimate resulting from the simulation of test flow (2). A striking difference is that the PDF describing the drag on the virtual square cylinder is symmetric. As a result, drag fluctuations of a given amplitude below the average are as much as likely as fluctuations of the same amplitude above the average. Moreover, the probability of extreme drag fluctuations decays faster in the case of the non-interacting obstacle. As a matter of fact, it is illustrated in 2.13b that the corresponding PDF is well described by an algebraic PDF.

2.4 CONCLUSION

In this chapter we introduced two test cases for the investigation of extreme fluctuations of the drag force acting on a square cylinder immersed in a turbulent flow. The geometry of both test flow (1) and test flow (2) was deliberately chosen as simple as possible. The reason for this choice is twofold. First, both test flows must be computed over very long durations. As a result, a simple geometry is required for the simulation to be performed in a reasonable amount of time. Second, we will use the test flows as test cases for rare events algorithms. However, both test flow (1) and test flow (2) show enough complexity to provide a solid base for the assessment of the relevance of rare algorithms based approaches in CFD problems. We believe that the application of rare event techniques to these simple flows will lay the groundwork for future complex applications.

At the heart of the numerical simulations discussed in this section—and throughout this manuscript— is the Lattice Boltzmann Method. This relatively recent approach to Computational Fluid Dynamics receives increasing interest from scientists in a wide variety of fields,

and therefore benefits from an ever-growing community of users. The research activity around the Lattice Boltzmann Method is intense, as the method is particularly attractive for a large range of applications. Among them is the simulation of flows with simple geometries aligned with the lattice. For such flows, the algorithmic simplicity of the LBM leads to minimal implementation efforts with respect to conventional CFD methods. In this project we make use of an enhanced formulation of the LBM, called the central-moments based Lattice Boltzmann Method [143]. It yields exceptional stability, and therefore allows for the simulation of flows with high Reynolds numbers.

During the early stages of this thesis, test flow (1) was first implemented using the conventional LBGK [31] scheme, see section 2.1. Because of constraints on the stability of the simulation, only rather low Reynolds number could be achieved. For such low Reynolds numbers, around $Re \approx 700$, the evolution remains correlated for long times, as a result of the periodicity of the domain. Because of the difficulties associated with a large correlation time—mostly the large computational burden of simulating a large number of correlation times—test flow (2) was introduced, using the central-moments based LBM method. Test flow (1), as presented in this chapter, was only designed at a later stage of this work. As a result, extreme events in test flow (1) have not been addressed yet. *In the remainder of this manuscript, we therefore concentrate on test flow (2.)*

In the following chapter we describe the study of extreme drag fluctuations of the drag acting on the square cylinder embedded in test flow (2), on the basis of a very long simulation of the flow dynamics. In the subsequent chapters we describe the application of both the Giardina–Kurchan–Tailleur–Lecomte and Adaptive Multilevel Splitting to test flow (2) and discuss the corresponding results. More precisely, we will use them to sample extreme fluctuations of the instantaneous drag f_d, as well as the averaged drag, acting the square cylinders embedded in test flow (2). In particular we will investigate how efficiently both algorithms are capable of sampling and simulating flow dynamics leading to extremes, compared to a direct approach consisting in brute force sampling based on a very long simulation.

3

DYNAMICS OF EXTREME FLUCTUATIONS OF THE DRAG ACTING ON A BLUFF BODY IMMERSED IN A TWO-DIMENSIONAL TURBULENT FLOW

Solid bodies can be classified into two categories, depending on their shape, as illustrated in figure 3.1. On the one hand, for *streamlined bodies*, the boundary layer remains attached to the surface of the body. Consequently, the wake is characterised by a rather small spatial extent. Therefore, a high pressure is recovered near the downstream extremity of the object, referred to as the *base*. As a result, the force resulting from the pressure difference between the *forebody*—the upstream point of the body— and the base is small and drag essentially originates from the viscous friction induced by the flow in the boundary layer. Examples of streamlined bodies include aerofoils, race cars and aero bike helmets. On the other hand, *bluff bodies* are bodies which shape favours boundary layer separation, often simply mentioned as *flow separation*, that prevents the re-compression of the flow in the vicinity of the base. As a consequence, the base pressure is much lower than the forebody pressure, which leads to high levels of *pressure drag*, also referred to as *form drag*, due to its connection with the shape of the body. Roughly speaking, flow separation is favoured by large cross-sectional areas, as well as sharp corners. Typical examples of bluff and streamlined bodies are pictured in figure 3.1.

As a matter of fact, most bodies are bluff bodies. Therefore accurate characterisation of flows past bluff bodies is of major importance in a wide variety of fields. Examples include wind loading on tall buildings, bridges [85, 91, 159] or wind turbines [74, 87, 114]. In ground vehicle aerodynamics, one of the major objective is reduction of the pressure drag induced by the detached flow in the rear of a car or a truck. It is motivated by the important energy savings that can be achieved by designing more aerodynamic vehicles [23, 157].

In most of these studies however, the description is limited to the average behaviour of the flow and typical fluctuations. Even though they are rare, extreme fluctuations of the drag can be expected to have a strong impact, both on the immersed body and on the structure of the flow. In bluff body aerodynamics, a key question is the description and quantification of the respective contribution of the coherent structures forming in the vicinity of the body to the overall drag [47, 142]. We believe the study of flow configurations leading to extreme values of the drag can provide useful information that will help deepen our

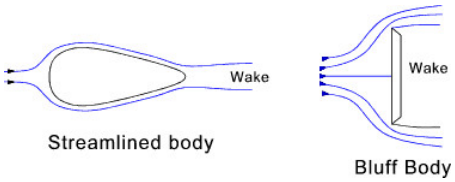


Figure 3.1: Typical examples of streamlined and bluff bodies. Blue lines represent the trajectory of fluid particles, referred to as *streamlines* (see note 3). In the case of the streamlined body, streamlines stay attached to the surface of the obstacle. A a result, the spatial extent of the wake is diminished, and a high pressure is recovered at the downstream point of the body. By contrast, the important cross flow section of bluff bodies entails a flow separation, as the trajectory of fluid particles detach from the surface of the body. A a result, the pressure near the downstream surface of the obstacle is considerably lower than the pressure near the upstream surface, resulting in a large drag.

understanding of the physical mechanisms underlying strong drag forces on bluff bodies.

In this chapter we propose a description of flow dynamics resulting in extreme fluctuations of the drag acting on a square cylinder immersed in a turbulent flow. We first investigate fluctuations of the instantaneous drag, denoted as f_d and defined as the total contribution of the pressure and viscous efforts over the surface $\mathcal{S}_b$ of the body:

$$f_d(t) = \int_{\mathcal{S}_b} (-p\mathbf{I} + \boldsymbol{\tau}) \cdot \mathbf{e_x} \mathrm{d}\mathcal{S}, \tag{3.1}$$

where $I_{ij} = \delta_{ij}$, $\boldsymbol{\tau}$ is the viscous stress tensor, $\mathbf{e_x}$ a unit vector aligned with the mean flow direction and $\mathrm{d}\mathcal{S}$ a surface element.

Additionally, we also provide a description of extremes of the time-averaged drag, denoted as F_T and defined as

$$F_T(t) = \frac{1}{T} \int_t^{t+T} f_d(\tau) \mathrm{d}\tau. \tag{3.2}$$

In the following the averaging window T will always be chosen significantly greater than the correlation time of the instantaneous drag τ_c, typically $T = 10\tau_c$.

Extreme fluctuations of the instantaneous drag f_d must be associated with atypical flow configurations, resulting in a very high, or low, value of the pressure difference between the forebody and the afterbody. In many applications, the reliability of immersed structures is guaranteed up to a fixed threshold for the drag, or related mechanical efforts. On the other hand, the characterisation of fluctuations of the averaged drag F_T is useful in cases where the structure of interest exhibits a typical response time, of the order of magnitude of T. In the following,

we focus on positive extreme fluctuations, *i.e.*, very large values of the drag force.

Note 1. The correlation time τ_c

In the following, we denote by τ_c the correlation time *of the instantaneous drag f_d. It refers to the typical time over which the drag de-correlates from its previous values, under the effect of large scale turbulent velocity fluctuations. In practice, we define this correlation time as the time over which the autocorrelation vanishes, i.e. $\mathbb{E}[X(t)X(t+\tau_c)] - E[X]^2 \approx 0$. The autocorrelation is computed over the whole control simulation by means of the Wiener-Khintchine theorem. In practice, we find for test flow (2) that $\tau_c \approx 4 \times U_0/L$ where U_0 is the mean flow velocity and L the cross-section of the obstacle. Please refer to chapter 2, section 2.2.2.2 for more details about the correlation time and its computation for both test flows.*

We sampled extreme fluctuations based on a very long numerical simulation of the test flows presented in chapter 2. Due to the two-dimensional nature of the flows and their relative simplicity, hundreds of thousands of correlation times could be simulated in roughly a weektime. Because the return period of typical fluctuations is close to the correlation time, the simulation of the flow over such very long durations allowed to sample very rare events with respect to typical events. In section 3.1, the sampling of extreme drag fluctuations from a long drag timeseries is described. Then, section 3.2 describes the study of instantaneous fluctuations. In section 3.2.1, it is shown that the major contribution to the extreme drag fluctuations is due to exceptional pressure drops in the vicinity of the base of the cylinder. By contrast, the pressure fluctuations near the forebody due to the upstream turbulence plays a lesser role. The flow dynamics leading to such extreme pressure drops—and therefore extreme drag fluctuations— are then described in section 3.2.2. In the present chapter, we concentrate on extremes for test flow (2). The flow fields for the four highest sampled fluctuations are shown to display very similar features, suggesting a common dynamical scenario for the formation of extreme drag configurations. A deeper analysis then reveals that extremes can be described in terms of two scenarii, in both of which a low pressure region is constrained in the vicinity of the base by the surrounding flow. A discussion of the temporal aspects of such dynamics is also provided.

Section 3.3 focuses on fluctuations of the averaged drag over $10\tau_c$. To begin with, section 3.3.1 illustrates that extreme values of the average drag result either from a few number of intense fluctuations of the instantaneous drag, or from a series of a greater number of fluctuations occurring in a sequence. However, it is not clear from the data if one scenario is dominating. Section 3.3.2 describes an analogy based on simple stochastic dynamics with a fast decay of the correlations. It is shown that for an algebraic and Gaussian PDF,

fluctuations of the average are dominated by a single fluctuation or a sequence of independent fluctuations, respectively. More importantly, the case of an exponential PDF is shown to be a marginal case for which both contributions are equally likely. The connection with the drag fluctuations then lies in the statistics of rare drag fluctuations observed in chapter 2, that are well described by an exponential PDF.

3.1 SAMPLING OF EXTREMES FROM A TIMESERIES

An extreme fluctuation of the drag, whether it is instantaneous or averaged, refers to an atypical excursion from values close to the average or typical fluctuations, to regions located above a given threshold. The higher the threshold, the rarer the event. In this section, we define precisely what we consider as rare events and how we sample them from a simulation of the dynamics over long durations.

Let $\{X(t)\}_{0\leq t\leq T_{tot}}$ be a timeseries describing the evolution of an observable X over time. In the context of this work, X denotes either the instantaneous drag, *e.g.* $X(t) \equiv f_d(t)$, defined by (3.1), or the drag averaged over a duration T, *e.g.* $X(t) \equiv F_T(t)$, defined by (3.2). Furthermore, let τ_c be the correlation time of X, see note 1 on page 49. We then set a threshold a and define extremes of X as the local maxima over regions for which $X \geq a$. More precisely, we denote by $\{(t_i^\star, X_i^\star)\}_{1\leq i\leq N}$ the set of the N extremes sampled in the timeseries, where $t_i^\star$ denotes the time at which the maximum is attained and $X_i^\star$ its value. Let τ_f be the typical timescale for the formation of an extreme fluctuation of X. Roughly speaking, this is the time it takes to depart from typical values and attain the peak value, and then fall back to the typical region. Therefore, we expect τ_f to be of the order of magnitude of the correlation time τ_c. Extreme fluctuations of X a then defined as follows:

$$\begin{cases} t_i^\star = \underset{t_i\leq t\leq t_i+\tau_f}{max} X(t) \\ X_i^\star = X(t_i^\star) \end{cases} \tag{3.3}$$

with t_i defined as

$$t_i = \begin{cases} \min(t \geq 0 \mid X(t) \geq a), & \text{if } i = 1 \\ \min(t \geq t_{i-1} + \tau_f \mid X(t) \geq a) & \text{otherwise} \end{cases} \tag{3.4}$$

Figure 3.2 illustrates the sampling of extremes in the case of the instantaneous drag, with $a = 3\sigma$ and $\tau_f = 2\tau_c$ with σ the standard deviation computed over the timeseries.

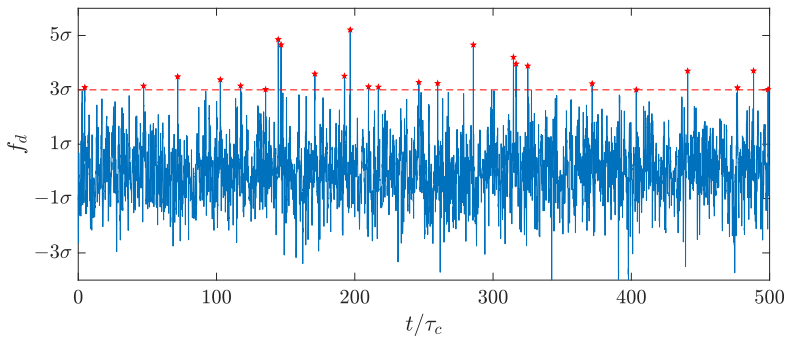


Figure 3.2: Sample timeseries for the instantaneous drag acting on the square cylinder in test flow (2). In this example the threshold is set to $a = 3\sigma$ and the corresponding sampled extreme fluctuations are marked by red stars. The sampling of these events is based on equations (3.3) and (3.4) with $\tau_f = 2\tau_c$.

As the length T_{tot} of the timeseries is fixed, increasing the threshold a will naturally lead to fewer and fewer events. Let $N_e(a)$ be the number of events resulting from the sampling of extremes with a threshold a. In practice, we would like to choose a so that the fluctuation is extremely rare. However, the timeseries must contain a statistically significant number of extreme fluctuations, say $N_e(a) = 100$ events. In order to choose a value for a, it is useful to recall the notion of *return time*, introduced in chapter 1. The return time $r(a)$ associated with an amplitude a is the typical timescale of occurrence of a fluctuation $X \geq a$. It is defined as $r(a) = \mathbb{E}[\tau(a,t)]$ where $\tau(a,t)$ is the *waiting time* for a fluctuation of amplitude a from time t: $\tau(a,t) = \min\{\tau \geq t | A(\tau) \geq a\} - t$. The return time is therefore the average time one must wait to observe fluctuations of amplitude above a. As a result, the number of events $N_e(a)$ sampled from a timeseries of duration T_{tot} with a threshold a is

$$N_e(a) \approx \frac{T_{tot}}{r(a)} \tag{3.5}$$

In order for the sampled fluctuations to be extreme, we must set a so that $r(a) \gg \tau_c$. In practice, test flow (2) has been simulated over $T_{tot} = 10^6 \tau_c$. In the following, we refer to this simulation as the *control run*. In order to sample roughly $N_e(a) = 100$ events, the threshold a must then be set so that $r(a) \approx 10^4 \tau_c$. Figure 3.3 illustrates the return time plot associated with fluctuations of the instantaneous drag for test flow (2). It displays the amplitude of the fluctuations as a function of the return time, and has been computed on the basis of a timeseries for f_d spanning $10^5 \tau_c$. The computation of return times from a timeseries is described in chapter 7.

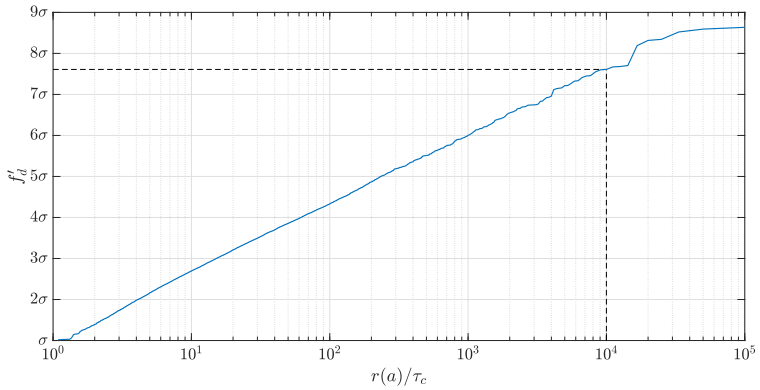


Figure 3.3: Return time plot for the instantaneous drag fluctuations based on a timeseries of duration $T_{tot} = 10^5\tau_c$ for test flow (2). The fluctuation is defined as $f'_d = (f_d - \bar{f}_d)/\sigma$ with $\bar{f}_d$ the average and σ the standard deviation computed over the timeseries. It illustrates that a return time $r(a) = 10^4\tau_c$ corresponds to fluctuations of order $a \approx 7.6\sigma$. The computation of such plot is described in chapter 7.

3.2 FLUCTUATIONS OF THE INSTANTANEOUS DRAG

Section 3.1 described the sampling of extreme fluctuations of an observable X from a timeseries $\{X(t)\}_{0\leq t\leq T_{tot}}$. In this section, we focus on the instantaneous drag: $X(t) \equiv f_d(t)$. test flow (2), described in 2, have been simulated over $T_{tot} = 10^6\tau_c$. Following figure 3.3, the threshold was set to $a = 7.6\sigma$ in order to sample roughly 100 events having a return time of *at least* $10^4\tau_c$. This resulted in a set of 104 events. These extreme events were then re-simulated in order to compute the corresponding velocity and pressure fields, as well as other relevant observables, as explained in note 2 on page 52. Figure 3.4 displays both the PDF of f_d computed on the basis of the timeseries and the amplitude of the sampled events. It illustrates that these fluctuations are indeed located in the far tail of the PDF.

Note 2. Re-simulation of the sampled extreme fluctuations

The state of the flow, e.g. the ensemble of Lattice Boltzmann populations at each computational node, as well as the velocity and pressure fields could not be stored at each LBM timestep along the simulation of the flow over $10^6\tau_c$, which represent roughly 2×10^9 timesteps. Indeed, doing so would have entailed tremendous memory storage, not to mention the potential computational overhead of constantly writing large amount of data on disk. As a consequence, during the control run, the state of the flow was only periodically written on disk. This allows to re-simulate each identified fluctuations from the nearest saved state so as to compute the velocity and pressure fields, and any other observable of interest over this particular event. This approach is actually crucial to the investigation of the physical mechanisms underlying extreme drag fluctuations, as the hydrodynamic quantities of interest were not known at the time the control run was carried out.

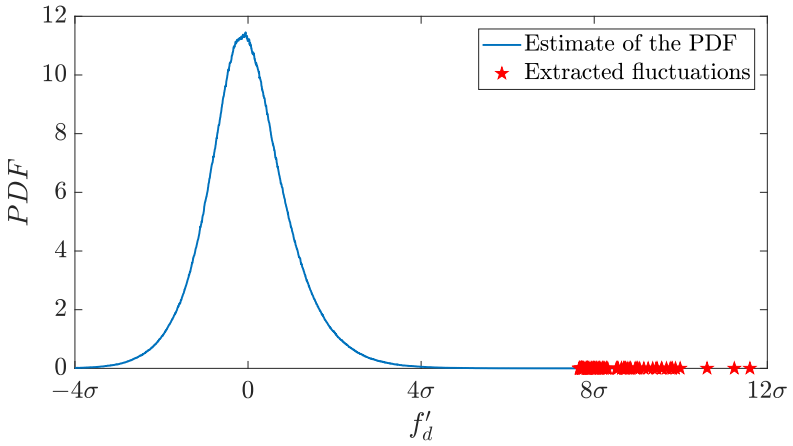


Figure 3.4: **Continuous blue line**: Estimate of the PDF for the fluctuations of the instantaneous drag f'_d in test flow (2), computed over a timeseries of duration $T_{tot} = 10^6\tau_c$. The fluctuation is defined as $f'_d = (f_d - \bar{f}_d)/\sigma$ with $\bar{f}_d$ and σ the average and standard deviation computed over the timeseries. **Red stars**: Sampled fluctuations above the threshold $a = 7.6\sigma$.

This section provides a description of extreme fluctuations of the instantaneous drag acting on the square cylinder in test flows 1, on the basis of the events sampled from the simulation of the flow over $10^6\tau_c$. First, section 3.2.1 discusses the respective role of the forebody pressure fluctuations due to the turbulent upstream flow and pressure fluctuations in the vicinity of the base of the cylinder. It shows that, in all sampled extreme fluctuations, drag fluctuations are dominated by the effect of exceptional pressure drops at the base. Then, section 3.2.2 describes the flow fields associated with several extreme events sampled from the timeseries. It leads to the classification of extremes in two categories, according to the physical scenario leading to the extreme pressure drop causing the drag fluctuation.

3.2.1 *Contribution of forebody and base pressure to the drag fluctuation*

For bluff bodies in high-*Re* flows, the total drag force is dominated by the pressure difference between the *forebody*, the solid surface in contact with the upstream flow, and the *base* in contact with the low pressure region resulting from the flow separation induced by the obstacle. For instance, in both test flow (1) and test flow (2), it was verified that the contribution of the viscous stress tensor to the integral in (3.1) accounts for less than 0.01%. Consequently, we neglect the contribution of viscous effects to the drag and solely retain the inertial effects, *i.e.* the difference between the forebody pressure and the base pressure.

In the case of a laminar upstream flow, drag fluctuations can be expected to result from flow structures forming near the base of the

obstacle. In [47], Fabiane *et al* proposed a decomposition of the flow past a cylinder, in order to quantify the relative contribution of two different areas:

- The external flow region, accounting for both the viscous stress and the pressure difference resulting from flow separation
- The back flow region, accounting for vorticity generated near the base

For moderate Reynolds numbers, for which the wake exhibits periodic vortex roll-up, it is shown that most of the fluctuations originate from the back flow region, and are strongly correlated with the vortex roll-up dynamics. In addition, drag fluctuations are reported to increase with the Reynolds number, as vortices are emitted closer and closer to the base.

The influence of upstream turbulence on the drag acting on a particle has been investigated in [82]. It is shown that an increasing upstream turbulent intensity leads to a shortening of the wake, and therefore to a decreasing of the averaged base pressure due to steeper averaged velocity gradients along the mean flow axis. In addition, it is reported that the averaged forebody pressure remains unchanged as the upstream turbulence intensity is increased, therefore leading to an overall increase of the drag. Turbulence intensity is also shown to increase the intensity of typical drag fluctuations, indicating that they are mainly connected with large scale fluctuations of the upstream flow. However, extreme fluctuations have not been addressed.

For a turbulent upstream flow, it is not clear whether extreme drag fluctuations are caused by strong pressure drops related to vorticity interactions occurring at the base of the obstacle, or exceptional forebody pressure resulting from the upstream turbulence. In the following we address this question based on the extreme events sampled from the control simulations of test flow (2).

For each of the sampled events, the drag fluctuation f'_d is split into the contribution of the forebody pressure fluctuation p'_f and base pressure fluctuation p'_{base}. Respectively denoting by $\overline{f_d}$, $\overline{p_f}$ and $\overline{p_{base}}$ the average drag, forebody pressure and base pressure, this decomposition writes

$$f'_d = f_d - \bar{f}_d = (p_f - \overline{p_f}) - (p_{base} - \overline{p_{base}}) = p'_f - p'_{base} \tag{3.6}$$

where p_f and p_{base} denote the instantaneous forebody and base pressure, computed by integration of the pressure over the upstream or downstream surface of the square, respectively. Average quantities are computed as time averages over the whole control timeseries. Figure 3.5 displays both the relative contribution of the forebody pressure fluctuation p'_f and base pressure fluctuation p'_{base} to the total

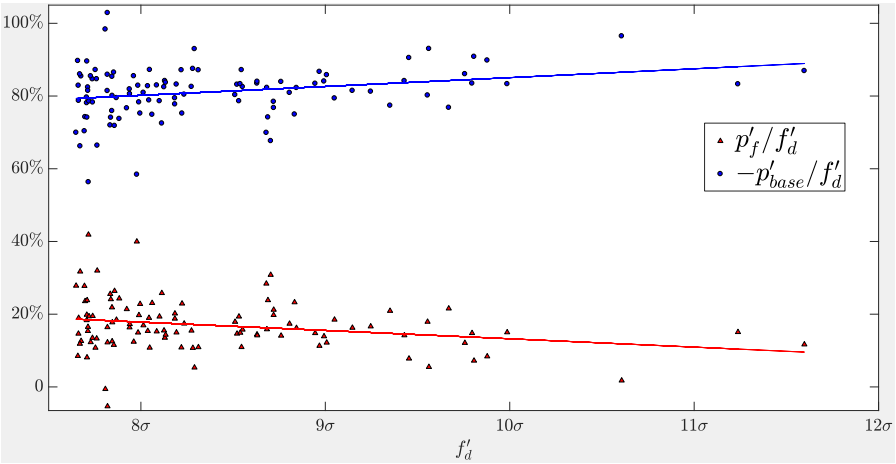
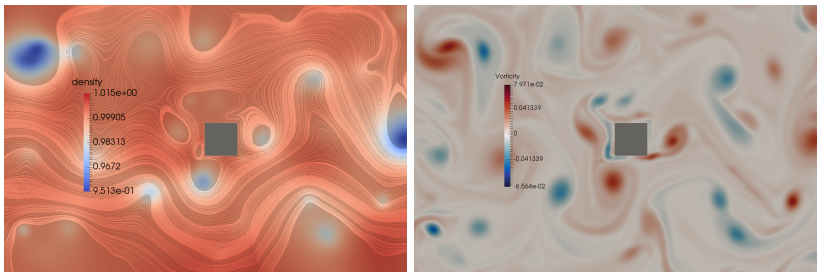


Figure 3.5: Contribution of the forebody and base pressure fluctuation to the overall drag fluctuation for the 104 events extracted from the control run for test flow (2). Both contributions are computed as p'_{base}/f'_d and p'_f/f'_d where the prime indicates fluctuations with respect to the average value computed over the timeseries. To each value of the drag corresponds a value for the base pressure fluctuation (blue dots) and for the forebody pressure fluctuation (red triangles). Note that the sum of these two values is 1 and that a negative contribution from the forebody pressure indicates events for which an extreme fluctuation occurs even though the forebody pressure is lower than its average. This type of event has been been only observed once among the 104 events. This figure illustrates that drag fluctuations are mostly caused by fluctuations of the base pressure. In contrast, the forebody pressure does not deviate as far from its average value. In addition, both thick continuous lines result from a least-squares fit of the data to a second order polynomial. They highlight that the segregation between the two contributions seems to increase as the amplitude of the fluctuation increases.

fluctuation f'_d, for the 104 extreme events extracted from the control simulation of test flow (2). One can see that, in each and every case, the major contribution to the drag fluctuation originates from the base pressure fluctuation. Furthermore, figure 3.5 shows that, as the amplitude of the fluctuation increases, the contribution from the base pressure fluctuation also increases, with respect to the contribution from the forebody pressure. This indicates that, similarly to the case of a laminar flow impacting an obstacle, drag reduction can efficiently be achieved by mitigating pressure fluctuations at the base of the obstacle.

3.2.2 *Qualitative description of flow configurations leading to extreme drag fluctuations*

In section 3.2.1, it was observed that, despite the turbulent nature of the upstream flow, pressure fluctuations in the vicinity of the forebody



(a) Density field and streamlines (b) Vorticity field

Figure 3.6: Flow fields corresponding to a typical value of the drag acting on the square cylinder: $f_d \approx \bar{f}_d$ where $\bar{f}_d$ denotes the average drag computed over the control timeseries for test flow (2).

play little role in the extreme drag fluctuations sampled from the control simulation. In contrast, the majority of the fluctuations were shown to originate from exceptional pressure drops near the base of the cylinder.

In this section, we investigate the flow dynamics leading to such pressure drops. The starting point of this study is the visualisation of the flow fields corresponding to a few extremes. Remarkably, we illustrate that they all display very similar configurations. From these observations, we draw up hypothesis regarding the dynamical scenario leading to such extremely low values of the base pressure and confront them with the whole ensemble of sampled extremes. Eventually, we will propose a classification of extremes into two categories, depending on the corresponding flow dynamics.

3.2.2.1 *Illustration of a typical event*

Before we describe the case of extreme fluctuations, we begin by characterising a flow configuration corresponding to a typical event. Figure 3.6 displays the vorticity field and density field for a flow configuration resulting in a drag close to the average value. Recall that the density field is related to the pressure field through the state equation $p(\mathbf{x}) = c_s^2 \rho(\mathbf{x})$ where p denotes the pressure, ρ the density and c_s^2 the speed of sound in the fluid. See chapter 2, section 2.1 for more details about this relation. In addition, figure 3.6a displays the velocity streamlines. These lines are tangential to the velocity vector field in each point of the domain, and gives a graphical representation of advection by the flow at a given time. Streamlines are further discussed in note 3. As can be seen in figure 3.6b, the vicinity of the base is surrounded by irrotational flow. As a consequence, the pressure difference between the forebody and the base must solely result from the contribution of the flow separation induced by the obstacle. Recall that viscous effects are neglected with respect to pressure effects. As a matter of fact, vorticity produced by shear alongside the boundaries of the cylinder is advected downstream by the mean flow. This can be seen in

figure 3.6. Positive vorticity—coloured in red— is produced along the bottom boundary layer, resulting in a positive vortex. However, this vortex forms far enough from the base so that is does not perturb the irrotational flow surrounding the base of the obstacle. Therefore, it does not lead to any pressure drop and, consequently, to any increase of the drag.

Note 3. Streamlines

Streamlines are a useful tool when it comes to visualising a vector field, for instance the velocity field describing the flow of a fluid at a given moment in time. A velocity streamline is a curve that is tangential to the velocity field at each point in space. For a stationary flow, *streamlines correspond to the trajectory of fluid particles. Even though flows concerned in this work are unstationary, visualisation of streamlines is useful to provide a graphical representation of advection by the flow at a given moment in time. Let* $\mathrm{d}\mathbf{r}(\mathbf{x}) = (\mathrm{d}x(\mathbf{x}), \mathrm{d}y(\mathbf{x}))$ *be an infinitesimally small displacement along the streamline at position* $\mathbf{x}$*. The line is defined by*

$$\mathbf{u} \times \mathrm{d}\mathbf{r} = \mathbf{0} \Leftrightarrow \frac{\mathrm{d}x}{\mathrm{d}y} = \frac{u_x}{u_y} \tag{3.7}$$

In practice, a streamline is computed by integrating equation (3.7) *from an initial* seed point $\mathbf{r}_0$*. In this work, we used Paraview [1] for flow visualisation. Streamlines are computed by specifying an ensemble of seed points from which* (3.7) *is integrated both forward and backward in time.*

3.2.2.2 *Extreme events*

In the following we illustrate that extreme fluctuations actually result from vorticity being trapped in the vicinity of the base of the cylinder, in contrast with typical fluctuations where it is advected downstream by the mean flow. We denote by $t^\star$ the time corresponding to the peak drag for a given extreme. Following note 2, the events sampled from the control simulations for test flow (2) have been re-simulated from $t^\star - 2\tau_c$ to $t^\star + 2\tau_c$. As an illustration, figure 3.7 displays the evolution of the drag over time for the four events having the highest amplitude in the control simulation for test flow (2). Figures 3.8 and 3.9 illustrate the corresponding configuration of the flow for the same fluctuations. More precisely, figure 3.8 displays the vorticity field in the vicinity of the square cylinder, at $t = t^\star$. One can see that, in each case, vorticity is produced very close to the base of the obstacle. In addition, its amplitude is significantly higher than the vorticity fluctuations displayed in figure 3.6 for a typical event. This vorticity originates from the interaction of the flow with the obstacle. In the examples of figure 3.8, vorticity is produced through viscous interactions at the bottom boundary of the obstacle. Note that the four greatest fluctuations in the control simulation for test flow (2) result from the production of positive vorticity along the bottom boundary. We stress here that this is a mere coincidence as the geometry, as well as its discretisation, is symmetric with respect to the pipe axis.

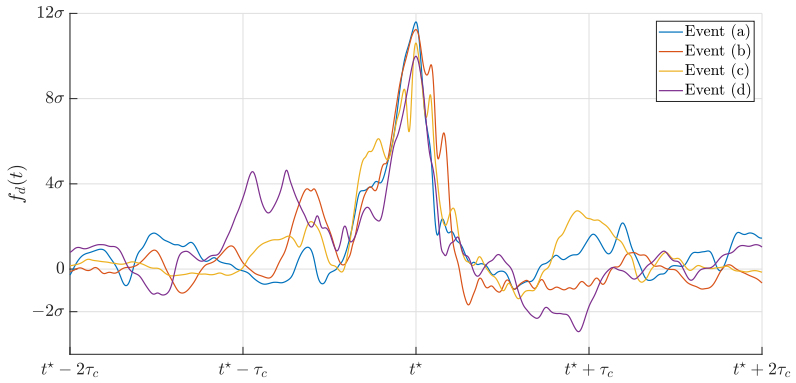
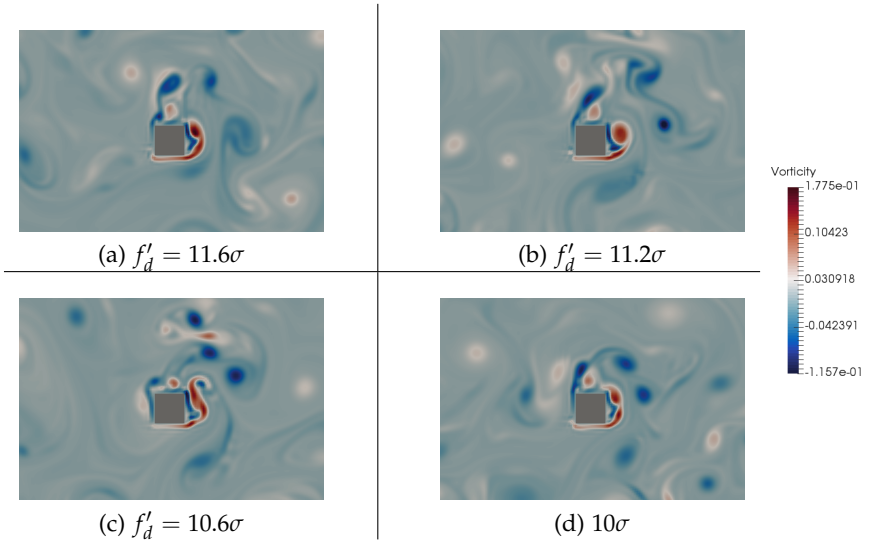


Figure 3.7: Drag timeseries for the four extreme fluctuations having the highest amplitude in the control simulation for test flow (2). Timeseries are centred around $t = t^\star$, denoting the instant for which the maximum drag is attained. The duration τ_c denotes the correlation time of the drag, see note 1 on page 49. Extreme fluctuations, *i.e.* excursions to atypical values followed by a relaxation to typical values, happens over roughly a correlation time. Vorticity and pressure fields at $t = t^\star$ for the four events can be found in figures 3.8 and 3.9. Flow dynamics for event (a) are illustrated in figure 3.10.

Figure 3.9 displays the density field at $t = t^\star$ as well as the streamlines for the velocity field, see note 3. Figure 3.9 highlights that the pressure drop responsible for the drag fluctuation results from the formation of a strong vortex localised very close to the obstacle. In addition, a vortex pair is localised further downstream, consisting of two vortices with a sign opposite to the vortex in the vicinity of the base, as shown in figure 3.8. Visualisation of streamlines in figure 3.9 suggests that this downstream vortical region shields the strong vorticity located at the base from advection by the mean flow.

Remarkably, the four extreme events featured in figures 3.8 and 3.9 display very similar flow configurations in which strong vorticity is produced through viscous shear along a boundary of the obstacle tangential the to flow direction. In contrast with the typical event depicted in figure 3.6, this vorticity is not advected by the mean flow and develops very close to the base, leading to a strong pressure drop.

In order to understand better the development of such a configuration, we now describe an example of flow dynamics leading to an extreme fluctuation. This example is based on the event featured in figures 3.8 and 3.9. Figure 3.10 illustrates the evolution of the vorticity field from one half of a correlation time before the peak drag. It shows that from roughly $t^\star - \tau_c/2$ to $t^\star - \tau_c/3$, the top boundary layer is perturbed by a vortex advected by the flow above the obstacle. This can be seen in frames 3.10a trough 3.10g. The interaction of this vortex with the top boundary layer results in a boundary layer separation



(a) $f'_d = 11.6\sigma$ (b) $f'_d = 11.2\sigma$

(c) $f'_d = 10.6\sigma$ (d) 10σ

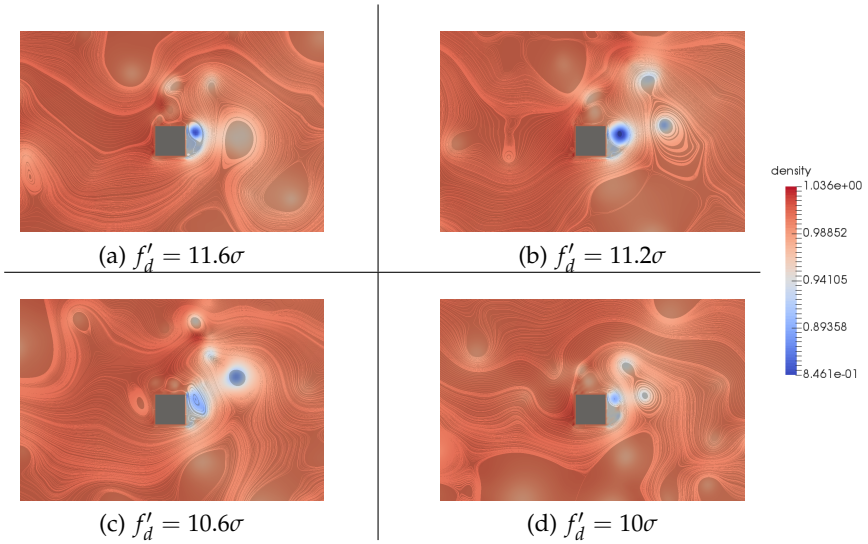
Figure 3.8: Vorticity field at $t = t^\star$ for the four highest drag fluctuations in the control simulation for test flow (2). The corresponding drag timeseries can be found in figure 3.7. The high value for the drag results from the formation of a strong negative (red) vortex inducing a pressure drop at the base of the obstacle. The formation of such a structure is aided by important vorticity production at the bottom boundary of the obstacle coupled with the influence of positive vorticity in the vicinity of the base.

that eventually leads to the formation of a large—of the scale of the obstacle—negative vorticity region further downstream of the cylinder, visible in frame 3.10k. This vortex does not have a direct impact on the pressure at the base of the obstacle. However, it interacts with vorticity produced along the bottom boundary of the cylinder, as illustrated by frames 3.10k to 3.10p. Indeed, by contrast with the typical event depicted in figure 3.6, vorticity produced along the bottom boundary is not advected downstream. Instead, it is transported in the close vicinity of the base by the downstream negative vorticity region resulting from the separation of the top boundary layer.

3.2.2.3 *Type 1 events*

The description of the flow for the four highest fluctuations in the control simulation leads to the following hypothesis: extreme drag fluctuations are caused by intense vorticity localised very close to the base of the obstacle, generated by a strong shear layer on either the top or bottom boundary, as well as the action of a large opposite vorticity region originating from a boundary layer separation on the opposite boundary.

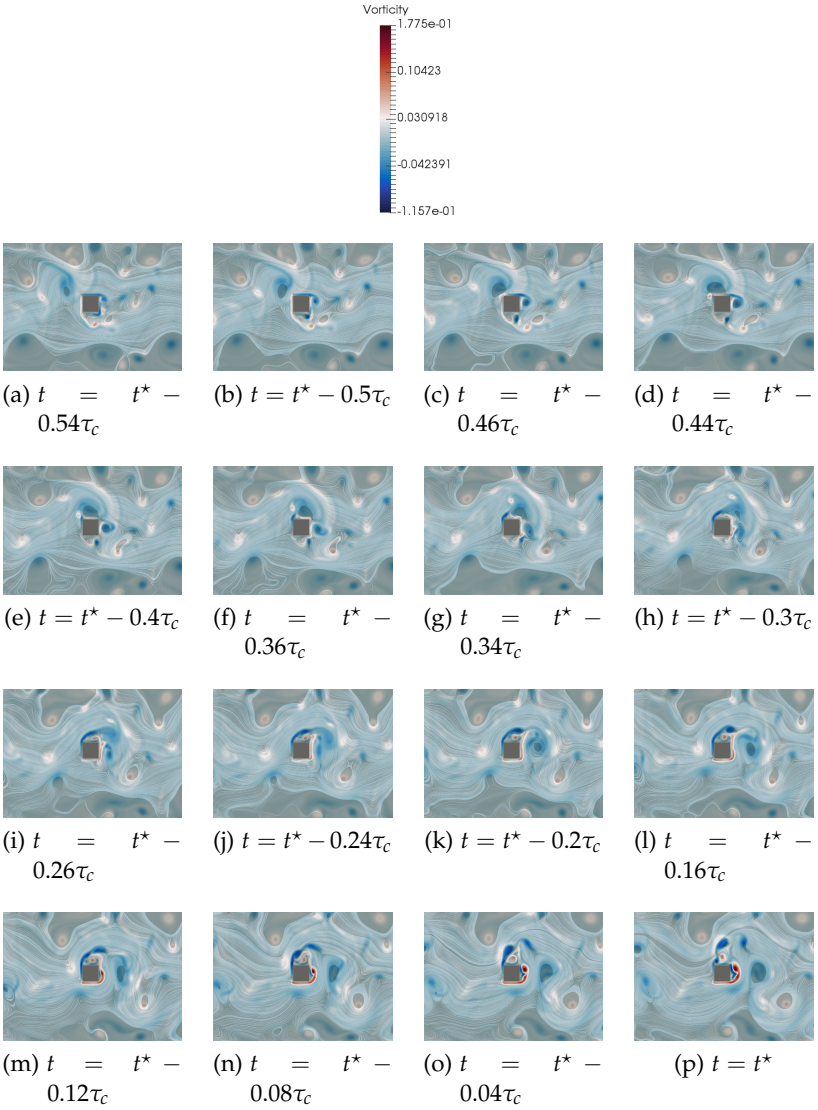
We now test the validity of this scenario on the basis of the 104 events extracted from the control simulation of test flow (2). To do so, we highlight the correlation between an increase of the shear along either the top or the bottom boundary of the cylinder and an increase



(a) $f'_d = 11.6\sigma$ (b) $f'_d = 11.2\sigma$

(c) $f'_d = 10.6\sigma$ (d) $f'_d = 10\sigma$

Figure 3.9: Density field at $t = t^\star$ for the four highest drag fluctuations in the control simulation for test flow (2). Recall that the density ρ is proportional to the pressure p, following the ideal gas state equation $p = c_s^2 \rho$. See chapter 2 and appendix B. Additionally, velocity streamlines are depicted, representing advection by the flow at $t = t^\star$, see note 3. Blue areas indicate areas of lower pressure while red regions indicate regions of higher pressure. The formation of a vortex very close to the base boundary leads to a strong pressure drop. The downstream vortices originating from the top boundary layer separation are clearly visible and constrain the formation of the base vortex to a region very close to the base boundary.



(a) $t = t^\star - 0.54\tau_c$ (b) $t = t^\star - 0.5\tau_c$ (c) $t = t^\star - 0.46\tau_c$ (d) $t = t^\star - 0.44\tau_c$

(e) $t = t^\star - 0.4\tau_c$ (f) $t = t^\star - 0.36\tau_c$ (g) $t = t^\star - 0.34\tau_c$ (h) $t = t^\star - 0.3\tau_c$

(i) $t = t^\star - 0.26\tau_c$ (j) $t = t^\star - 0.24\tau_c$ (k) $t = t^\star - 0.2\tau_c$ (l) $t = t^\star - 0.16\tau_c$

(m) $t = t^\star - 0.12\tau_c$ (n) $t = t^\star - 0.08\tau_c$ (o) $t = t^\star - 0.04\tau_c$ (p) $t = t^\star$

Figure 3.10: Flow dynamics corresponding to a particular type 1 event, which drag timeseries and flow fields at $t = t^\star$ are displayed in figures 3.7, 3.8 and 3.9, respectively. In each frame the vorticity field is displayed, as well as the velocity streamlines, representing advection by the flow at each instant. See note 3 for a discussion of streamlines. The maximum of the drag is attained for $t = t^\star$ and corresponds to frame 3.10p. The sequence starts with a shear boundary layer forming over the top boundary of the obstacle (frames 3.10a to 3.10d). This boundary layer separates (frames 3.10e to 3.10j) and results in the formation of a large positive eddy in the vicinity of the base of the obstacle (frames 3.10k and 3.10l). In the meantime, frames 3.10j to 3.10m depict the formation of an attached strong shear layer at the bottom boundary. Eventually, frames 3.10n to 3.10p illustrate that this results in vorticity production very close to the base, aided by the large positive eddy originating from the top boundary layer separation.

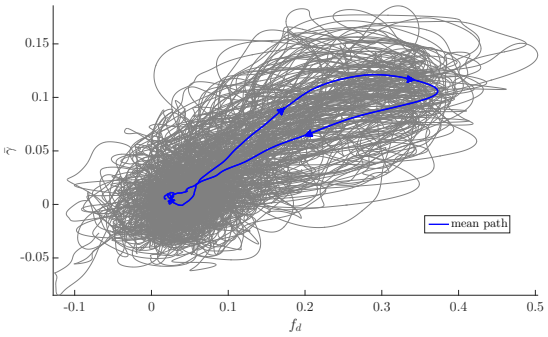


Figure 3.11: Averaged shear on either the top or bottom boundary of the obstacle as a function of the drag. Grey lines denote the trajectories in the (f_d, σ) for the 86 type 1 events. The thick blue line represent the mean path. It was verified that every event results from a clockwise excursion away from the region of typical events, indicating that drag fluctuations are preceded by an increase of the shear on either the top or bottom boundary.

of the drag. More precisely, we consider the averaged shear $\overline{\gamma}$ along the top or the bottom boundary:

$$\overline{\gamma} = \frac{1}{L} \int_{\mathcal{S}_b} \frac{\partial u_x(\mathbf{x})}{\partial y} \mathrm{d}\mathbf{x} \tag{3.8}$$

where L denotes the diameter of the cylinder, u_x the longitudinal component of the velocity field and $\mathcal{S}_b$ the surface of either the top or the bottom boundary, depending on the event. For each extreme fluctuation in the control run, $\overline{\gamma}$ is plotted as a function of the instantaneous drag f_d, for $t^\star - 2\tau_c \leq t \leq t^\star + 2\tau_c$. Each extreme event in the control simulation can therefore be associated to a path in the space $(f_d, \bar{\gamma})$. The resulting plots are displayed in figure 3.11. For $t^\star - 2\tau_c \leq t \leq t^\star - \tau_c$ and $t^\star + \tau_c \leq t \leq t^\star + 2\tau_c$, paths concentrate in the region describing typical values for both $\overline{\gamma}$ and f_d. As illustrated in figure 3.7, the drag abruptly varies for $t^\star - \tau_c \leq t \leq t^\star + \tau_c$. Correspondingly, paths in the $(f_d, \bar{\gamma})$ space display excursions to atypical values for both $\bar{\gamma}$ and f_d. Interestingly, these excursions always go clockwise, that is, $\overline{\gamma}$ attains its maximum value before f_d does. This confirms that the increase of $\overline{\gamma}$ acts as a precursor for extreme drag fluctuations.

As a matter of fact, this correlation between $\bar{\gamma}$ and f_d was observed for roughly 80% of the extreme events sampled from the control simulation. Figure 3.11, as well as visualisation of the corresponding flow indicate that these events can all be described by dynamics very close to the ones described in figure 3.10. In the following we refer to this events as *type 1 events*.

3.2.2.4 *Type 2 events*

The remaining 20% of the extracted extreme fluctuations must be ruled by different dynamics. Visualisation of the corresponding flow field suggests that these can actually be described by a common alternative scenario, and in the following we will refer to these events as *type 2 events*.

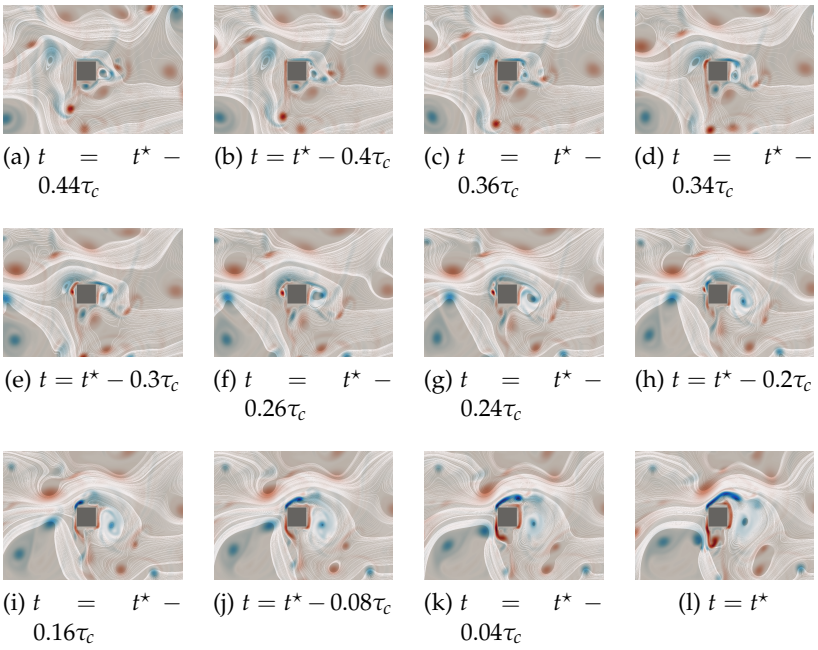
Figure 3.12 illustrates the dynamics of one of the type 2 events sampled from the control simulation. It displays the vorticity field and velocity streamlines at several instants from $t \approx t^\star - \tau_c/2$ until $t = t^\star$, where $t^\star$ denotes the time at which the peak fluctuation is attained. In addition, figure 3.12m illustrates the corresponding drag timeseries, for $t^\star - 2\tau_c \leq t \leq t^\star + 2\tau_c$. In this scenario, vorticity is still produced by shear on the top or bottom boundary, but at a much earlier stage. Figure 3.12a shows that a strong shear layer forms over the top boundary for $t \approx t - \tau_c/2$. It results in a large recirculation bubble in the vicinity of the base, that forms over a timescale of roughly $\tau_c/2$. Eventually, figure 3.12l shows the vorticity field and streamlines at $t = t^\star$. The recirculation bubble is detached from the base and forms a large area of negative vorticity. This results in a strong upward transverse velocity, as illustrated by the concentration of the streamlines along the base of the obstacle, leading to a local pressure drop. The corresponding velocity field is shown in figure 3.13b.

In order to test this scenario, we proceed in a similar way as for type 1 events, characterised by figure 3.11. For the 20% of sampled fluctuations that do not correspond to type 1 events, we wish to assess if the base transverse velocity fluctuations can be considered a signature of the drag fluctuations. More precisely, for each event we measure the transverse velocity u_y over a region close to the base of the obstacle and compute the average:

$$\overline{u_y}^{base} = \frac{1}{\mathcal{A}} \int_{\mathcal{S}} u_y(\mathbf{x}) \mathrm{d}\mathbf{x} \tag{3.9}$$

where $\mathcal{S}$ denotes the integration region, and $\mathcal{A}$ the corresponding area. The integration region $\mathcal{S}$ is pictured in figure 3.13b. The average base transverse velocity $\overline{u_y}^{base}$ is then plotted as a function of the drag for each event, for $t^\star - 2\tau_c \leq t \leq t^\star + 2\tau_c$. Similarly to figure 3.11, figure 3.13a displays the corresponding paths in the two dimensional space $(f_d, \overline{u_y}^{base})$. For $t \in [t^\star - 2\tau_c; t^\star - \tau_c/2] U [t^\star + \tau_c/2; t^\star + 2\tau_c]$, paths concentrate in the region describing typical values for $\overline{u_y}^{base}$ and f_d. However, we verified that for $t^\star - \tau_c/2 \leq t \leq t^\star + \tau_c/2$, each path consists in an excursion far away from this region. Furthermore, these excursions go clockwise, suggesting that an increase of $\overline{u_y}^{base}$ acts as a precursor for a increase of the drag f_d.

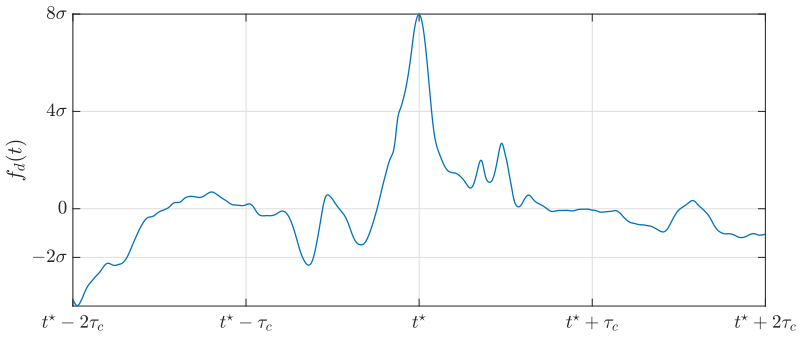
Even though all the 104 sampled events can be considered rare—they all have a return time of *at least* 10^4 correlation times—they span a



(a) $t = t^\star - 0.44\tau_c$ (b) $t = t^\star - 0.4\tau_c$ (c) $t = t^\star - 0.36\tau_c$ (d) $t = t^\star - 0.34\tau_c$

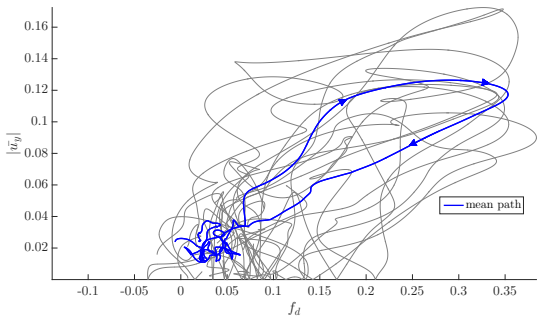
(e) $t = t^\star - 0.3\tau_c$ (f) $t = t^\star - 0.26\tau_c$ (g) $t = t^\star - 0.24\tau_c$ (h) $t = t^\star - 0.2\tau_c$

(i) $t = t^\star - 0.16\tau_c$ (j) $t = t^\star - 0.08\tau_c$ (k) $t = t^\star - 0.04\tau_c$ (l) $t = t^\star$

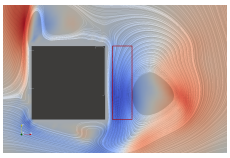


(m) Evolution of the drag f_d over time, centred on the extreme fluctuation at $t = t^\star$

Figure 3.12: Formation of a type 2 drag fluctuation over time. Streamlines are coloured according to the transverse velocity u_y. Red areas indicate regions of upward transverse velocity while blue areas indicate regions of downward transverse velocity. The maximum of the drag is attained for $t = t^\star$ and corresponds to frame 3.10l. The sequence illustrates the formation of a large eddy in the vicinity of the base of the obstacle, resulting from vorticity produced by a shear layer forming at the top boundary (frames 3.10a to 3.10e). The eddy is then detached from the base as flow separation occurs at the top boundary. As illustrated in frame 3.10l, the detached eddy lead to a strong upward velocity at the base of the obstacle, leading to a low pressure.



(a) Base transverse velocity as a function of the drag.



(b) Spatial integration of u_y. Blue indicates downward velocity. Red indicates upward velocity.

Figure 3.13: Base averaged transverse velocity $\overline{u_y}^{base}$ as a function of the drag, for the 18 type 2 events. The velocity $\overline{u_y}^{base}$ refers to the transverse velocity magnitude u_y averaged over a small region of space in the vicinity of the base of the obstacle. This region is depicted by the red rectangle in figure 3.13b. Grey lines in figure 3.13a represent the trajectories for each type 2 events. The thick blue line represents the mean path. In the space $(f_d, \overline{u_y}^{base})$, the 18 type 2 events result from a clockwise excursion away from the region describing typical values. This indicates that drag fluctuations are preceded by an increase of the transverse velocity magnitude.

wide range of fluctuation amplitudes. Having segregated the ensemble of events into two different scenarii, we would like to assess if one particular scenario leads to higher fluctuations than the other one. Figure 3.14 shows the intensity of the fluctuations corresponding to each of the 104 extracted events. Events are sorted according to their intensity. It shows that, among the ensemble of sampled events, higher fluctuations are only achieved through type 1 events.

3.2.3 *Conclusion*

From a long control simulation of the flow, roughly one hundred extreme fluctuations have been extracted. They all have a return time greater than 10^4 correlation times of the instantaneous drag. The analysis of the respective contribution of the forebody and base pressure to the overall drag fluctuation leads to the conclusion that, even with a turbulent incoming flow, fluctuations of the base pressure play a major role in the occurrence of unusually large drag events. Visualisation of the flow dynamics for the sampled events resulted into the ensemble being split into two different types of events. Type 1 events represent 80% of the sampled events and correspond to the formation of a small, intense vortex very close to the base of the obstacle. This vortex results from vorticity production by a strong shear layer along either the top or bottom boundary of the obstacle, as well as the influence of a large downstream vortical region resulting from a boundary layer separation over the boundary which is opposite to the one of the previously

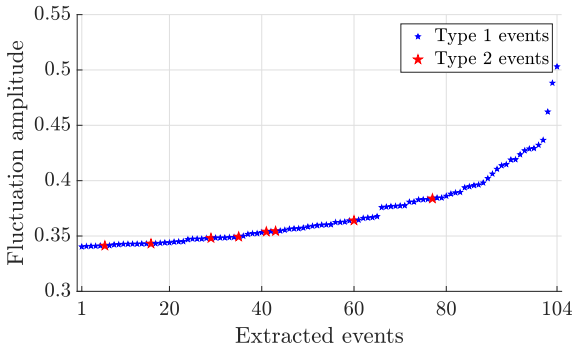


Figure 3.14: Classification of the 104 events sampled from the control simulation of test flow (2), according to the corresponding fluctuation amplitude. The highest fluctuations in the set of sampled events result from type 1 events.

mentioned shear layer. This second vortex forces the smaller vortex to form very close to the base. One could wonder about the persistence of such an event. Can the base vortex be blocked for duration of the correlation time ? Actually, it is never observed. Indeed, the downstream vortex is rapidly advected by the mean flow, therefore freeing the base vortex which is, in turn, advected downstream. Such fluctuations therefore happens on a relatively short timescale with respect to the correlation time, as illustrated by figure 3.7. Instead, what makes this type of event unusual is that vorticity generated by viscous shear on the top or bottom surface is not directly advected downstream, but concentrates near the base. Type 2 events represent the remaining 20% of the sampled events. In this scenario, the pressure drop results from an intense transverse velocity in the vicinity of the base, generated by a detached vortex resulting from flow separation at either the top or bottom boundary of the obstacle.

3.3 FLUCTUATIONS OF THE AVERAGED DRAG

In section 3.2, we presented a qualitative study of the mechanics underlying extreme fluctuations of the instantaneous drag. We showed that they correspond to excursions to very high values of the drag, with the overall fluctuation—the excursion followed by the relaxation—lasting roughly one correlation time. In this section we turn to extreme fluctuations of the *averaged* drag over a macroscopic duration, that is a duration several times bigger than the typical timescale of the large scales of the upstream turbulence. Considering the interaction of turbulence with a structure, atypical values of the averaged drag are expected to have a high impact for systems having a large response time with respect to the timescale of the surrounding turbulence. Examples of responses are the deformation of structures such as wind

turbine blades, or oscillations of tall building under fluctuating wind loads.

Following a strategy similar to section 3.2, we sample extreme averaged drag fluctuations from the control simulation for test flow (2), spanning $T_{tot} = 10^6\tau_c$. Let F_T be the drag averaged over T. From the control timeseries for the instantaneous drag f_d, we deduce the control timeseries $\{F_T\}_{T\leq t\leq T_{tot}}$ by integration over time:

$$F_T(t) = \frac{1}{T}\int_{t-T}^{t} f_d(\tau)\mathrm{d}\tau, \quad \text{for } T \leq t \leq T_{tot} \tag{3.10}$$

In the following, the time-average is performed over a duration $T = 10\tau_c$. Note that τ_c is of the order of the typical timescale of the large scale velocity and pressure fluctuations of the upstream flow, see note 1. Therefore, $T = 10\tau_c$ can be considered a macroscopic duration with respect to the flow past the obstacle. As mentioned in section 3.1—and observed in figure 3.7— extreme fluctuations of the instantaneous drag f_d are expected to develop over roughly a correlation time. Let τ_c^T be the correlation time for the averaged drag F_T. Because of the correlation induced by the integration, we expect $\tau_c^T \approx T$. With the length of the control timeseries fixed, larger integration times T therefore leads to the sampling of fewer events. In a way of conclusion, the choice of $T = 10\tau_c$ results from the balance between a time that is long enough to be considered larger than the timescale of large-scale turbulent fluctuations and small enough so that enough fluctuations can be sampled from the control timeseries. In the following, F_T denotes the averaged drag over $T = 10\tau_c$, unless specified otherwise.

Furthermore, we denote by σ_T the standard deviation computed over the control timeseries for F_T. Note that the control timeseries for F_T has a total duration $T_{tot} - T \approx T_{tot} \approx 10^6\tau_c \approx 10^5\tau_c^T$. Similarly to section 3.2, the choice of the threshold a for the sampling of extreme fluctuations is based on the return time plot for the fluctuations of F_T, computed over the control timeseries (not shown). This leads to $a = 3.7\sigma_T$, corresponding to a return time of $10^3\tau_c^T$ in order to sample roughly 100 events from the control timeseries, resulting in 85 independent extreme fluctuations for F_T.

The main question we address in this section concerns the structure of the fluctuations of the averaged drag. Do extreme values of the averaged drag over a macroscopic duration result from a series of statistically independent low-amplitude fluctuations ? Or from a few number of very large fluctuations ? In section 3.3.1 we describe several extreme fluctuations of F_T sampled from the control timeseries. More precisely, the instantaneous drag timeseries corresponding to these extreme values of F_T are displayed. Their analysis suggests that both very large fluctuations and unusual series of independent positive

typical fluctuations equally participate in the extreme values of the averaged drag. In oreder to better understand this property, we propose an analogy with simple stochastic systems with a fast decay of the correlation time in section 3.3.2. We illustrate that this property may result from the exponential tail statistics of the instantaneous drag, observed in chapter 2

3.3.1 *Examples of extreme fluctuations of the averaged drag*

Examples of timeseries of the instantaneous drag f_d corresponding to extreme fluctuations of the average drag, with $T = 10\tau_c$, are given in figure 3.15. In each case, several timeseries corresponding to typical events are displayed for comparison. Figure 3.15 illustrates that the occurrence of extremes for the average drag cannot easily be reduced to a few numbers of well-defined scenarii. Indeed, the featured timeseries display very different structures, even though their respective average all lead to fluctuations between 4.4σ and 5.1σ. For instance, in figure 3.15c the exceptionally large value of the average results from two very intense fluctuations, as well as a large number of positive typical fluctuations of order σ. In figure 3.15b, the extreme value for the overall average results from the succession of approximately 7 independent fluctuations around 3σ, while negative fluctuations never go beyond $-\sigma$. In general, it can be seen in figure 3.15 that extreme fluctuations of the averaged drag over $10\tau_c$ result from timeseries in which fluctuations are almost always positive. Furthermore, both very large fluctuations, $f_d \geq 5\sigma$, as well as more typical values, $\sigma \leq f_d \leq 2\sigma$, seem to play a role in the extreme values of the average. On the one hand, very large fluctuations are much less likely to occur, but have a strong impact of the average. On the other hand, typical fluctuations have a lesser impact, but they are more likely. Therefore, they can occur a greater number of times in the timeseries.

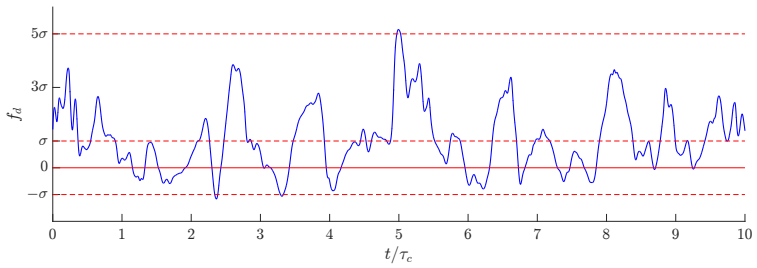
From the qualitative study of the 85 sampled events, we define two contributions to extreme values of the averaged drag:

1. The contribution of very large fluctuations, typically occurring once or twice in the timeseries.

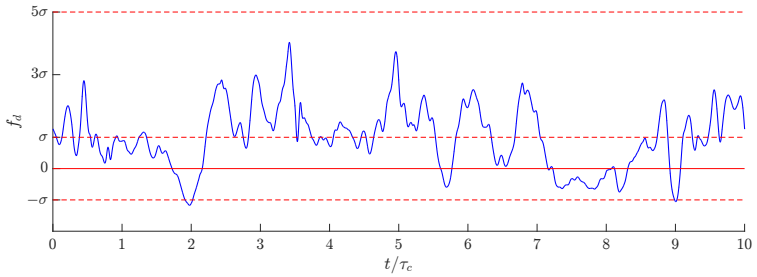
2. The contribution of series of typical positive independent fluctuations.

However, on the basis of the 85 events sampled from the control run, it is not clear if contribution (1) or (2) dominates.

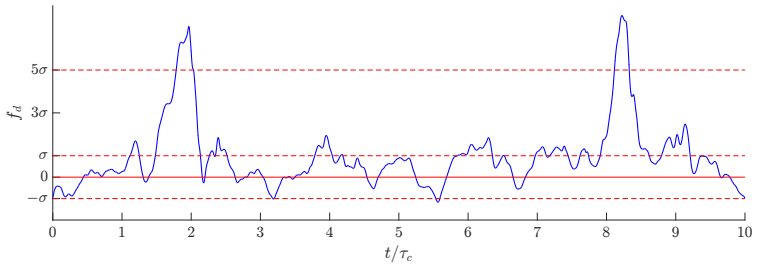
In the following, we consider a drag timeseries over several correlation times τ_c as a sequence of several independent fluctuations over a timescale τ_c. We motivate this approach by observing that the instantaneous drag f_d has rapidly decaying temporal correlations. In the next section, we describe the analysis of extreme fluctuations for



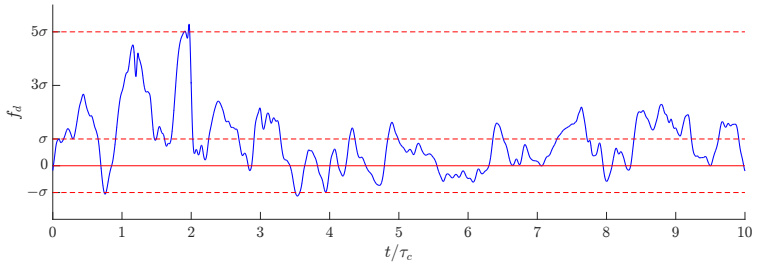
(a) $F_T = 5.1\sigma_T$



(b) $F_T = 4.7\sigma_T$



(c) $F_T = 4.5\sigma_T$



(d) $F_T = 4.4\sigma_T$

Figure 3.15: Instantaneous timeseries corresponding to the four highest fluctuations of the averaged drag on the control run (thick blue line). The grey lines in the background represents timeseries for which the overall average is typical.

one-dimensional stochastic processes with a fast decay of temporal correlations.

3.3.2 *Average extremes for some simple random processes*

Recall from chapter 2 that rare fluctuations of the instantaneous drag f_d are well described by an exponential PDF. That is:

$$\mathcal{P}(f \leq f_d \leq f + \epsilon) \underset{f \to \infty}{\sim} e^{-\alpha f} \mathrm{d}\epsilon$$

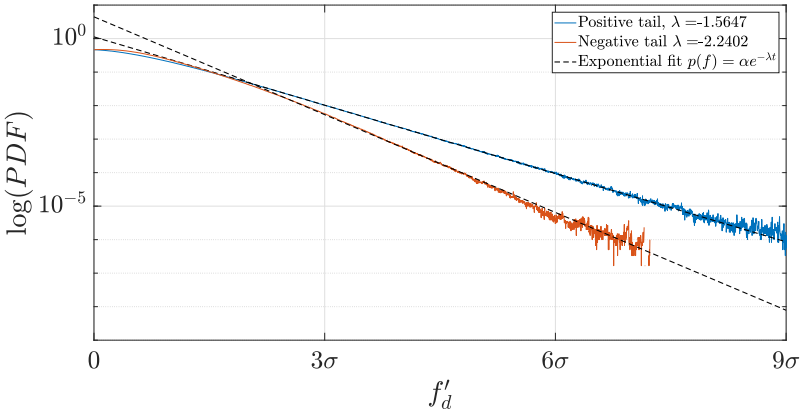
This is illustrated in figure 3.16a on page 71. In addition, figure 3.16b illustrates that the dynamical process defined by the time evolution of the drag f_d over time has exponentially decaying temporal correlations. This property is illustrated in figure 3.17 in which a process having short-range, exponential, temporal correlations is compared to a process with long-range, algebraic correlations.

In this section, we describe some statistical properties of extremes for one dimensional stochastic dynamics, with different stationary PDFs. Accordingly, these dynamics all present a fast decay of the temporal correlations. For such dynamics, the evolution over several correlation times can therefore be described as a sequence of independent fluctuations. Section 3.3.2.1 shows that, for a sequence of i.i.d. variables, an extreme value of the average over the sequence originates from different scenarii depending on the underlying PDF. More importantly, we show that the case of an exponential PDF is a marginal case.

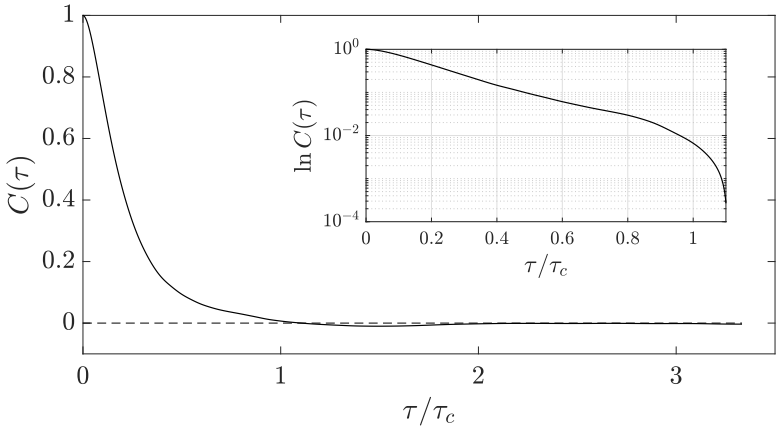
Finally, we illustrate these results by sampling extreme values of the time-average of one dimensional stochastic dynamics with rapidly decaying temporal correlations. It confirms that a process with an exponential PDF is a marginal case, in which neither large fluctuations nor series of positive typical fluctuations preferentially contribute to a very large value of the average. This results are consistent with the observations of section 3.3.1 and figure 3.15 concerning the structure of extreme fluctuations of the average drag.

3.3.2.1 *Extremes of a sequence of i.i.d. variables*

In the previous paragraph, we stressed that the evolution of the drag over time can be viewed as a temporal process with a very fast decay of the correlations over time. Accordingly, a fluctuation of the averaged drag F_T, with $T = 10\tau_c$, may be considered as the result of a sequence of statistically independent fluctuations. Motivated by this observation, we propose in this section a study of the typical extremes for a finite sequence of independent, i.i.d. random variables.

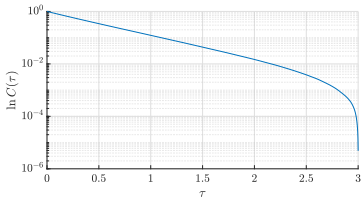


(a) PDF for the drag fluctuations f'_d

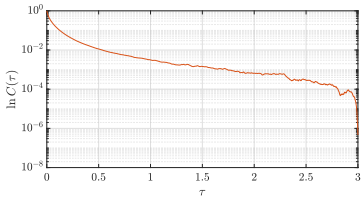


(b) Autocorrelation function of the the instantaneous drag f_d

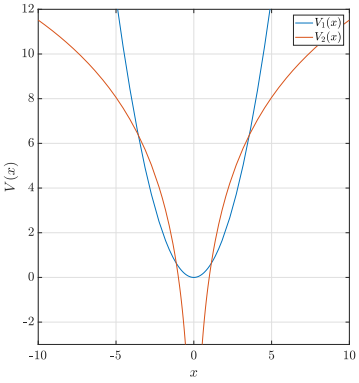
Figure 3.16: **(a)** PDF describing the statistics of the instantaneous drag fluctuations $f'_d = (f_d - \bar{f}_d)/\sigma$. This PDF has been estimated on the basis of a timeseries spanning $T_{tot} = 10^6\tau_c$. $\bar{f}_d$ and σ denote the average and the standard deviation computed over the whole timeseries, respectively. The linear fits highlight the exponential behaviour of the the PDF for rare events. **(b)** Autocorrelation function for the instantaneous drag f_d, defined as $(\mathbb{E}[f_d(t+\tau)f_d(t)] - \mathbb{E}[f_d]^2)/\sigma^2$ and estimated from a timeseries spanning $T_{tot} = 10^6\tau_c$. This figure shows the exponential decay of the correlations over time.



(a) Exponential decay of the correlations



(b) Algebraic decay of the correlations



(c) Potentials $V_1(x)$ and $V_2(x)$

Figure 3.17: Temporal correlations for two stochastic processes described by (3.15) with a PDF $\mathcal{P}(x) \underset{x\to\infty}{\sim} 1/|x|^5$. The difference lies in the choice for the potential term in (3.15). Figure 3.17b illustrates the autocorrelation function of x for $V(x) \equiv V_2(x) = 5\ln(x\tanh(\beta x) + \epsilon) \underset{|x|\to\infty}{\sim} 5\ln(|x|)$ with $\beta = 100$ and $\epsilon = 10^{-2}$. Figure 3.17a illustrates the autocorrelation function for $V(x) \equiv V_1(x) = x^2/2$. One can see that the choice of the potential term has a drastic impact on the range of the temporal correlations. In the first case (figure 3.17b), correlation display an algebraic decay. In the second (figure 3.17b), correlations decay exponentially.

Let $\{X_n\}_{1\leq n\leq N}$ be a sequence of N independent realisations of a $\mathbb{R}$-valued random process described by a PDF $\mathcal{P}$. Furthermore, let a be the sum over the sequence:

$$a = \sum_{n=1}^{N} X_n.$$

In the following we consider two limit cases:

1. The value of the average a/N results from *a single* extreme realisation of the order of $X_n \approx a$. The $(N-1)$ remaining realisations take typical values.

2. The value of the average a/N results from all N realisations in the sequence having roughly the value a/N.

As an illustration, case (1) models extremes such as the one pictured in figure 3.15, in which the high value of the averaged drag results from only two very large fluctuations. In contrast, case (2) is illustrated by figure 3.15, in which the high value of the overall drag results from a sequence of more typical fluctuations.

Let us denote by p_1 the probability that an extreme value of the average over the sequence $\{X_n\}_{1\leq n\leq N}$ results from case (1), and p_2 the probability that it results instead from case (2). Assuming independence of the realisations, one can check that:

$$p_1(\sum_{1}^{N} X_n = a) \approx \mathcal{P}\left(\frac{a}{N}\right)^N \quad \text{and} \quad p_2(\sum_{1}^{N} X_n = a) \approx \mathcal{P}(a) \tag{3.11}$$

The ratio $p_1(a)/p_2(a)$ thus describes the respective contribution of both cases to the extreme value of the average over the sequence of realisations. In the following we consider the case of three different PDFs from which the sequence $\{X_n\}_{1\leq n\leq N}$ is drawn. We illustrate that, in the limit of rare events, this ratio depends on the behaviour of the tail of $\mathcal{P}$.

GAUSSIAN PDF In this case we choose $\mathcal{P}(x) \underset{x\to\infty}{\sim} Ce^{\frac{x^2}{2}}$, with C a prefactor independent of x. As a result,

$$\frac{p_1}{p_2} \underset{a\to\infty}{\sim} Ce^{-\frac{a^2}{2}(1-\frac{1}{N^2})} \underset{a\to\infty}{\to} 0 \tag{3.12}$$

Consider a sequence of N Gaussian distributed random variables which overall average is a large value a/N. Equation (3.12) illustrates that it is very likely that the large average results from all the realisations having roughly the same value close to a/N. By contrast, it is very unlikely that the average is due to a single realisation having a value close to a.

ALGEBRAIC PDF In this case we chose $\mathcal{P}(x) \underset{x\to\infty}{\sim} C/x^{\alpha}$. As a result,

$$p_2(a) \approx \left(\frac{N^{\alpha}}{a^{\alpha}}\right)^{N} \underset{a\to\infty}{\sim} CN^{N\alpha}a^{-N\alpha}$$

The ratio can therefore be expressed as

$$\frac{p_2}{p_1} \underset{a\to\infty}{\sim} CN^{N\alpha} \times a^{\alpha-N\alpha} \underset{a\to\infty}{\to} 0 \tag{3.13}$$

Consider a sequence of N power law distributed random variables which overall average is a large value a/N. Equation (3.13) illustrates that it is very likely that the large average is due to a single realisation having a value close to a. The other realisations take typical values distributed around the average. By contrast, it is very unlikely that the average results from all the realisations having roughly the same value close to a/N.

EXPONENTIAL PDF In this case we chose $\mathcal{P}(x) \underset{x\to\infty}{\sim} Ce^{-\alpha x}$. As a result,

$$\frac{p_2}{p_1} \underset{a\to\infty}{\sim} C\left(e^{-\alpha\frac{a}{N}}\right)^{N} e^{-\alpha a} = 1 \tag{3.14}$$

This case is therefore marginal. Consider a sequence of N i.i.d. variables distributed according to a PDF with an exponential tail. Say the average over a realisation of the sequence is a large value a/N. This value of the average can result from the realisation of a single variable having a value close to a. Additionally, it can also result from all the realisations having roughly the same value close to a/N. Equation (3.13) suggests that these two scenarii are equally likely.

3.3.2.2 *Illustration of extremes for 1D random dynamics with a fast decay of the correlations*

For a finite sequence of i.i.d. variables, we computed in the previous section the probabilities that an unusually high average over the realisations result from either a single very large fluctuation, or from the succession of independent lower amplitude fluctuations. These probabilities depend on the underlying PDF.

In this section we consider simple random processes for which temporal correlations decay very rapidly over time. Recall that this is the case of the temporal process defined by the evolution of the drag over time. For this process, the autocorrelation function was shown to decay exponentially as the lag increases. This is illustrated in figure 3.16b. When a dynamical process displays a fast decay of temporal correlations, a given realisation over several correlation times

can therefore be interpreted as a sequence of independent realisations over one correlation time.

Let $x(t)$ be a $\mathbb{R}$-valued random process defined by the following stochastic differential equation:

$$\dot{x} = \frac{dV}{dx} + \sqrt{2a(x)}\eta(t) \tag{3.15}$$

where the potential term $V(x)$ describes the deterministic part of the dynamics and η is a Gaussian white noise. Furthermore, the amplitude of the stochastic part is allowed to vary with x through the term $a(x)$. The stationary distribution for equation (3.15) can be shown to be of the form **[ref1]**

$$P_S(x) \propto \frac{1}{a(x)} \exp\left(-\int_0^x \frac{1}{a(u)}\frac{dV}{du}\mathrm{d}u\right). \tag{3.16}$$

The rate of decay of the temporal correlations of x is linked to the potential $V(x)$. As an example, consider the process defined by (3.15) and:

$$V(x) \underset{|x|\to\infty}{\sim} \alpha \ln(|x|) \tag{3.17}$$

$$a(x) = 1 \tag{3.18}$$

From equation (3.16), the corresponding stationary PDF in the limit of rare events is algebraic, that is: $\mathcal{P}(x) \underset{|x|\to\infty}{\sim} |x|^{-\alpha}$. Figure 3.17b illustrates that the corresponding correlation function decays slowly over time. This is connected to the logarithmic structure of the potential.

As a second example, let us now consider the process defined by equation (3.15) and

$$V(x) = \frac{x^2}{2} \tag{3.19}$$

$$a(x) = (1 + x^2)/\alpha \tag{3.20}$$

One can check from (3.16) that this choice also leads to an algebraic PDF: $\mathcal{P}(x) \underset{|x|\to\infty}{\sim} |x|^{-\alpha}$. However, in contrast with the first example, figure 3.17a shows that this process displays an exponential decay of temporal correlations.

In the following, we numerically integrate (3.15) over long durations in order to sample extreme fluctuations of the time-average of x. To illustrate the results of section 3.3.2.1, we consider three cases :

GAUSSIAN PDF $V(x) = x^2/2$ and $a(x) = 1$, leading to $\mathcal{P}(x) \propto e^{-x^2/2}$.

ALGEBRAIC PDF $V(x) = x^2/2$ and $a(x) \underset{x\to\infty}{\sim} x^2/\alpha$, leading to

$$\mathcal{P}(x) \underset{x\to\infty}{\propto} 1/|x|^{\alpha}$$

EXPONENTIAL PDF $V(x) \underset{|x|\to\infty}{\sim} |x|$, leading to $\mathcal{P}(x) \underset{x\to\infty}{\propto} e^{-\alpha|x|}$

The corresponding autocorrelation functions all display an exponential decay of the correlations.

Given a timeseries for the instantaneous process $\{x(t)\}_{0\leq t\leq T_{tot}}$, we are interested in extremes for the the time-averaged process defined as

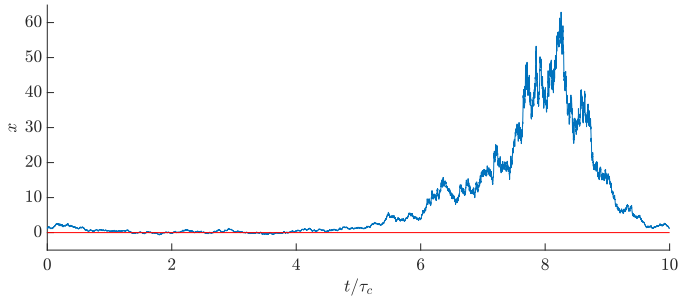
$$X_T(t) = \int_{t-T}^{t} x(t)\mathrm{d}t \text{ for } T \leq t \leq T_{tot} \tag{3.21}$$

In each case, the dynamics described by equation (3.15) has been computed over $T_{tot} = 10^6\tau_c$ where τ_c denotes the correlation time of the process estimated by the vanishing time of the autocorrelation function of the process. Following (3.21), the timeseries for the averaged process has then been obtained by performing a moving average over the timeseries for the instantaneous process $\{x(t)\}_{0\leq t\leq T_{tot}}$.

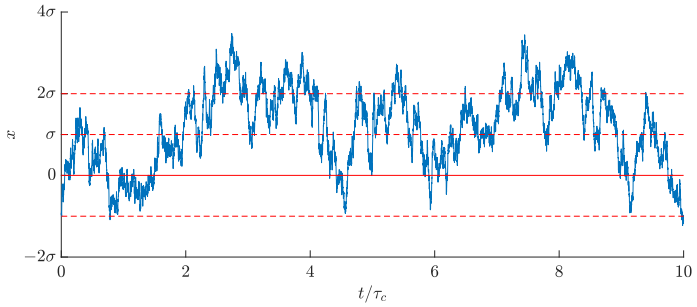
Figure 3.18 illustrates extremes of the time-averaged process over $T = 10\tau_c$, for each of the three statistics. The averaging duration T was set to mimic the averaging of the instantaneous drag in section 3.3.1. It shows that extremes of the averaged process originate from different behaviours, depending on the nature of the PDF for the underlying process. In the case of the algebraic PDF, displayed in figure 3.18a, the high value of the average results from a single, exceptionally large fluctuation while the rest of the timeseries concentrates around typical values. In contrast, for the Gaussian process displayed in figure 3.18b, the extremely high value of the overall average results from a large number of relatively typical *positive* fluctuations over roughly a correlation time. Figure 3.18c illustrates that the exponential PDF is a marginal case. Indeed, the large value of the average results both from unusually large fluctuations of the instantaneous process (around $t = \tau_c$ and $t = 5\tau_c$) and a large number of independent *positive* fluctuations of the order of 2σ.

Recall that the statistics of extreme positive fluctuations of the instantaneous drag f_d are well described by an exponential PDF, as shown in figure 3.16a. Furthermore, the autocorrelation function $C(\tau) = \mathbb{E}[f_d(t)f_d(t+\tau)] - \mathbb{E}[f_d]^2$ was found to decay exponentially as a function of the lag τ. As a consequence, the evolution of the drag over time can be modelled by a stochastic process with short-range temporal correlations, in which the perturbations from small-scale turbulence fluctuations play the role of the noise.

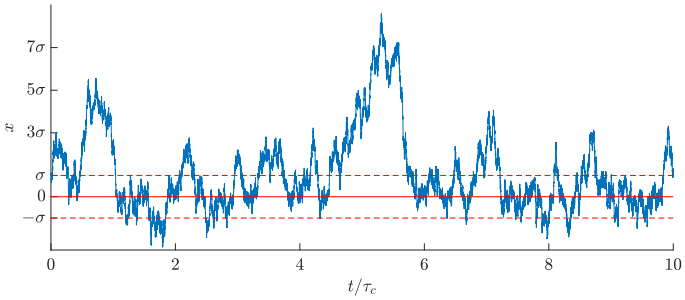
In this context, the interpretation of figure 3.15 on page 69 is consistent with the results from the analysis of the statistical properties of the one-dimensional stochastic dynamics presented in this section. Figure 3.15 illustrates that the temporal structure of the extreme fluc-



(a) Power law process : one unique large fluctuation



(b) Gaussian process : long lasting excursions away from the average



(c) Exponential process : intermediate case

Figure 3.18: Sample timeseries related to extreme fluctuations of the time-average of the process $x(t)$ defined by (3.15) for three stochastic dynamics: a process with a power law stationary PDF(3.18a), a Gaussian process(3.18b) and a process with an exponential PDF(3.18c). The averaging window has been set to $T = 10\tau_c$. The three timeseries displayed in this figure correspond to the highest fluctuations in the averaged timeseries. This figure illustrates that fluctuations for the averaged process originate from different mechanisms depending on the stationary distribution of the process. In all three cases, the average is 0 and materialised by a red straight line.

tuations of the averaged drag could not easily be reduced to either one of the two following configurations:

1. The extreme value of the average results from only a few number of very large fluctuations, while the major part of the timeseries concentrates around typical values. See for instance figure 3.15c.

2. The extreme value of the average results from a unusually large number of independent positive fluctuations, relatively typical with respect to case (1). See for instance figure 3.15a

As a matter of fact, in this section we showed that a process with short-range temporal correlations for which rare events are described by an exponential PDF is a marginal case. As a consequence, extreme fluctuations of averaged drag do not preferentially originate from either scenario (1) or (2).

3.4 GOING FURTHER: THE NEED FOR RARE EVENT ALGORITHMS

Extreme fluctuations of the drag force, whether it is instantaneous or time-averaged, correspond to events in which it departs from typical values to reach a given threshold of very high amplitude. The characterisation of the mechanical processes that lead to such extreme events is crucial for application. Indeed, it leads to a better understanding and/or control on the behaviour of systems immersed in turbulent flows.

3.4.0.1 *Summary of the main results*

In this chapter, we investigated the dynamics of very rare fluctuations of the drag acting on a square cylinder embedded in a two-dimensional channel flow, in which upstream turbulence is generated by a grid. We showed that extreme fluctuations of the instantaneous drag are related to violent pressure drops occurring in the vicinity of the base of the cylinder. By contrast, the upstream pressure does not vary as much. Furthermore, the study of 100 events with a return period larger than 10^4 correlation times suggested that extreme fluctuations result from only two scenarii. The dominant scenario features the interaction of two vortical structures that result in a strong concentration of vorticity near the base of the obstacle. This scenario describes the majority of the events, and in particular the most extreme.

In section 3.3, we illustrated the timeseries corresponding to very high values of the time-averaged drag over 10 correlation times. It suggested that such extreme values can result from two mechanisms. On the one hand, the occurrence of a small number of extreme fluctuations of the instantaneous drag, while the rest of the timeseries displays typical, relatively small positive fluctuations. On the other hand, a larger number of fluctuations of moderate amplitude which

sum eventually leads to a very high value of the averaged drag. Interestingly, these two scenarii have been found to play an equal role in the sampled extremes. In section 3.3.2, we illustrated the relative importance of these two scenarii for extreme values of the average of three different one-dimensional stochastic processes. We showed that, for a process with an exponential tail, the two scenarii are equally likely.

3.4.0.2 *Rare event algorithms*

In this chapter, extreme events have been sampled by means of direct sampling. That is, we simulated the flow over very long durations in order to sample a significant number of events that we considered as rare. We stress that this approach is made possible by the relative simplicity of the flow, which allows us to compute very long timeseries in a reasonable amount of time. Naturally, such a direct sampling is impossible for more complex flows, such as three-dimensional flows around complex geometries at very-high Reynolds numbers. Additionally, the study presented in this chapter is limited. Indeed, to sample a larger number of fluctuations, or rarer fluctuations, longer simulations are required. For instance, to perform an analysis over 10 times more events with similar amplitudes, the dynamics must be simulated over a duration 10 times longer. The control simulations involved in this study have been performed in a wallclock time of the order of the week. Ten times longer timeseries therefore lead to a wallclock time of several months. In addition, it would generate a tremendous amount of data, most of which would be irrelevant to the study of extremes.

In this thesis we propose a novel route for the computational study of extremes in turbulent flows. Instead of relying over very long simulations, we consider the application of *rare events algorithms*, originating from statistical physics. The objective of these algorithms is to modify the statistics of the sampling *in a controlled way*, so as to favour the occurrence of rare events. To this date, such algorithms have been successfully applied to relatively simple dynamics, compared to turbulent flows [21, 54, 158]. Their relevance for complex deterministic dynamics, such as turbulent flows, remains unclear. This question is addressed in the following chapters.

Part II

RARE EVENTS ALGORITHMS

The study of rare events by means of direct sampling, *i.e.* very long simulations, is limited. Indeed, targeting events with a specific probability, the dynamics must be simulated on a duration that grows like the inverse square-root of this probability. For turbulent flows, which numerical simulation often require important computing time, such direct approach is simply unfeasible. In the subsequent chapters we assess the applicability and efficiency of two different rare events algorithms for rare event sampling in turbulent flows. The underlying idea of these algorithms is to modify the sampling of trajectory space by following an ensemble of weighted copies of the dynamics. Even though such techniques have been successfully applied to rather simplified stochastic dynamics, it is not clear if they are useful for complex deterministic systems, such as turbulent flows.

4

THE GIARDINA–KURCHAN–TAILLEUR–LECOMTE ALGORITHM

The previous chapter illustrated the study of extremes drag events, on the basis of a very long numerical simulation of test flow (2). This could only be achieved thanks to the relative simplicity of the flow, allowing for the integration of the dynamics over a long duration. However, such an approach is hopeless for more complex flows in an industrial or environmental context, such as high Reynolds flows past cars or wind turbines. Indeed, such long simulation would require tremendous computational resources and computation time, if even technically feasible. In such cases, performing highly-resolved simulation of the flow over a few turnover times is already a challenge in itself. How to optimise computational effort to capture dynamical events with a return time much larger than the typical timescale of the flow ?

This chapter introduces the GKTL algorithm, a rare event algorithm designed to compute large deviation rate functions by means of *importance sampling*. Roughly speaking, the GKTL algorithm allows for a biased sampling of trajectories, selecting preferentially rare trajectories considered as more *important*. Eventually, it results in sampling flow trajectories according to a biased probability measure, for which typical events are rare events with respect to the original dynamical measure of the flow. In contrast, directly simulating the flow over long durations amounts to unbiased sampling, often referred to as *direct sampling* or *direct simulation*.

In section 4.1.1, we give a brief introduction to *importance sampling*, a general statistical method at the basis of the GKTL algorithm. Then, section 4.1.2 describes the algorithm itself, as well as how importance sampling is effectively achieved in order to sample rare trajectories. The relation between the GKTL algorithm and large deviation theory is highlighted in section 4.1.3. As an illustration, the GKTL algorithm is applied on the OU process in section 4.1.4. In section 4.2, we specifically discuss the implementation of the GKTL algorithm for turbulent dynamics.

4.1 THE GKTL ALGORITHM

Over the past few decades, there has been an intense renewal of interest in *large deviation theory*. Indeed, it offers an insightful and pow-

erful description of both equilibrium and non-equilibrium statistical physics [165]. Accordingly, important research effort has been made towards the development of computational tools capable of computing *large deviation rate functions*, that are analogous to free energies in non-equilibrium systems. Such quantity encodes the statistical properties of the fluctuations, including rare events.

Among these numerical methods is the *cloning algorithm* [57, 59, 103], introduced in 2006 by Giardina, Kurchan and Peliti [59] to compute large deviations of time averaged quantities in discrete Markov processes. A year later, similar ideas were applied to the simulation of trajectories of atypical chaoticity in nonlinear deterministic systems, such as the Fermi-Pasta-Ulam-Tsinguou chain [158]. Note that the algorithmic procedure is actually rooted in older ideas, introduced in Diffusion Monte Carlo [89] and *Go with the Winners* algorithms [68, 167].

The algorithms rely on the parallel evolution of an ensemble of independent copies of the system. At a fixed period τ, copies are either discarded from the ensemble or duplicated, at a rate that depends on their history over the period. In this way, the population is dynamically enriched with copies exhibiting large fluctuations of a given observable of the dynamics. This leads to *importance sampling* in trajectory space, *i.e.* the generation of an ensemble of trajectories in phase space that is drawn for a biased path measure that favours the rare events of interest. The cloning algorithm has been given different names in the literature: the Giardina–Kurchan–Tailleur–Lecomte (GKTL) algorithm, the cloning algorithm or simply the Diffusion Monte Carlo algorithm. In the following, it is referred to as the GKTL algorithm. Note that a very similar method is the Del-Moral-Garnier algorithm [118].

Because it works directly in trajectory space, the GKTL algorithm does not rely on the *a priori* knowledge of the stationary distribution. It is therefore directly applicable to non-equilibrium dynamics. The GKTL outputs an ensemble of trajectories of the dynamical system that are associated to large fluctuations of a time-averaged observable of interest. In addition, a statistical weight is attributed to each sampled trajectory, allowing for the computation of expectation values according to the unbiased probability.

In the following section we give a brief introduction to the concept of *importance sampling*, at the basis of the GKTL algorithm.

4.1.1 *Importance Sampling*

Importance sampling [22, 163] is a general strategy for reducing the variance of statistical estimators:

$$\mathbb{E}[f]_\rho = \int \mathrm{d}X f(X)\rho(X) \approx \frac{1}{N}\sum_{n=1}^{N} f(X_n). \tag{4.1}$$

Where f denotes the value of an observable. The expectation value of f with respect to ρ, *i.e.* $\mathbb{E}_\rho[f]$, is estimated on the basis of a sample mean over a set $\{X_n\}_{1\leq n\leq N}$ of independent realisations drawn from ρ. This follows from the law of large numbers. For a fixed N, the statistical error affecting the sample mean in (4.1) can be reduced by sampling realisations from a different PDF, which we denote by $\tilde{\rho}$. This probability distribution is chosen so that it is centred on realisations that contribute the most the average in (4.1).

The very general idea of importance sampling appears in numerous fields. An example is the numerical estimation of thermal averages in equilibrium systems, such as

$$\langle O\rangle_\beta = \frac{\int \mathrm{d}\mathbf{r} e^{-\beta U(\mathbf{r})}O(\mathbf{r})}{\int \mathrm{d}\mathbf{r} e^{-\beta U(\mathbf{r})}} = \int \mathrm{d}\mathbf{r}\mathcal{P}_\beta^{(G)}(\mathbf{r})O(\mathbf{r}), \tag{4.2}$$

where $\mathcal{P}_\beta^{(G)}$ denotes the Gibbs measure associated to the inverse temperature $\beta = 1/k_bT$. The integrals runs over a space of states denoted by $\mathbf{r}$. The weight of these states depends on the temperature of the system and are given by the Boltzmann factor $e^{-\beta U(\mathbf{r})}$ where $U(\mathbf{r})$ is the energy of the system in state $\mathbf{r}$. In order to compute the average (4.2), one could think of generating states $\mathbf{r}$ uniformly and accept or reject them with a probability proportional to $\mathcal{P}_\beta^{(G)}$. This direct sampling approach is however unpractical. Indeed, in practice the state space contains an extremely large number of states, and only a small subset actually contributes to the average in (4.2). As a result, virtually all proposed states would end up rejected, resulting in a very inefficient sampling. Instead, a common approach to the computation of thermal averages is Markov Chain Monte Carlo (MCMC) sampling. In this approach a sequence of correlated realisations is constructed, which stationary distribution is $\mathcal{P}_\beta^{(G)}$. This process is commonly referred to as *importance sampling* in the statistical physics community. The most famous MCMC sampling algorithm is the *Metropolis-Hastings* algorithm [34, 122].

In the context of rare event sampling, the tails of a distribution ρ can be efficiently sampled using a biased distribution $\tilde{\rho}$ for which rare events of ρ are typical. In the following we describe importance sampling in the context of rare event sampling. We first motivate the method by illustrating the inefficiency of a direct approach in which

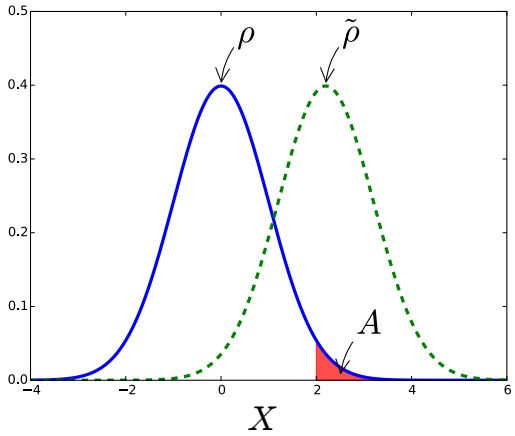


Figure 4.1: Illustration of importance sampling. Sampling of rare events in the set A coloured in red is made efficient by sampling a biased distribution $\tilde{\rho}$, for which such events are typical.

realisations are drawn from the original PDF ρ. We then discuss the variance reduction resulting from the sampling from an alternative distribution. In particular, we show that there exist an optimal choice of this distribution for which zero variance is achieved.

4.1.1.1 *Importance sampling for rare events*

Let X be a realisation of a random process described by the probability density ρ. Furthermore, let A be a region in the tails of ρ, such that $p_A = \mathcal{P}(X \in A) \ll 1$. In the following we address the numerical computation of the probability that a realisation falls into the region A:

$$p_A = \mathbb{E}_\rho[1_A] = \int dX \rho(X) 1_A(X), \tag{4.3}$$

with $1_A = 1$ if $X \in A$ and $1_A = 0$ otherwise. A most straightforward way of computing such probability is to draw an ensemble of independent realisations $\{X_n\}_{1 \leq n \leq N}$ and to count how many fall into the region A. This results in a direct estimator of (4.3), analog to (4.1):

$$\hat{p}_A = \frac{1}{N} \sum_{n=1}^{N} 1_A(X_n). \tag{4.4}$$

Denoting the standard deviation of $\hat{p}_A$ by $\sigma(\hat{p}_A)$, the relative error on the estimate is given by $err(\hat{p}_A) = \sigma(\hat{p}_A)/\hat{p}_A$. For small probabilities p_A, the relative error reads

$$err(\hat{p}_A) \approx \frac{1}{\sqrt{p_A N}}, \tag{4.5}$$

with N the number of independent realisations. In order to maintain the relative error constant as p_A becomes smaller and smaller, the

sample size must then grow as $1/p_A$. Direct sampling of rare events thus involves tremendous computational effort and is prohibitive for computationally demanding dynamics such as turbulent flows.

Importance sampling instead consists in drawing realisations from a different PDF $\tilde{\rho}$, and to rewrite (4.3) as

$$p_A = \int dX \tilde{\rho}(X) \frac{\rho(X)}{\tilde{\rho}(x)} 1_A(X) = \tilde{E}[\mathcal{L}(X) 1_A(X)], \tag{4.6}$$

where $\tilde{\mathbb{E}}[\cdot]$ denotes the expectation value with respect to $\tilde{\rho}$ and $\mathcal{L}(X) = \rho(X)/\tilde{\rho}(X)$. The corresponding *importance sampling estimator* for p_A is

$$\tilde{p}_A = \frac{1}{N} \sum_{n=1}^{N} \mathcal{L}(\tilde{X}_n) 1_A(\tilde{X}_n). \tag{4.7}$$

where the $\{\tilde{X}_n\}_{1 \leq n \leq N}$ are drawn from $\tilde{\rho}$. One can check that $\tilde{p_A}$ is also an unbiased estimator of p_A, *i.e.* $\tilde{\mathbb{E}}[\tilde{p}_A] = \mathbb{E}[\hat{p}_A]$. Why bother introducing $\tilde{\rho}$? This can be understood by considering the variance of $\tilde{p}_A$:

$$\begin{aligned} \sigma^2(\tilde{p}_A) &= \frac{\tilde{\mathbb{E}}[\mathcal{L}^2(X) 1_A(X)] - \tilde{\mathbb{E}}^2[\mathcal{L}(X) 1_A(X)]}{N} \\ &= \frac{1}{N} \left(\int dX \rho(X) L(X) 1_A(X) - p_A^2 \right). \end{aligned} \tag{4.8}$$

One can see that, choosing $\mathcal{L}(X) = \mathcal{L}^{(opt)}(X) = p_A 1_A(X)$, the variance vanishes. This amounts to choose $\tilde{\rho}(X) = \frac{\rho(X)}{p_A} 1_A(X)$, a PDF in which all the mass is located in the region A. Naturally, such a ratio $\mathcal{L}^{(opt)}(X)$ cannot be chosen in practice, as it is based on the quantity one wants to compute, *i.e.* p_A. The underlying idea of importance sampling is that significant variance reduction can still be achieved by choosing $\tilde{\rho}$ so that $\mathcal{L}$ is close to $\mathcal{L}^{(opt)}$. In other words, this means choosing $\mathcal{L}$ so that the rare events of interest are typical for $\tilde{\rho}$, thus significantly reducing statistical errors.

In the remaining of this section we explain how importance sampling can be achieved at the level of the trajectories of a dynamical system, such as a turbulent flow. The choice of the biased measure relies on an *exponential change of measure*, which is motivated by large deviation theory related to the asymptotics for the probability of a large-time averaged observable of the dynamics.

4.1.1.2 *Importance sampling extremes of dynamical observables*

For dynamical systems, importance sampling must be implemented at the level of trajectories. A trajectory $\{\mathbf{X}(t)\}_{0 \leq t \leq T_a}$ refers to a dynamical solution of the numerical model over a time-interval $[0; T_a]$. Let o

be an observable over the trajectories and O_{T_a} its time-integral over a duration T_a

$$O_{T_a}[\{\mathbf{X}(t)\}_{0\leq t\leq T_a}] = \int_0^{T_a} o[\mathbf{X}(t)]\mathrm{d}t.$$

The GKTL algorithm is a procedure to sample trajectories according to an importance ratio $\mathcal{L}_k[\{\mathbf{X}(t)\}_{0\leq t\leq T_a}]$, that gives a heavier weight to trajectories having a higher value of O_{T_a}.

More precisely, let $\mathcal{P}_0\left(\{\mathbf{X}(t)\}_{0\leq t\leq T_a} = \{\mathbf{x}(t)\}_{0\leq t\leq T_a}\right)$ denote the stationary path measure of the system. It is the probability density that a trajectory $\{\mathbf{X}(t)\}_{0\leq t\leq T_a}$ corresponds to a given trajectory $\{\mathbf{x}(t)\}_{0\leq t\leq T_a}$. For the sake of convenience, in the following we write $\mathcal{P}_0\left(\{\mathbf{x}(t)\}_{0\leq t\leq T_a}\right)$ as a shorthand notation for $\mathcal{P}_0\left(\{\mathbf{X}(t)\}_{0\leq t\leq T_a} = \{\mathbf{x}(t)\}_{0\leq t\leq T_a}\right)$. The GKTL algorithm samples trajectories according to [120, 130]

$$\mathcal{P}_k\left(\{\mathbf{x}(t)\}_{0\leq t\leq T_a}\right) = \underbrace{\frac{\exp\left(k\int_0^{T_a} o[\mathbf{x}(t)]\mathrm{d}t\right)}{\mathbb{E}_{\mathcal{P}_0}\left[\exp\left(k\int_0^{T_a} o[\mathbf{x}(t)]\mathrm{d}t\right)\right]}}_{\mathcal{L}_k[\{\mathbf{X}(t)\}_{0\leq t\leq T_a}]} \mathcal{P}_0\left(\{\mathbf{x}(t)\}_{0\leq t\leq T_a}\right) \tag{4.9}$$

where $\mathbb{E}_{\mathcal{P}_0}[.]$ denotes the ensemble average with respect to the path measure $\mathcal{P}_0$. The distribution $\mathcal{P}_k$ is thus analogous to the biased distribution $\tilde{\rho}$ featured in figure 4.1, and $\mathcal{P}_0$ to ρ. The importance ratio $\mathcal{L}_k[\{\mathbf{X}(t)\}_{0\leq t\leq T_a}]$ depends on a free parameter k, that rules how much weight is given to trajectories having larger values of O_{T_a}. The form of the importance ratio originates from the large deviation behaviour of the probability for the averaged observable $\frac{1}{T_a}\int_0^{T_a} o[\mathbf{x}(t)]\mathrm{d}t$, in the limit of large T_a. The link between the GKTL algorithm and large deviation theory is highlighted in section 4.1.3.

4.1.2 *The GKTL algorithm*

This section describes the algorithm that results in the sampling of trajectories according to the biased measure $\mathcal{P}_k$, defined in (4.9). The algorithm relies on the simulation of an ensemble of copies in parallel, from an initial time $t = 0$ to a final time $t = T_a$. For a turbulent flow, this means carrying out an ensemble of simulations, starting on independent initial conditions. Along their evolution in time, copies are either cloned or discarded from the ensemble with a period $\tau < T_a$, according to their history. More precisely, trajectories are selected according to the value the time average of a given observable o. In out application, o will be the instantaneous drag acting on an obstacle immersed in a turbulent flow. The algorithm thus consists in $M = T_a/\tau$ consecutive stages, each one consisting of the independent evolution of the copies from $n\tau$ to $(n+1)\tau$, followed by a selection procedure.

The algorithm is designed so that, at a time $t_n = n\tau$, a copy j is given a weight

$$W_j = \frac{\exp\left(k \int_{n\tau}^{(n+1)\tau} o_j[\mathbf{x}(t)]dt\right)}{\frac{1}{N_c}\sum_{l=1}^{N_c} \exp\left(k \int_{n\tau}^{(n+1)\tau} o_l[\mathbf{x}(t)]dt\right)} \tag{4.10}$$

As a result, *in average*, copy j is replaced by W_j clones[1]. It will be checked further that this amounts to sample trajectories of duration T_a according to the biased measure $\mathcal{P}_k$, see equation (4.9).

4.1.2.1 *Description of the algorithm*

Let there be N_c copies of a dynamical system of interest. The corresponding trajectories over the interval $[0; T_a]$ are denoted by $\{\mathbf{x}^{(j)}(t)\}_{0\leq t\leq T_a}$. Additionally, let us partition the interval $[0; T_a]$ into sub-intervals of length τ. The GKTL algorithm can be outlined as follows :

- Initialisation : $\mathbf{x}^{(j)}(t=0) = \mathbf{x}_0^{(j)}, \ 1 \leq j \leq N_c$
- FOR n from 1 to $M = \frac{T_a}{\tau}$
  - FOR j from 1 to N_c
    - Compute dynamics from $t = n\tau$ to $t = (n+1)\tau$ for clone j
    - Give clone j a weight $w_j = \exp\left(k \int_{n\tau}^{(n+1)\tau} o[\mathbf{x}^{(j)}(t)]\mathrm{d}t\right)$
  - Compute the sum of the weights over the j copies :

$$Z_n = \frac{1}{N_c}\sum_{j=1}^{N_c} \exp\left(k \int_{n\tau}^{(n+1)\tau} o[\mathbf{x}^{(j)}(t)]\mathrm{d}t\right)$$

  - FOR j from 1 to N_c, copy j is replicated into m_j clones with

$$m_j = E[\frac{w_j}{Z_n} + \epsilon]$$

    where $E[\cdot]$ represent the integer part and ϵ is a random number drawn uniformly in $[0; 1[$. If $m_j = 0$, the copy is discarded.

The procedure described above leads to effectively sampling trajectories according to the biased measure $\mathcal{P}^{(k)}$. Indeed, the selection stage acts like a bias that modifies the probability of observing a trajectory in the ensemble.

1 The precision *in average* is required because W_j is not necessarily an integer. See the full description of the algorithm further down.

To illustrate this, let us compute the probability of observing a given trajectory $\{\mathbf{x}(t)\}_{0\leq t\leq\tau}$ in the ensemble after the first selection stage, at time $t=\tau$. It can be done by considering how the selection modifies the original probability $\mathcal{P}_0$. In the limit of large N_c,

$$Z_1 = \frac{1}{N_c}\sum_{j=1}^{N_c} e^{k\int_0^\tau o[\mathbf{x}^{(j)}(t)]\mathrm{d}t} \underset{N_c\to\infty}{\sim} \mathbb{E}_{\mathcal{P}_0}\left[e^{k\int_0^\tau o[\mathbf{X}^{(j)}(t)]\mathrm{d}t}\right], \tag{4.11}$$

where the empirical average over the N_c independent and identically distributed random variables $\{e^{k\int_0^\tau o(X^{(j)}(t))\mathrm{d}t}\}_{1\leq j\leq N_c}$ approximates the expectation value. Using the definition of the weights (4.10), one finds

$$\mathcal{P}_k^{(1)}\left(\{\mathbf{x}(t)\}_{0\leq t\leq\tau}\right) \underset{N_c\to\infty}{\sim} \frac{e^{k\int_0^\tau o[\mathbf{x}(t)]\mathrm{d}t}}{\mathbb{E}\left[e^{k\int_0^\tau o[\mathbf{x}(t)]\mathrm{d}t}\right]_{\mathcal{P}_0}}\mathcal{P}_0\left(\{\mathbf{x}(t)\}_{0\leq t\leq\tau}\right) \tag{4.12}$$

The procedure is then iterated over the second evolution and selection stage, and $\mathcal{P}_k$ thus becomes

$$\mathcal{P}_k^{(2)}\left(\{\mathbf{x}(t)\}_{0\leq t\leq 2\tau}\right) \underset{N_c\to\infty}{\sim} \frac{e^{k\int_0^{2\tau} o[\mathbf{x}(t)]\mathrm{d}t}}{\mathbb{E}\left[e^{k\int_0^\tau o[\mathbf{x}(t)]\mathrm{d}t}\right]_{\mathcal{P}_0}\mathbb{E}\left[e^{k\int_\tau^{2\tau} o[\mathbf{x}(t)]\mathrm{d}t}\right]_{\mathcal{P}_k^{(1)}}}\mathcal{P}_0\left(\{\mathbf{x}(t)\}_{0\leq t\leq 2\tau}\right) \tag{4.13}$$

Furthermore, one can see that from (4.13):

$$\mathbb{E}\left[e^{k\int_0^\tau o[\mathbf{x}(t)]\mathrm{d}t}\right]_{\mathcal{P}_0}\mathbb{E}\left[e^{k\int_\tau^{2\tau} o[\mathbf{x}(t)]\mathrm{d}t}\right]_{\mathcal{P}_k^{(1)}} = \mathbb{E}\left[e^{k\int_0^{2\tau} o[\mathbf{x}(t)]\mathrm{d}t}\right]_{\mathcal{P}_0}. \tag{4.14}$$

Equation (4.9) then follows by induction, in the limit $N_c\to\infty$.

At this point, an important subtlety must be stressed. Relation (4.11) holds true after the *first* selection stage, if the N_c trajectories start on independent initial conditions. However, in general, trajectories in the ensemble are not mutually independent and the law of large numbers does not apply. Still, it can be shown [117] that (4.11) holds true for subsequent iterations. This has been assessed for a whole family of genealogical algorithms, including this one. An additional result is that the typical relative error entailed by this approximation is of order of $1/\sqrt{N_c}$.

4.1.3 *The GKTL algorithm to compute large deviation rate functions*

The cloning algorithm was first designed to compute large deviations rate functions numerically. In the limit of large observation times, the tail statistics of additive observables can be described by *large deviation theory*. The central object of large deviation theory is

the large deviation *rate function*, which describes the rate of decay of the probability as fluctuations from the typical value gets more and more rare. Let O_T be an observable defined as a time-integral as such $O_T = \int_0^T o(t)\mathrm{d}t$. It can be shown that the probability distribution verifies a *large deviation principle*, that is:

$$P\left(\frac{1}{T}\int_0^T o(t)\mathrm{d}t = a\right) \underset{T\to\infty}{\asymp} e^{-TI(a)} \tag{4.15}$$

where the symbol $f_T \underset{T\to\infty}{\asymp} g_T$ denotes logarithmic equivalence as T goes to infinity: $\ln(f_T) \underset{T\to\infty}{\sim} \ln(g_T)$. The large deviation principle can be interpreted as the distribution of O_T/T getting more peaked as T grows, and asymptotically described by a unique rate function $I(a)$. See chapter 1 for a brief introduction to large deviation theory.

As a matter of fact, the GKTL algorithm does not directly compute the rate function. Instead, it yields its Legendre transform, which corresponds to the SCGF for the observable O_T. The SCGF is a useful object when working with large deviations. It is discussed in the next section.

4.1.3.1 *The Gartner-Ellis theorem*

When working with a probability distribution, say $\mathcal{P}(x)$, it often proves useful to introduce the *cumulant generating function* defined as $\ln \mathbb{E}[e^{kx}]$. In view of (4.15), large deviation theory makes extensive use of the *scaled* cumulant generating function defined as

$$\lambda(k) = \lim_{T\to\infty} \frac{1}{T} \ln \mathbb{E}[e^{Tkx}]. \tag{4.16}$$

This definition can be written as

$$\mathbb{E}[e^{Tkx}] \underset{T\to\infty}{\asymp} e^{T\lambda(k)}.$$

Therefore, the large deviation principle (4.15) leads to

$$\int e^{T(ka-I(a))}\mathrm{d}a \underset{T\to\infty}{\asymp} e^{T\lambda(k)}.$$

In the limit $T \to \infty$, a saddle point approximation writes

$$\lambda(k) = \sup_a [ka - I(a)]. \tag{4.17}$$

The scaled cumulant generating function is often easier to compute than the rate function. For instance in equilibrium statistical physics, numerous methods are available to compute free energies, which corresponds to the cumulant generating function of minus the entropy. An important theoretical question is then to determine under which hypothesis relation (4.17) can be inverted.

The Gärtner-Ellis theorem states that, *if* λ *is differentiable on* $\mathbb{R}$, then the rate function I exists and is the Legendre transform of λ :

$$I(a) = \sup_k [ka - \lambda(k) | k \in \mathbb{R}] \tag{4.18}$$

Therefore, a common strategy to compute rate function is to first compute the SCGF, and deduce the rate function from (4.18). To that end, the GKTL algorithm yields the SCGF.

4.1.3.2 *The SCGF from the GKTL algorithm*

Looking back at the definition of the biased measure $\mathcal{P}^{(k)}$(4.9), one notices that the denominator of the importance ratio can be linked to the SCGF $\lambda(k)$. It can be seen from equations (4.13) and (4.14) that it corresponds to the product of the normalisation factors for each selection/mutation procedure, defined in relation (4.11)

$$Z(k,T) = Z_1 Z_2 ... Z_{n-1} Z_n \underset{N_c \to \infty}{\sim} \mathbb{E}\left[e^{k \int_0^{T_a} o[\mathbf{X}(t)] \mathrm{d}t} \right]_{\mathcal{P}_0} \underset{T_a \to \infty}{\sim} e^{T\lambda(k)}. \tag{4.19}$$

The SCGF can thus be computed from the normalizers Z_n as

$$\lambda(k) \approx \frac{1}{T} \ln \prod_{n=1}^{N} Z_n, \tag{4.20}$$

in the limit of large number of copies N_c and integration time T_a.

Results from convexity analysis yield that *if* λ is differentiable, *e.g.* the Gärtner-Ellis theorem applies, *then* the rate function is convex. Equation (4.18) then translates into

$$I(a) = k(a)a - \lambda(k(a)), \quad \text{whith } k(a) \text{ defined by } \lambda'(k(a)) - a = 0, \tag{4.21}$$

which can be used to compute the rate function.

In the limit $T \to \infty$, one can show from (4.16) that the expectation value of the time-average O_T/T with respect to the biased distribution $\mathcal{P}_k \propto e^{kO_T}\mathcal{P}_0$ verifies

$$\mathbb{E}[O_T/T]_{\mathcal{P}_k} \underset{T \to \infty}{\to} \lambda'(k). \tag{4.22}$$

4.1.4 *Illustration on the Ornstein–Ulhenbeck process*

As an illustration of the GKTL algorithm, we now present its application to a simple one-dimensional stochastic process for which the large deviation rate function and corresponding SCGF can be derived

analytically. In the following we consider an OU process $x(t)$ defined by the following stochastic differential equation:

$$\dot{x}(t) = -x(t) + \sqrt{2}\eta(t), \tag{4.23}$$

where η is a Gaussian white noise. In this case the autocorrelation function is

$$C(\tau) = \frac{\mathbb{E}[x(t)x(t+\tau)]}{\sigma^2} = e^{-\tau}, \tag{4.24}$$

where $\sigma^2 = \mathbb{E}[x^2]$. Note that $\mathbb{E}[x] = 0$. We now write the rate function explicitly. Let us consider the time-averaged process X_T defined as

$$X_T(t) = \frac{1}{T}\int_t^{t+T} x(t)\mathrm{d}t. \tag{4.25}$$

The PDF for X_T can be shown to be Gaussian with mean $\mu = 0$ and standard deviation σ_T. Furthermore, one can show that, in the limit of large integration times:

$$\sigma_T^2 \underset{T\to\infty}{\sim} \frac{2}{T} \tag{4.26}$$

As a result, the PDF of X_T writes:

$$\lim_{T\to\infty} \mathcal{P}(X_T = a) = \sqrt{\frac{T}{4\pi}} \exp\left(-T\frac{a^2}{4}\right) \tag{4.27}$$

The rate function $I(a)$ describing the decay of the probability for the fluctuations of X_T in the limit of large T is thus:

$$I(a) = \frac{a^2}{4} \tag{4.28}$$

and the corresponding SCGF is given by (4.17):

$$\lambda(k) = \sup_a [ka - I(a)] = k^2 \tag{4.29}$$

In order to illustrate the GKTL algorithm, we first compute $\lambda(k)$ from a direct sampling approach. More precisely, we generate a very long timeseries $\{x(t)\}_{0\leq t\leq T_{tot}}$ by simulating the process (4.23) over $T_{tot} = 10^6$. An estimate of $\lambda(k)$ is then computed on the basis of this timeseries. The estimation of a SCGF from a timeseries is discussed in chapter 5. We refer to this estimate as the *direct estimate*. this estimate can only accurately estimate $\lambda(k)$ on a finite range $k \in [k_{min}; k_{max}]$. Indeed, for values $k > k_{max}$, or equivalently $k < k_{min}$, the fluctuations having a major contribution in the average in (4.16) are too rare to be sampled in statistically significant numbers. We now compute

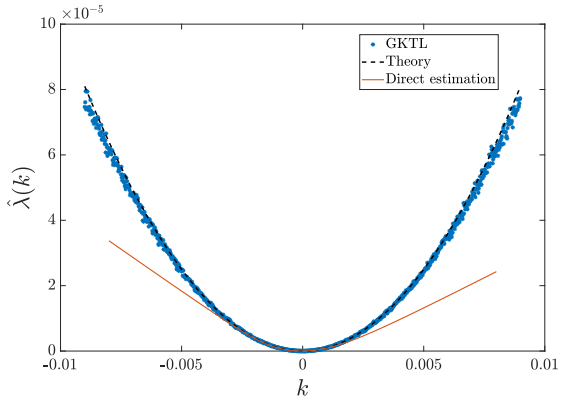


Figure 4.2: Estimation of the Scaled Cumulant Generating Function (SCGF) from a direct estimation and from the GKTL algorithm. A first estimate is computed from a timeseries $\{x(t)\}_{0\leq t\leq T_{tot}}$ with $T_{tot} = 10^4$ (continuous line). The blue stars represent the result (4.20) of 1000 independent runs of the GKTL algorithm for various values of k. Each run ha a computational cost of 10^4. This figure illustrate that the GKTL algorithm is capable of computing the SCGF for values of k, unreachable from a direct sampling approach, for a fixed computational cost. The rate function $I(a)$ can be deduced from the estimate of the SCGF using (4.21)

an estimate of λ with the GKTL algorithm. We use $N_c = 256$ copies. The total length of the trajectories is set to $T_a = 40$, and the cloning period is set to $\tau = 1$. As a result, the evolution/cloning sequence is repeated $M = 40$ times. We define the *computational cost* of this experiment as the total time over the process (4.23) has been simulated: $Nc \times T_a \approx 10^4$. Importantly, it corresponds to the computational cost of the direct estimate: $T_{tot} = 10^4$.

Figure 4.2 shows that, *for a similar computational cost*, the GKTL algorithm provides a far better estimate of the rate function. For values in the vicinity of $k = 0$, both the direct estimation and the GKTL algorithm provide reliable estimates. In this region, roughly $-0.002 \leq k \leq 0.002$, typical fluctuations largely contribute to the average in (4.16). For values further away from $k = 0$ however, the direct estimate is unable to yield an accurate estimate, as too few fluctuations are sampled. By contrast, the GKTL algorithm biases the sampling of these fluctuations through the cloning mechanism, and therefore provides a much better estimate. The GKTL algorithms preferentially selects copies that display short-lived fluctuations over the cloning period τ. As a results, it samples trajectories for which the succession of short lived fluctuations over τ results in a long-lived fluctuation over $T_a = M\tau$. to illustrate this, we consider the time-averaged process X_T defined in (4.25) with $T = 40$, a value large enough so that the large deviation limit $T \to \infty$ can be considered valid.

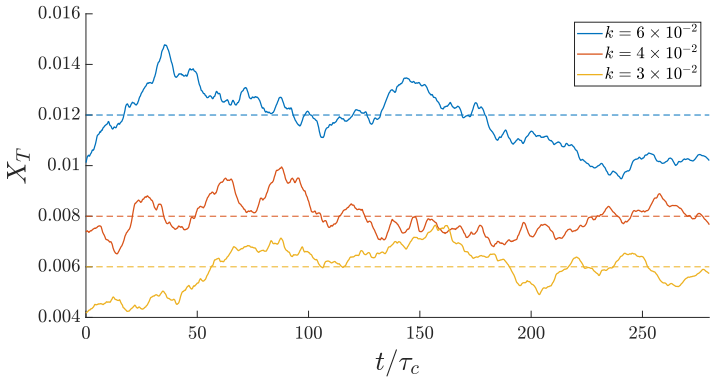


Figure 4.3: Average value of the moving average X_T over the $N_c = 256$ trajectories sampled by the GKTL algorithm for three different values of the bias k. The value of the averaging window is set to $T = 40$, a value large enough so that the large deviation limit $T \to \infty$ can be considered valid. As a matter of fact, the consistency of this approximation is verified by the typical values of X_T along the trajectories, which verifies equation (4.22). The duration of the trajectories sampled by the algorithm is $T_a = 280$.

Figure 4.3 displays the average of X_T along the trajectories sampled by the GKTL algorithm with $T_a = 280$. The average is computed as an empirical mean over the N_c trajectories:

$$\mathbb{E}[X_T]_{\mathcal{P}_k}(t) \approx \frac{1}{N_c} \sum_{j=1}^{N_c} X_T^{(j)}(t) \mathrm{d}t \quad t \in [T; T_a] \tag{4.30}$$

Where $X_T^{(j)}(t)$ denotes the moving average over a duration T computed over the trajectory $\{x^{(j)}(t)\}_{0 \leq t \leq T_a}$ for copy j, defined over $[T; T_a]$. The subscript $\mathcal{P}_k$ denotes the ensemble average with respect to the biased measure $\mathcal{P}_k$, see equation (4.9) for a definition of $\mathcal{P}_k$. Figure 4.3 illustrates the property (4.22). Indeed, in the large time limit, the average value with respect to $\mathcal{P}_k$ is $\lambda'(k)$. In this example, it follows from (4.29) that $\lambda'(k) = k$.

4.1.5 *Can the GKTL algorithm provide similar results for turbulent flows ?*

The GKTL algorithm achieves importance sampling at the level of trajectories in dynamical systems. Trajectories are sampled according to the biased measure $\mathcal{P}_k$ (4.9), instead of the stationary measure $\mathcal{P}_0$. In this way, trajectories corresponding to long-lived fluctuations of the observable of interest are sampled with a higher probability. Furthermore, the importance ratio $\mathcal{P}_k/\mathcal{P}_0$ is computed by the algorithm. Indeed, the GKTL algorithm yields the SCGF, from which statistics with respect to $\mathcal{P}_0$ can be computed over a set of trajectories sampled from the biased measure $\mathcal{P}_k$. Finally, the knowledge of the SCGF in the vicinity of a given value $k = k^\star$ allows for the computation of the rate

function $I(a^\star)$, with $a^\star = \lambda'(k^\star)$. It describes the decay of the probability of rare fluctuations of the averaged observable of amplitude $a^\star$, as the average duration is increased, in the large-duration limit. The rate function can then be used to recover the PDF through the large deviation principle (4.15).

We stress here that the dynamics are *not* modified, as importance sampling is only achieved through the selection procedure. This makes the GKTL algorithm very general, as it is independent of the underlying dynamics. More specifically, there are *a priori* no restrictions to the application of the algorithm to turbulent dynamics. In the following section, as well as in chapter 5, we assess the practicality and efficiency of the GKTL algorithm for rare event sampling in turbulent flows. The study is based on test flow (2), presented in chapter 2. It is motivated by the fact that rare events in most turbulent flows are not accessible through a direct sampling approach, because of the large computational effort required to compute the dynamics. The GKTL algorithm therefore appears like a promising tool to perform computational studies of extreme events, simulating the dynamics of the flow over durations that are accessible to numerical simulations. In the following section, we discuss several practical and general aspects of the application of the GKTL algorithm to turbulence. To begin with, we explain that randomness must be artificially introduced in the dynamics of the copies and discuss several solutions. In a second part, we describe several implementation strategies, with a focus on parallel computing. The actual application of the GKTL algorithm to the test flows is presented in chapter 5. Several experiments are described and the corresponding results are analysed and discussed.

4.2 APPLICATION OF THE GKTL ALGORITHM TO A TURBULENT FLOW

The previous section described the GKTL algorithm in its general form, unregarding of implementation issues. In spite of the relative simplicity of the algorithm, its implementation for complex, deterministic dynamics deserves discussion.

The randomness of turbulent dynamics results from the strong variations of the macroscopic quantities describing the flow, as well as extreme sensibility to initial conditions. However, turbulent flows are described by the Navier-Stokes equations, which corresponding dynamics is deterministic. The deterministic nature of the dynamics is a hindrance to the application of rare events algorithms. For instance, in the GKTL algorithm, this entails that descendants of a given copy created at time $n\tau$ will follow the exact same path from $n\tau$ to $(n+1)\tau$, resulting in a poor sampling of trajectory space.

A workaround is to introduce small perturbations in the dynamics, exploiting the sensibility to initial conditions in order to separate trajectories. Naturally, this breaks one of the key feature of the GKTL algorithm: the fact that only statistics, and not dynamics, are modified. However, for highly chaotic systems for which sensibility to initial conditions is extreme, the amplitude of the perturbation can be chosen small enough so that it does not impact the statistics of the dynamics. Still, the perturbation will cause trajectories to separate over a finite time T_L, related to the *Lyapunov time* of the dynamics. This duration can be thought of as the time it takes for two trajectories starting on very close initial conditions to decorrelate. These aspects are illustrated and discussed in section 4.2.1.

The number of spatial mesh points needed to fully resolve a turbulent flow can be expected to grow as a power of the Reynolds number [51]. In practice, this implies that saving a state of the flow at a given time in memory comes at an important storage cost. The GKTL algorithm is based on the parallel simulation of a great number of flows, which means that at a given time, not only one state must be stored in memory, but N_c states. Being intrinsically parallel, the GKTL algorithm benefits from modern computer architectures. In particular, on computer clusters, the copies can be *distributed* across several computing nodes. Not only it allows for faster simulation—in terms of wallclock time)—but it also mitigates the memory load on one single node. However, the cloning stage entails communications between the nodes, involving transfers of a large amount of data from one to another. In section 4.3, we discuss the parallel implementation of the GKTL algorithm. In addition, the question of the impact of the communications on the overall performance of the algorithm is addressed.

4.2.1 *Perturbation of the trajectories*

For clones of a copy to separate over time, randomness must be introduced at the level of the dynamics. It can be achieved using a perturbation acting all along the dynamics, that is, adding a random component to the dynamics varying at each timestep of the numerical model. For instance, the inlet velocity in test flow (2) could be affected by a Brownian perturbation at all times. An alternative is to introduce an instantaneous perturbations after the cloning stage, perturbing the restart states of descendant clones.

The two options have been used in different previous works, see for instance [158] and [177]. On the one hand, adding a constant small noise to the dynamics provides a clearer theoretical framework. Indeed, the effect of noise can be studied considering the corresponding stochastic differential equation in the weak noise limit. Note that the

noise can also model an external perturbation with a physical reality, for instance a fluctuating upstream volumetric flow rate. However this approach results in a modification of the dynamics itself, and not only the initial conditions of the clones. In practice this can lead to difficulties, for instance if the dynamics is computed through external software which code cannot be modified to implement such a perturbation. In the second approach, the effect of the instantaneous perturbation introduced following the cloning stage is harder to analyse. However, we do not conduct a precise analysis of the effect of the perturbation in this work. As a result, we opt for the latter approach.

The state of the flow at time t is characterised by the pressure[2] field $\rho(\mathbf{x})$ and velocity field $\mathbf{u}(x)$. A natural way of perturbing such a state is to introduce a perturbation velocity field $\delta\mathbf{u}(\mathbf{x})$ and its corresponding density field $\delta\rho(\mathbf{x})$. Let us give an example: say copy j is cloned twice, yielding j_1 and j_2, following the cloning stage at iteration n. For the dynamics of j_1 and j_2 to separate over time, the state of one of the two clones, say j_2, is perturbed. More precisely, the initial state for iteration $n+1$ for copy j_2 becomes $\mathbf{X}_{j_2}(t = n\tau) = (\mathbf{u}^j + \delta\mathbf{u}, p^j + \delta p)$. Because the flow is turbulent, the perturbation is expected to result in an exponential divergence of the trajectories of j_1 and j_2 over time. More precisely, the trajectories for j_1 and j_2 are expected to decorrelate over a timescale τ_L like

$$||\mathbf{u}_1(t) - \mathbf{u}_2(t)|| \sim ||\delta\mathbf{u}||e^{t/\tau_L} \tag{4.31}$$

where

$$||\mathbf{u}||(t) = \left(\int |\mathbf{u}(\mathbf{x},t)|^2 d\mathbf{x}\right)^{1/2} \quad \text{with} \quad |\mathbf{u}(\mathbf{x},t)| = \sqrt{u_x^2 + u_y^2}$$

The case of the Lattice Boltzmann Method is somewhat special. Indeed, it differs from conventional CFD methods as it does not directly compute the dynamics of the macroscopic fields such as the velocity or pressure. Instead, it computes the dynamics of an ensemble of mesoscopic degrees of freedom—referred to as *populations* in section 2.1—from which macroscopic observables can be computed by simple averaging over the mesoscopic populations sitting on a given computational node. However, populations are not easily deduced from a given macroscopic state, for instance prescribed values for the initial velocity and pressure $(\mathbf{u}_0, p_0)$ [152]. For incompressible flows, a common practice is to initialise the populations at equilibrium:

$$f_i(\mathbf{x}, t = 0) = f^{eq}(\rho_0, \mathbf{u}_0(\mathbf{x})), \tag{4.32}$$

2 Within the LBM context, it is often more convenient to work with the density ρ. Recall that, following the weak compressibility approximation, the density is proportianal to the pressure through the ideal gas state equation $p = c_s\rho$. See appendix B for a description of the LBM.

with ρ_0 a uniform density field. However, the resulting pressure field is not consistent with the Navier-Stokes equations as it does not solves the Poisson equation corresponding to the initial velocity field $\mathbf{u}_0$ [110]. Furthermore, discarding the non-equilibrium part of the populations makes the initial populations inconsistent with derivatives of the velocity field, and especially strain rate tensors. As a consequence, the simulation suffers from initial layers [24]. Therefore, a common practice is to first simulate the flow over a transient regime, following initialisation, before any measurements are performed.

The issue of the initialisation of the LBM has been addressed in several works, following two different routes. The first approach is to solve the Poisson equation directly, and then approximate the non equilibrium part of the populations using a regularisation procedure [99, 152]. A more recent approach consists in an iterative scheme that both solves the Poisson equation and provides the initial values for the non-equilibrium populations [110]. This method was then enhanced in subsequent works [24, 83].

In the context of the GKTL algorithm, populations must be initialised on the perturbed macroscopic state $(\mathbf{u} + \delta\mathbf{u}, \rho + \delta\rho)$ at each iteration. In order to avoid the resolution of the Poisson equation for each clone, after each cloning stage, two alternative perturbation methods were explored:

1. A spatio-temporal perturbation of the forcing field over a short period of time following the initial condition.

2. The addition of a random perturbation on the restart state itself, however acting at the level of the mesoscopic Lattice Boltzmann populations, instead of the velocity and pressure fields.

4.2.1.1 *Perturbation of the forcing field*

If a perturbation cannot be applied directly on the state of the flow, one can act on its external parameters, such as the forcing field for test flow (1) or the inlet velocity for test flow (2). See chapter 2 for a presentation of test flow (1) and test flow (2). To illustrate this, we focus on test flow (1), in which a perturbation is added to the external forcing driving the flow downstream. In this context, the constant and homogeneous forcing field $F_0\mathbf{e}_x$ is replaced by

$$\mathbf{F}(\mathbf{x}) = F_0 \times (1 + \epsilon\eta(\mathbf{x}, t))\mathbf{e}_x \quad \text{with} \quad \eta(\mathbf{x}, t) = \eta_x(\mathbf{x})\eta_t(t),$$

where ϵ is a scaling parameter so that $\epsilon\eta(\mathbf{x}, t) \ll 1$ and $\mathbf{e}_x$ a unit vector aligned with the channel axis. In the following we denote by τ_p the timescale of the perturbation. Furthermore, we require that the perturbation occur on a timescale much shorter than the cloning period of the GKTL algorithm, *i.e.* $\tau_p \ll \tau$. As a consequence, we restrict the

choice of η so that $\eta_t(t) \underset{t \to \tau_p}{\to} 0$. Such perturbation is then characterised by the corresponding added momentum per unit volume $\epsilon \tau_p F_0$.

A first choice for η is to keep the homogeneous structure of the forcing, *i.e.* $\eta_x(\mathbf{x}) = 1$, while abruptly varying its intensity over a timescale τ_p. The simplest form for $\eta_t(t)$ is a step function with unit value over $[0, \tau_p]$. For the perturbation to be successful we also require that the timescale over which trajectories separate is much smaller than the cloning period τ.

However, we observed in numerical experiments that an abrupt increase of the forcing amplitude generates density waves propagating over the whole domain, as illustrated in figure 4.4. The greater the value of ϵ, the greater the amplitude of the waves. Unfortunately, we observed that the order of magnitude for the amplitude ϵ necessary for a rapid separation of the trajectories result in the emission of waves with a drastic effect on the drag force acting on the obstacle. Modulating the perturbation by a smoother function of time such a Gaussian pulse, keeping the added momentum constant, did not mitigate the amplitude of the density waves. It is also possible to give the perturbation η a spatial structure. For instance, $\eta_x(\mathbf{x})$ was set to be a realisation of a Gaussian random field with Gaussian covariance function $\langle \eta_x(\mathbf{x} + \mathbf{r}) \eta_x(\mathbf{x}) \rangle \propto \exp(-\mathbf{r}^2 / 2L_{corr}^2)$ where L_{corr} was set to match the typical size of the large scale eddies. Numerical experiments indicated that such forcing does not lead to a significantly faster separation oh the trajectories, and still requires high values for ϵ, causing density waves to alter the flow dynamics.

The practicality of the perturbation depends on two properties. First, it must occur on a timescale shorter than the cloning period, *i.e.* $\tau_p \ll \tau$. Second, the resulting separation of trajectories must take place over the timescale of the cloning period, *i.e.* $\tau_L \approx \tau$.

In the next chapter, we justify that a good choice of τ is $\tau \approx \tau_c$, with τ_c the *correlation time* of the drag acting on the obstacle, see section 5.1.2.1. The correlation time is the timescale over which the value of the drag decorrelates from its previous values. For a discussion of the correlation time, see chapter 3, page 49 Choosing τ of the order of the correlation time τ_c, we concluded that the design of the perturbation acting on the force density driving test flow (1) is very difficult, as it results in spurious density waves polluting the simulation. In addition, we made similar observations in the case of the perturbation of the inlet velocity for test flow (2). Note that this limitation originates from the intrinsic compressible nature of the Lattice Boltzmann Method, allowing density waves to propagate in the computational domain.

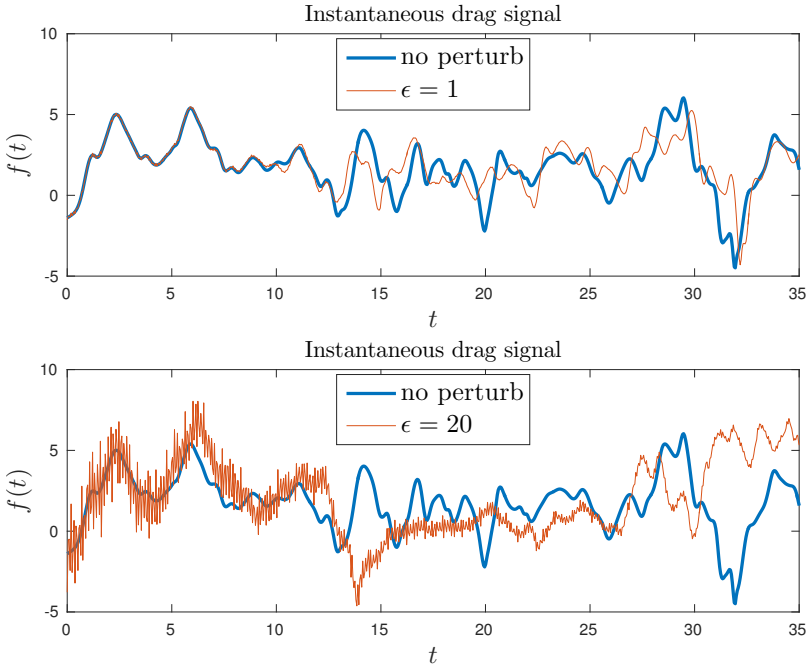


Figure 4.4: Drag timeseries illustrating the separation of two trajectories over time, subject to a small perturbation at time $t = 0$. The perturbation corresponds to abrupt increase of the forcing amplitude over $\tau_p = 0.1$. The top panel illustrates the case where the amplitude of the perturbation is limited so that no density waves are emitted due to the perturbation. The bottom panel pictures the propagation of density waves through the fluid domain as the amplitude of the perturbation is increased so that the two trajectories separate over a few correlation times. In this case the measurement of the drag strongly suffers from the emission of density waves.

4.2.1.2 *Perturbing a the level of the populations*

As stated in the introduction of section 4.2.1, it is difficult to initialise a LBM simulation on the basis of prescribed macroscopic fields such as the velocity or pressure. An alternative is to introduce the perturbation directly at the level of the mesoscopic populations manipulated by the LBM. See section 2.1 for a discussion of the LBM. In this section we address the design of such a perturbation. To start off with, we note that the Lattice Boltzmann Equation (B.1) is linear, up to nonlinear terms located in the collision operator. As explained in appendix C, under the LBGK approximation, these terms are of order one in the Mach number, $\mathcal{O}(\mathbf{u}/c_s)$. See section 2.1 and appendix B for a description of the Lattice Boltzmann Equation (LBE) and the LBGK approximation.

Such observation suggests that a linear combination of solutions of the lattice BGK equation can itself be considered as a solution, up to errors scaling like $\mathbf{u}/c_s$. In the following we denote by $\{f_i(\mathbf{x})\}_{1\leq i\leq 8}$ the set of populations on the lattice node $\mathbf{x}$. A perturbation is designed

as a random linear combination of solutions $\{f_i^{(n)}(\mathbf{x})\}_{1\leq n\leq N}$, precomputed before hand through a simulation of the flow in stationary regime. At each lattice node $\mathbf{x}$, populations are then replaced as

$$f_i(\mathbf{x}) \longrightarrow f_i(\mathbf{x}) + \epsilon \sum_{n=1}^{N} \alpha_n f_i^n(\mathbf{x}), \quad 1 \leq i \leq 9, \tag{4.33}$$

with the set $\{\alpha_n\}_{1\leq m\leq N}$ drawn uniformly from $[0,1]$ and ϵ the amplitude of the perturbation. The perturbed populations are then rescaled so that no mass is added in the system. For more details concerning this procedure, see appendix C.

Figure 4.5 illustrates the separation of two trajectories following a perturbation computed as (4.33). It displays the time evolution of the drag force corresponding to each trajectories. By contrast with a perturbation affecting the external forcing—see section 4.2.1.1—the amplitude of the emitted density waves is greatly reduced, for a similar separation timescale.

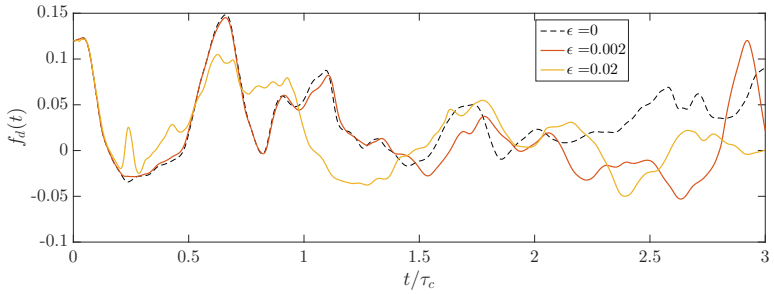


Figure 4.5: Separation of trajectories after the introduction of a small perturbation at $t = 0$, for two values of the amplitude of the perturbation ϵ. Each curve represent the drag timeseries corresponding to each trajectory. Each one of them results from the integration of the dynamics from the same initial condition. The dashed timeseries denoted by $\epsilon = 0$ was obtained by integrating the dynamics without introducing any perturbation. The perturbation is introduced at the level of the mesoscopic populations involved in the LBM simulation of the flow, according to equation (4.33). By contrast with figure 4.4, the drag force is not polluted by spurious density waves propagating in the domain.

4.3 IMPLEMENTATION OF THE GKTL ALGORITHM FOR TURBULENT FLOWS

In the previous section we described an important aspect of the application of the GKTL algorithm to turbulent flows. Indeed, randomness must be artificially introduced in the dynamics as the trajectories are cloned, so that copies can separate over time. In this section we turn to practical aspects concerning the implementation of the GKTL algorithm for complex dynamics such as turbulent flows. More pre-

cisely, we discuss the interplay between the simulation of the flow on the one hand, and the sampling algorithm on the other hand. For simple, one dimensional dynamics such as the OU process discussed in section 4.1.4, implementation of the GKTL algorithm is straightforward. The state for each copy, *i.e* one real number, can be stored in memory at all times, for instance in an array which values are exchanged following the cloning stage. During iteration n, copies can simply be simulated from $t = (n-1)\tau$ to $n\tau$ in a sequence.

Note that the evolution of a copy over $[(i-1)\tau; \tau]$ is independent from the evolution of the other copies. Therefore, there is no reason to perform the corresponding computations in a sequence: they could very well be carried out in parallel. For simple systems such as the OU process, using hundreds of copies, parallelization of the evolution stages is certainly not worth the additional implementation effort. However, the situation is drastically different for complex dynamics such as turbulent flows, for which the computation of the dynamics involves a lot more computations. In this case, parallelisation of the algorithm is required for a practical use of the algorithm.

An attractive property of the GKTL algorithm is its *embarrassingly parallel* nature. An algorithm is said to be embarrassingly parallel when it can easily be decomposed into parallel tasks, with very little or no coordination needed between them. This class of algorithms greatly benefit from modern computer architectures, that are able to run many concurrent independent tasks at the same time. At a larger scale, such algorithms are a great fit for computer clusters, *i.e.* aggregates of interconnected computers, or *nodes*. Such architectures usually allow programs to perform faster, due to the high level of parallelism available. However, in general, nodes do not share the same pool of memory, and if data produced on a given node is needed on another, it must travel through the links of the network, therefore resulting in an overhead time. On the one hand, no matter how efficient it is in theory, an algorithm that requires frequent communications between distant nodes may perform very badly on a computer cluster, its efficiency decreasing with the number of involved computing nodes. Such algorithm is said to have poor *scalability*. On the other hand, because they are composed of independent tasks, embarrassingly parallel offer very good scalability. In the GKTL algorithm, coordination is required between the different copies during the cloning stage. It will thus offer very good scalability if the cost of simulating the dynamics during the evolution stage greatly exceeds the overhead entailed by the cloning stage. It is expected for turbulent flows, as the simulation of the dynamics is computationally expensive.

In this section we discuss several parallel implementations of the GKTL algorithm. On the basis of the test flows presented in section 2, we illustrate that the overall time spent on communications between nodes in the cloning stage account for less that 5% of the time required

to simulated the dynamics over the evolution stage. In the following we start by discussing an implementation in which the computation of the dynamics and the cloning stage are performed by two different programs. Then we present a *message-passing* implementation in which nodes communicate their state to each other.

4.3.1 *Separated implementation*

The GKTL algorithm can be decomposed into two independent steps. During the iteration i, the copies first evolve from $(n-1)\tau$ to $n\tau$: this is referred to as the *evolution stage*. Next, each copy is attributed a weight which value depends on the evolution of *all* the copies. Then, copies are either cloned or discarded from the ensemble depending on the value of their relative weight (4.10). Finally, the final state of the discarded copies for $t = i\tau$ is erased and replaced by a the final state of a cloned copy. These last three points are grouped into one single step: the *cloning stage*. See section 4.1.2.1 for a description of the algorithm. The evolution stage can easily be performed in parallel, as the simulation of the dynamics of a copy does not require information from other copies. By contrast, the cloning stage is a *global* operation. In order to compute the weight of a copy, information must be gathered from all the copies.

A way to implement the GKTL algorithm is to separate the two stages into two different programs. This allows for a straightforward parallelisation as several independent instances of the program computing the dynamics can be run in parallel on different nodes. This program must write the final state of the copy on disk, as well as the corresponding weight w_j, where j denotes the index of the copy, see page 89. When all the instances are done, another program reads in the weights and compute the relative weights (4.10) for each copy. It then computes the number of clones for each copy and assigns the different final states at iteration i to the corresponding copies. They will then serve as initial conditions for iteration $i+1$. In practice a third program can be used to effectively perform the algorithm, that is scheduling the execution of the cloning and execution stages.

This implementation is the most straightforward way of performing the GKTL algorithm when the dynamics are computed by external software. By external software we denote a program which code is not accessible or too complex to be embedded in a unified GKTL code. For instance, this would be the case for applications of the GKTL algorithm with industrial CFD packages such as ANSYS Fluent [4], openFOAM [86] or proLB [79]. The main advantage of this implementation is its simplicity. However, it leads to reading and writing large amount of data on disk. We stress that file input/output are relatively slow operations with respect to accessing data in memory. However, for highly turbulent flows in complex geometries, the cost

of performing the simulation of the copies over the evolution stage is expected to always greatly exceed the cost of the cloning stage.

For simpler systems, such as the test flows involved in this works—see chapter 2— we describe in the next section an alternative implementation. In this implementation the GKTL algorithm is performed by a single program in which simulation of copies is performed by processes communicating with each other through *message-passing*.

4.3.2 *Message-Passing implementation*

In this thesis we apply the GKTL algorithm on the simple test flows presented in chapter 2. As mentioned in chapter 2 and detailed in appendix A, the simulation of the flow is accessible through simple library calls within a C++ program. As a consequence, it can easily be embedded in a unified code that perform the GKTL algorithm. This implementation makes use of *distributed parallelism*. In the following, it is assumed that the N_c copies of the flow are evenly distributed across a set of $\{P_i\}_{0\leq i\leq N_p-1}$ processes, that reside on separate computing nodes, with separate memory. Note that the current state of each copy is stored in memory. This is made possible by the two-dimensional geometry of the test flows, as well as their relatively coarse mesh. More precisely, the state of a copy corresponds to the set of lattice Boltzmann populations $\{f_i\}_{0\leq i\leq 8}$ at each lattice nodes. For instance, for test flow (2), it represents $513 \times 129 \times 9$ reals numbers. In double precision, this amounts to roughly $4MB$ of data. We stress that the state of the N_c copies do not reside in the same pool of memory, but instead are dispatched among the N_p nodes. On the HPC facilities involved in this work [128] (Tier-2 facility), the typical available memory per process is 15GB. Therefore, the upper limit for the number copies N_c does not originate from memory limitations, but rather from computing time.

Because each copy resides in a separate memory pool, processes exchange information during the cloning stage through *message-passing*. Message-passing parallelism introduces a programming model in which computing nodes are viewed as *processes*, and functions are used to explicitly transmit data to one process to another. Message passing parallelism was developed by the Message-Passing Interface (MPI) Forum in 1993 and is without contest the most widely used method for writing distributed parallel programs. Note that MPI is not a programming language, but rather a standard describing a set of functions that can serve as building blocks for parallel programs. Many implementations of MPI are available. This works makes use of the open source library openMPI [52].

4.3.2.1 *Centralised implementation*

In practice, a process computes the dynamics of N_c/N_p copies in a sequence, and the corresponding node hosts the corresponding states

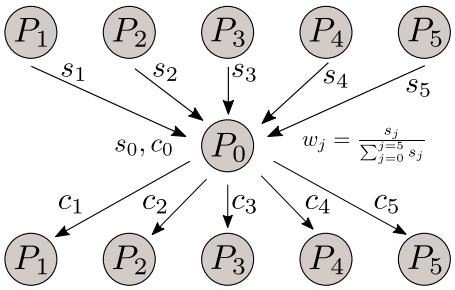


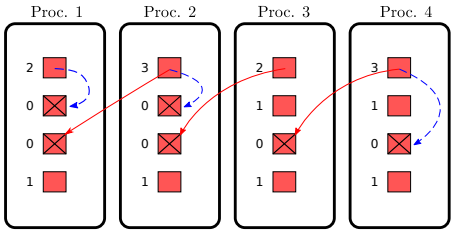
Figure 4.6: Parallel, centralised implementation of the GKTL algorithm. In this example we assume that each one of the processes $\{P_i\}_{0\leq i\leq N_p-1}$ hosts one copy, *i.e.* $N_p = N_c$. The evolution of the copies is computed locally within each process, as well as the corresponding weight w_j, see page 89. Then, the master process P_0 gathers the weights and compute the relative weight W_j (4.10) for each copy j. On the basis of these weights, the master process P_0 computes the number of clones for each copy and decides of the communications between the processes in order to distributes the states of cloned copies to discarded ones. Finally, it sends the corresponding information, denoted by $\{c_j\}_{0\leq j\leq 5}$, to the other processes $\{P_i\}_{1\leq i\leq N_p-1}$. Note that the master process is a process like any other, *i.e.*, it hosts a copy and participates in the communications.

in memory. As mentioned above, the cloning stage is a *global* operation. To compute the relative weights (4.10), information must be gathered from all the processes. We therefore define a *master process*, denoted by P_0, in which the weights of each copies will be centralised. As a result, P_0 is in charge of computing the relative weights of each copies. It then compute the number of clones for each copy and then performs the cloning. Finally, it elaborates a *communication plan*, which describes the communication of states across the processes. For instance, a process hosting a discarded copy is ordered to receive a state from a process hosting a cloned copy. By contrast, a process hosting a cloned copy is ordered to send the corresponding state to a process hosting a discarded copy. This communication plan is then communicated to all the processes $\{P_i\}_{1\leq i\leq N_p-1}$, as illustrated in figure 4.6.

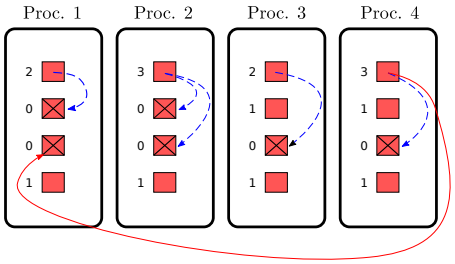
In practice, for each duplicate of a cloned copy, the master process P_0 cycles through the processes $\{P_i\}_{1\leq i\leq N_p-1}$, looking for a discarded copy to replace. Note that the amount of communications between distant nodes can be reduced by preferentially distributing states of cloned copies among copies within the same process. This is illustrated in figure 4.7.

4.3.2.2 *Decentralised implementation*

In a very recent work [21], an alternative to the above *centralised implementation* is proposed. It was published shortly after the above implementation was designed. This implementation does not involve a master process. The relative weight W_j for each copy j is computed in



(a) Basic redistribution of cloned copies



(b) Preferential redistribution within the process

Figure 4.7: Schematic representation of two communication patterns between processes during the cloning stage. In this example, each four processes host four copies of the flow, represented by red boxes. The number of clones produced by each copy is indicated to the left side of the box. Ticked boxes indicate copies discarded from the ensemble. Processes holding copies producing more than one clone must redistribute the clones among processes. Data transfers between different process are represented by a continuous red arrow, while data transfers involving copies within the same process are pictured by dashed blue arrows. In the second implementation (fig. 4.7b), communications are planned so that clones are redistributed among the same process preferentially, thus greatly reduced to amount of data transfer between different processes.

parallel by each process through a collective communication, referred to as `MPI_AllReduce`. Furthermore, the cloning stage is implemented in a different way than the one presented in section 4.1.2.1, which allows each individual processes to determine its own communications. The index of the required clones are then exchanged among process through a second collective communication of type `MPI_AllGather`. An important reduction of the number of communications is achieved by packing every communications from process A to process B into one single message, and to plan communications ensuring that they always go one way: either from process A to process B or from B to A.

The optimisation of the cloning stage in [21] is motivated by the fact that the simulation of the underlying dynamics requires relatively low computational effort. In this case, the computational cost of performing the communications during the cloning stage may not be negligible with respect to the cost of simulating the dynamics over the evolution stage. As a result, a reduction of the number of communications can provide a significant increase in performance. The situation is different with the simulation of turbulent flows, for which the computing time required to simulate the dynamics over the evolution time greatly exceed the computing time associated with the cloning stage. Therefore, we do not expect a significant increase in performance from the reduction of the communications. In the following we illustrate this remark on the basis of test flow (2). We show that the overall computational effort for the cloning stage represents less than 5% of the overall computational effort for the evolution stage.

4.3.3 *Average walltime for the evolution and cloning stages*

In this section we compare the computational cost of the dynamics of evolution stage with the computational cost of the cloning stage. Moreover, the cloning stage is decomposed into:

- The transfer of the individual weights w_j from the processes $\{P_i\}_{1\leq i\leq N_p-1}$ to P_0, the calculation of the relative weights W_j as well as the elaboration of the communication plan, as well as its broadcast to the processes $\{P_i\}_{1\leq i\leq N_p-1}$
- The ensemble of point-to-point communications through which processes send and receive states.

To illustrate the relative weight of both the evolution and cloning stages, we perform three GKTL experiments with three different values of the bias k. These experiments are based on test flow (2), described in chapter 2. The number of copies is set to $N_c = 128$. The cloning period is $\tau = \tau_c$, with τ_c the typical correlation time of the drag acting on the obstacle. The evolution/cloning sequence is repeated $M = 40$ times, leading to an total length of the trajectories $T_a = 40\tau_c$. Note that

the algorithm terminates by a cloning step. In practice the number of processes N_p is chosen close to the number of copies N_c. The reason for that is the high computational cost of the evolution stage. Therefore, piling up several copies into one process leads to large computing times. In these experiments we set $N_p = N_c = 128$.

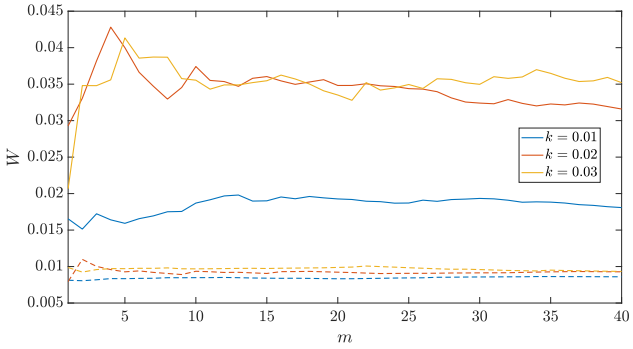
The computational effort for each step is quantified through its average elapsed real time, referred to as *walltime* in the following. The average is computed over the N_c copies. For instance, the average walltime for the evolution stage is:

$$\overline{w_{evo}(n)} = \frac{1}{N_c} \sum_{j=1}^{N_c} w_{evo}^{(j)}(n) \tag{4.34}$$

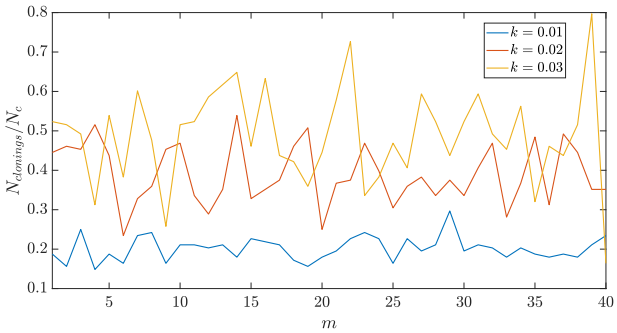
where $w_{evo}^{(j)}(n)$ denotes the measured walltime for the simulation of the dynamics of copy j over the cloning period τ at iteration n. Similarly, we denote by $\overline{w_{cl}}(n)$ the average walltime for the computations performed by the master process during the cloning stage. Finally, $\overline{w_{comms}}(n)$ denote the average walltime for the communications in which states are distributed across the N_p processes. Figure 4.8a illustrates the ratios $\overline{w_{comms}}(n)/\overline{w_{evo}}(n)$ and $\overline{w_{cl}}(n)/\overline{w_{evo}}(n)$ as a function of the iteration n for three different values of the bias k. Additionally, figure 4.8b displays the percentage of discarded copies that is discarded along the iterations of the algorithm. It illustrates that, as k is increased, more copies are discarded. It therefore results in a larger number of communications. As a result, figure 4.8a illustrates that the cost of the communication stage increases with k. Figure 4.8a shows that the walltime for the communications amounts for less than 5% of the walltime for the simulation of the dynamics over the cloning period τ.

In a way of conclusion, the optimisation of the cloning stage cannot be expected to yield significant performance gains. For complex systems, the bottleneck resides in the simulation of the dynamics over the cloning period. Note that the impact of the computations and communications associated with the cloning stage increases as the cloning period τ is decreased. However, the cloning period should not be chosen much shorter than the correlation time of the underlying dynamics. This is justified in chapter 5.

In the following the GKTL algorithm is implemented following the *message-passing centralised implementation* discussed in section 4.3.2. The simulation of the dynamics is encapsulated in the `pipeLBM` C++ library developed for this thesis, described in appendix A. Finally, the perturbation of the copies following the cloning stage is implemented at the level of the Lattice Boltzmann populations, as discussed in section 4.2.1.2 and appendix C. In the following chapter we present the application of the GKTL algorithm to the sampling of extreme drag



(a) (Continuous) Ratio of the average walltime for the communication following the cloning stage and the average walltime for the evolution stage. (Dashed) Ratio of the average walltime for the cloning and the average walltime for the evolution stage



(b) Number of clonings normalised by the total number of copies

Figure 4.8: **(a)** Comparison of the average walltime for the communications between processes following the cloning stage (continuous) and cloning stage (dashed) with the average walltime for the evolution stage. The average walltimes are computed over the $N_c = 128$ copies and represented as a function of the index m of the $M = 40$ iterations of the evolution/cloning procedure. The continuous lines represent $W \equiv \overline{w_{comm}}(n)/\overline{w_{evo}}(n)$ where $\overline{w_{comm}}$ denotes the average measured walltime for the point-to-point communications in which states are received and sent across the processes. This walltime is normalised by the average time of the evolution time $\overline{w_{evo}}$. The dashed lines represent $W \equiv \overline{w_{cl}}(n)/\overline{w_{evo}}(n)$ where $\overline{w_{cl}}$ denotes the average walltime for the computations performed by the master process for the cloning stage. **(b)** Number of created clones, or equivalently, number of discarded copies $N_{clonings}$, as a function of the iterations of the algorithm. One can see that, as k is increased, a larger part of the population is discarded. As a result, a larger number of communications is observed.

fluctuations in turbulent flows. In section 4.1.4, the algorithm was shown to significantly reduce the computational cost of the sampling of extreme fluctuations for the OU process However, it is not clear if similar gains can be achieved on the basis of turbulent flows.

5

IMPORTANCE SAMPLING LARGE DRAG FLUCTUATIONS WITH THE GIARDINA–KURCHAN–TAILLEUR–LECOMTE ALGORITHM

Chapter 4 introduced importance sampling, a widely used statistical method to sample rare events at a much lower computational cost than brute force sampling [22]. The underlying idea is to generate realisations of the process according to a biased PDF, from which extreme events are sampled with a higher probability—importance—than from the original PDF of the process. See figure 4.1 on page 86 for an illustration. The construction of the biased distribution depends on the application. When the PDF of the process of interest verifies a large deviation principle, a common method for designing the biased distribution is to perform an *exponential change of measure* [166]. In the case of the time-averaged drag, it reads:

$$\mathcal{P}_k\left(\{\mathbf{x}(t)\}_{0\leq t\leq T_a}\right) = \frac{1}{Z(k,T_a)}\exp\left(k\int_0^{T_a} f_d[\mathbf{x}(t)]\mathrm{d}t\right)\mathcal{P}_0\left(\{\mathbf{x}(t)\}_{0\leq t\leq T_a}\right) \tag{5.1}$$

where f_d denotes the instantaneous drag, and T_a the duration of the average. In addition, $\{\mathbf{X}(t)\}_{0\leq t\leq T_a}$ is a formal notation for a trajectory of the flow, in phase space. The distribution $\mathcal{P}_0\left(\{\mathbf{X}(t)\}_{0\leq t\leq T_a}\right)$ is the stationary PDF of the dynamics, or stationary measure, and $\mathcal{P}_k\left(\{\mathbf{X}(t)\}_{0\leq t\leq T_a}\right)$ is the biased distribution resulting from the exponential change of measure with parameter k. Finally, $Z(k,T_a)$ is a normalisation constant ensuring that $\mathcal{P}_k\left(\{\mathbf{X}(t)\}_{0\leq t\leq T_a}\right)$ is a normalised PDF. From the biased PDF $\mathcal{P}_k$, high values of $\int_0^{T_a} f_d[\mathbf{X}(t)]\mathrm{d}t$ are more likely to occur than from the original PDF $\mathcal{P}_0$. Because of the large deviation behaviour of the PDF of $\frac{1}{T_a}\int_0^{T_a} f_d[\mathbf{X}(t)]\mathrm{d}t$ described in the previous chapter, importance sampling based on (5.1) can be shown to be optimal in the limit $T_a \to \infty$. Furthermore, in this limit, $Z(k,T_a) \underset{T_a\to\infty}{\to} e^{T_a\lambda(k)}$ with $\lambda(k)$ the Scaled Cumulant Generating Function (SCGF) defined by $\lambda(k) = \lim_{T_a\to\infty}\frac{1}{T}\ln\mathbb{E}_{\mathcal{P}_0}[\exp(k\int_0^{T_a} f_d[\mathbf{X}(t)]\mathrm{d}t)]$.

The GKTL algorithm described in chapter 4 is a numerical method to sample realisations of a dynamical process according to the biased measure $\mathcal{P}_k$. It implements importance sampling at the level of trajectories, based on the simulation of an ensemble of copies of the system. The algorithm was successfully applied for relatively simple chaotic dynamics [57, 94, 158], involving only a few degrees of freedom. Re-

cently, it has been successfully applied to the computation of heat waves using a simplified climate model [130].

This chapter describes the application of the GKTL algorithm to the importance sampling of dynamical trajectories in the DNS of a 2D turbulent flow past an obstacle. In general, numerical simulations of fluid flows comes at a great computational cost, especially in industrial or geological contexts. In this case, direct sampling of rare events by means of the simulation of the flow over very long times is simply unthinkable. Therefore, numerical simulations are limited to the average behaviour of the system, as well as typical fluctuations. A successful application of the GKTL algorithm to the simulation of extreme events in turbulent flows could therefore pave the way to a whole new range of computational studies, providing numerical estimates for the statistics of extreme fluctuations as well as simulations of the flow patterns corresponding to such rare events. Such information would certainly prove useful in many applied domains, such as wind energy, car aerodynamics, or civil engineering. From a fundamental point of view, it could provide a deeper insight into the physics of turbulent flows, of which little is understood.

However, the practicality of rare event algorithms for actual complex systems is still an open question. The objective of this chapter is to establish of *proof of principle* for the application of the GKTL algorithm to turbulent flows. In this chapter we show that the GKTL algorithm is able to provide a reliable estimate for the tails of the PDF of the averaged drag acting on a an obstacle in a two-dimensional turbulent flow, in the limit where the average is taken over long durations. We also show that it allows for an improved sampling of extreme fluctuations of the averaged drag.

In the following we use the GKTL algorithm to generate trajectories of duration T_a according to the biased PDF (5.1). We are interested in sampling extreme fluctuations for the averaged drag over a duration $T \leq T_a$, computed as a moving average over the trajectories:

$$F_T(t) = \frac{1}{T}\int_{t-T}^{T} f_d(t)\mathrm{d}t \text{ with } T \leq t \leq T_a \tag{5.2}$$

where f_d denotes the instantaneous drag over the trajectories sampled by the algorithm.

The practicality of a rare event algorithm can be assessed based on how efficiently it samples extremes, and how accurately it computes their probability with respect to crude direct sampling. For the comparison to be fair, both must be compared on the basis of their *computational cost*, measured in terms of the overall duration over which the dynamics have been computed. A direct simulation of the flow over a duration T has a computational cost of T, and a GKTL run

involving the simulation of N_c copies over a duration T_a has a cost of $N_c \times T_a$.

The structure of this chapter is as follows. First, section 5.1 shows that the GKTL algorithm is able to efficiently compute the large deviation rate function for the time-averaged drag. The rate function describes the rate of decay of the probability of fluctuations of the averaged drag over long durations. As a result, we provide an estimate of the tails of the PDF describing the statistics of the fluctuations of the averaged drag. First, Section 5.1.1 describes the construction of a reference estimate for the rate function, based on a timeseries of finite duration. Secondly, section 5.1.2 discusses the parameters of the GKTL algorithm and illustrates that the rate function can be computed with the GKTL algorithm, involving a much lower computational cost than what would be required by a direct sampling approach.

In addition to the probability of rare events, the GKTL algorithm gives access to the corresponding dynamics. Section 5.2.2 describes how the GKTL can be used to study the flow dynamics leading to extreme drag fluctuations. As a matter of fact, the algorithm leads to an ensemble of discontinuous paths, resulting from copies being discarded and replaced by clones of others. Section 5.2.2 gives a technical account on how the continuous paths—those effectively distributed as $\mathcal{P}_k$—can be obtained. This can be done either on-the-fly over the course of the algorithm, or as a post processing step. The construction of the continuous paths is actually essential to compute their probability, as shown in section 5.2.4 Finally, section 5.2.5 illustrates that the GKTL is capable of sampling extremes of the averaged drag at a much reduced computational cost.

5.1 EFFICIENT COMPUTATION OF THE LARGE DEVIATION RATE FUNCTION

The GKTL algorithm has originally been designed to compute large deviations rate functions. Indeed, it outputs the SCGF, from which the rate function can be computed through a Legendre transform. See chapter 1 and section 4.1.3, as well as references therein for a discussion of large deviation rate functions. To this date, the GKTL algorithm have been mostly applied to rather simplified dynamics. One of the objectives of this work is to assess its relevance for turbulent flows. In this section, we compute the large deviation rate function describing the exponential decay of the probability of extreme fluctuations of the averaged drag. More importantly, we show that, with the help of the GKTL algorithm, we are able to compute the statistics of rare events at a much reduced computational cost, compared with direct sampling.

In order to validate the computation, a reference estimate of the SCGF is first computed on the basis of a drag timeseries obtained

from a brute force simulation of the dynamics over a duration T_{tot}. This computation is described in section 5.1.1. A second estimate of the SCGF is then computed using the GKTL algorithm *with a lower computational cost*. It is finally compared to a third estimate, directly computed from a timeseries involving the same computational cost as the latter estimate. It is shown in section 5.1.2.2 that the estimate from the algorithm correctly recovers the reference solution, while the direct estimate fails.

5.1.1 *Direct estimation of the rate function on the basis of a finite timeseries*

In the following we describe the construction of a reference estimate for the SCGF, describing the large deviation statistics for the time-averaged drag. This work is based on the methodology proposed by Rohwer *et. al* in [137].

Let $\{f_d(t)\}_{0\leq t\leq T}$ be a drag timeseries over a duration T. Recall the definition of the SCGF:

$$\lambda(k) = \lim_{T\to\infty} \frac{1}{T} \ln \mathbb{E}[e^{k\int_0^T f_d(t)\mathrm{d}t}] \tag{5.3}$$

An estimator for (5.3) can be constructed using a *block-averaging* method. In order to evaluate the ensemble average in (5.3), the time-series is split into blocks of equal length ΔT and the average is computed as an empirical mean over the blocks. The length of the blocks ΔT is chosen long enough so that blocks can be considered statistically independent. The resulting estimator for the SCGF is

$$\hat{\lambda}_{\Delta T}(k) = \frac{1}{T} \ln \frac{1}{M} \sum_{m=1}^{M} \exp\left(k \int_{(m-1)\Delta T}^{m\Delta T} f_d(t)\mathrm{d}t\right) \ , \ M = \frac{T}{\Delta T} \tag{5.4}$$

The SCGF being defined in the limit where the integration window goes to infinity, *e.g.* $T \to \infty$ in (5.3), the length ΔT must be taken much larger than the typical correlation time of the timeseries, that is : $\Delta T \gg \tau_c$. However, the larger ΔT, the fewer terms are involved in the empirical mean in (5.4), resulting in a poor estimate.

Additionally, for a fixed ΔT, the domain of validity of the estimate (5.4) is bounded. Indeed, as $|k|$ is increased, the estimate suffers from the so-called *linearization effect* [137], that originates from the finiteness of the timeseries. Because $\hat{\lambda}_{\Delta T}$ consists in a sum of exponentials, as k is increased it is dominated by the largest term in the sum. More precisely,

$$\hat{\lambda}_{\Delta T}(k) \underset{k\gg 1}{\sim} kF_{max} \ \text{ with } \ F_{max} = \max_{1\leq m\leq M} \{\int_{(m-1)\Delta T}^{m\Delta T} f_d(t)\mathrm{d}t\} \tag{5.5}$$

There is thus a threshold $k_c^{(+)}$ past which the estimate $\hat{\lambda}$ cannot be trusted. Similarly, as k decreases away from 0, the sum is dominated by

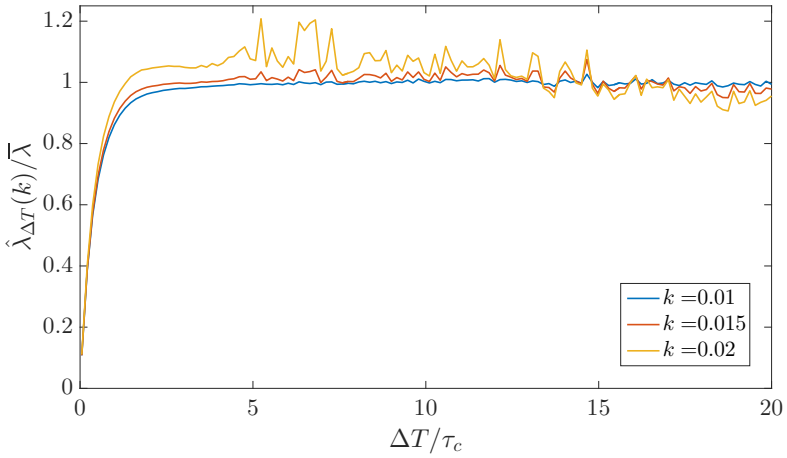


Figure 5.1: Convergence of the estimator of the SCGF defined in (5.4) as the block length ΔT is increased, for three different values of k. Each estimate is normalised by its average value over the range $\Delta T \in [10\tau_c; 20\tau_c]$, denoted as $\overline{\lambda}$. Convergence is achieved as the block length approaches the large-time limit involved in the definition of the SCGF. As ΔT is increased, the number of blocks M decreases due to the finiteness of the timeseries. As a result, fewer terms contribute to the empirical average in (5.4) and the corresponding statistical error increases. In the following we choose $\Delta T^\star = 6\tau_c$ as the value for which the estimate is converged, up to relative errors of the order of 5%.

the term involving the minimum over the timeseries. As a result, there is also a negative threshold $k_c^{(-)}$ past which the estimate $\hat{\lambda}$ cannot be trusted. In the following, we only focus on positive fluctuations and therefore only refer to $k_c^{(+)}$ as k_c. Moreover, it was shown in [137] that the empirical sum involved in (5.4) is normally distributed around its expectation value *only* for $0 \leq k \leq k_c/2$. Over this range, error bars can therefore be computed on the basis of the standard deviation computed over the set $\{\exp\left(k\int_{(m-1)\Delta T}^{m\Delta T} f_d(t)\right)\}_{1\leq m\leq M}$. For $k_c/2 \leq k \leq k_c$, even though $\hat{\lambda}(k)$ converges towards $\lambda(k)$, this convergence is non-Gaussian and error bars must be computed on the basis of independent repetitions of the estimate, thus requiring much more data.

To find a suitable block length one must therefore operate a trade-off between a large value of ΔT that guarantees the large-time limit $T \to \infty$ in (5.3) and a sufficient number of blocks that mitigates the linearization effect. An important issue is thus the estimation of the minimum block length $\Delta T^\star$ for which one can consider the limit $\lambda(k,T) \underset{T\to\infty}{\to} \lambda(k)$ as attained, up to a given a error tolerance.

Figure 5.1 illustrates the convergence of $\hat{\lambda}_{\Delta T}$ as the block length is increased. The minimal block length $\Delta T^\star$ is chosen so that the estimates are converged up to relative errors of 5%, leading to $\Delta T^\star \approx$

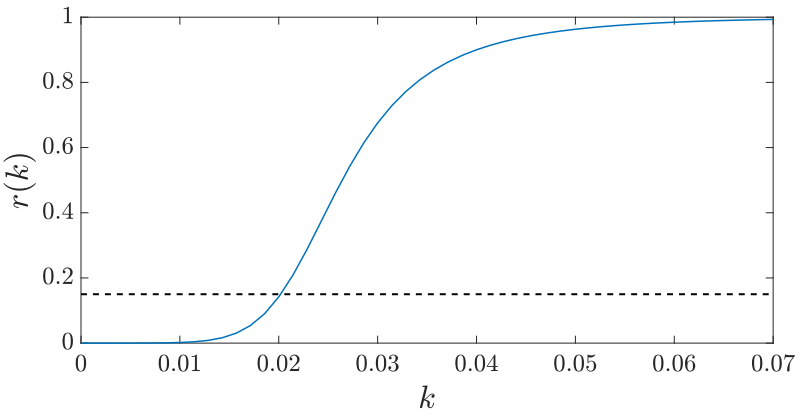


Figure 5.2: Contribution of the largest term in the sum in (5.4) for blocks of length $\Delta T^\star = 6\tau_c$, as a function of k. Because it is a sum of exponentials, the largest terms quickly dominates, resulting in artificial linear tails for the SCGF. In this work, we consider the estimate $\hat{\lambda}$ valid as long as $r(k) < 15\%$, that is, for $k \lesssim 0.02$.

$6\tau_c$. This criterion is arbitrary. On the one hand, it is strict enough to ensure satisfactory convergence of the estimate. On the the other hand, it is lax enough so that a statistically significant number of blocks is involved in the sum in (5.4). This criterion is actually validated *a posteriori* by the computation of the SCGF from the GKTL algorithm.

With the block length fixed, the region of validity of the estimate $\hat{\lambda}(k)$ can now be assessed. That is the critical value k_c corresponding to the inset of the linearization effect must be estimated. To do so we quantify the contribution of the largest term in the sum in (5.4). We therefore define the ratio $r(k)$ as follows:

$$r(k) = \frac{e^{kF_{max}}}{\sum_{m=1}^{M} e^{k\int_{(m-1)\Delta T}^{m\Delta T} f_d(t)\mathrm{d}t}} \quad \text{with} \quad F_{max} = \max_{1\leq m\leq M}\{\int_{(m-1)\Delta T}^{m\Delta T} f_d(t)\mathrm{d}t\}. \tag{5.6}$$

Figure 5.2 displays the evolution of $r(k)$ as k is increased. It illustrates that the contribution of the largest term in the sum of exponentials in (5.4) progressively increases as k in increased. As a result, it gives rise to an artificially linear estimate of the SCGF (5.5). The threshold k_c then is chosen as the value of k past which $r(k)$ is above 15%, leading to $k_c \approx 0.02$. Again, this criterion is arbitrary and justified *a posteriori* through the comparison of the direct estimate with respect to the estimate computed from the GKTL algorithm, see section 5.1.2. Note that the value of k_c depends on the total duration of the time-series T_{tot}: as T_{tot} increases, so does k_c. Furthermore, we stress that the value of k_c is not strictly fixed. Indeed, it must only be interpreted as a rough estimate of the upper bound of the range over which the estimate (5.4) can be trusted.

Eventually, figure 5.3a displays the direct estimate of the SCGF based on equation (5.4). The corresponding timeseries spans $T = 10^6\tau_c$ and $\Delta T = \Delta T^\star = 6\tau_c$. Additionally, figure 5.3a displays the estimate of the SCGF, assuming that the PDF of the averaged drag is Gaussian. It is defined as [160]:

$$\lambda^{(g)}(k) = 2\hat{\tau}k^2 \text{ with } \hat{\tau} = \lim_{t\to\infty} \frac{1}{\sigma^2} \int_0^t \left(\mathbb{E}[f_d(\tilde{t})f_d(\tilde{t}+\tau)] - E[f_d]^2\right) \mathrm{d}\tilde{t} \tag{5.7}$$

where σ denotes the standard deviation of the drag f_d. Figure 5.3b displays the corresponding rate functions, computed using the Gartner-Ellis theorem (4.21).

Figure 5.3 shows that, for small fluctuations, the statistics of the averaged drag F_T can be considered Gaussian. However, for $a \geq 0.02$, the departure from Gaussian statistics is clear. More precisely, figure 5.3a illustrates that the Gaussian estimate of the SCGF is below the direct estimate. As explained in note 1, this entails that a given fluctuation of the averaged drag $F_T \geq a$ is more probable than if the statistics were Gaussian: the PDF of F_T has *fat tails*.

The direct estimate shown in figure 5.3a will now serve as a reference solution for the estimation of $\lambda(k)$ using the GKTL algorithm.

Note 1. The PDF of the averaged drag has fat tails
In the large time limit, the large deviation principle writes:

$$\mathcal{P}(F_T = a) \underset{T\to\infty}{\asymp} e^{-TI(a)}. \tag{5.8}$$

The rate function can be written as a Legendre-Fenchel transform of the SCGF:

$$\begin{cases} I(a) = k_a a - \lambda(k_a) \\ \lambda'(k_a) = a. \end{cases} \tag{5.9}$$

As a consequence, the tangent to the SCGF in $k = k_a$ is defined by

$$y_a(k) = ak - I(a), \tag{5.10}$$

and its intersection with the Y-axis gives the value of the rate function $I(a)$. Therefore, the direct estimate of the rate function, $\hat{I}_{\Delta T}$, takes values below the Gaussian estimate $I^{(g)}(a)$. This can be seen in figure 5.3b. Following the large deviation principle (5.8), the probability that the averaged drag F_T takes a given value a is underestimated by the Gaussian approximation.

5.1.2 *Estimation of the SCGF using the GKTL algorithm*

In the previous section, we computed the SCGF out of a very long timeseries of the drag. We showed that the resulting estimate suffers from the finite size of the timeseries, and that its validity is limited to

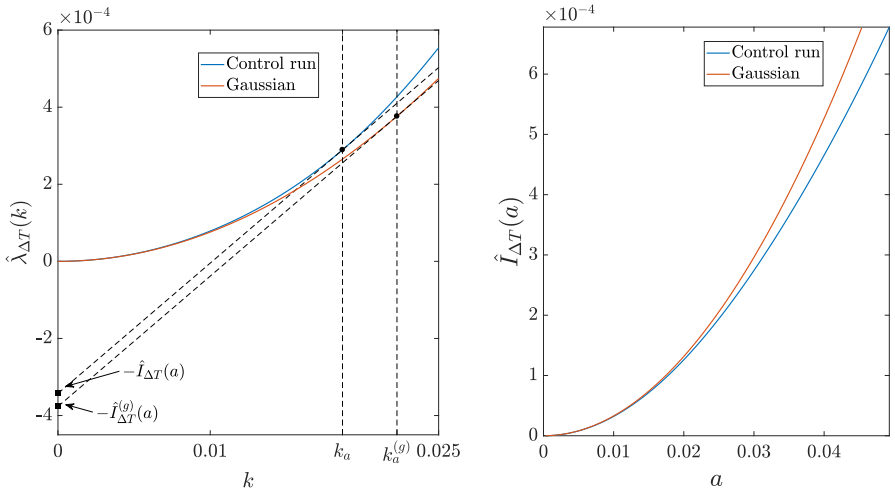


Figure 5.3: Reference estimates for the SCGF and rate function for the drag force acting on the square cylinder embedded in test flow (2). **(a)** Estimate of the SCGF $\hat{\lambda}_{\Delta T}(k)$, based on a timeseries spanning $T = 10^6\tau_c$, using the estimator (5.4) with blocks of length $\Delta T = \Delta T^\star = 6\tau_c$. Following figure 5.2, this estimate is only relevant until $k \leq k_c \approx 0.02$. This is due to the linearization effect (5.5). In addition, the Gaussian estimate (5.7) is displayed. As explained in note 1, the value of the rate function $I(a)$ for a particular k_a can be graphically deduced from the SCGF. **(b)** Estimate of the rate function, computed from $\hat{\lambda}_{\Delta T}(k)$ as a Legendre transform. More precisely, for each value of k we compute $\hat{I}_{\Delta T}(a) = ka - \hat{\lambda}_{\Delta T}(k)$, where $a = \hat{\lambda}'_{\Delta T}(k)$. This relation results from the Gartner-Ellis theorem (4.18).

a critical value k_c. Indeed, the computation of the SCGF involves the estimation of the average

$$\mathbb{E}\left[e^{k\int_0^T f_d(t)\mathrm{d}t}\right].$$

As k is increased, extreme values of $\int_0^T f_d(t)\mathrm{d}t$ have an increasing contribution to the average. For $k \geq k_c$, not enough samples are available in the timeseries for an accurate computation of the average. The objective of the GKTL algorithm, described in the previous chapter, is to sample preferentially these extreme values, leading to a better estimation of the SCGF.

In this section, we describe the computation of the SCGF using the GKTL algorithm for the statistics for the time-averaged drag. To begin with, we discuss the choice of the different parameters of the algorithm. Then, we compare the estimation of the SCGF resulting from the GKTL algorithm to the reference estimate obtained in the previous section. We show that the GKTL algorithm yields an accurate estimate of the SCGF in the region where the reference estimate is valid. In addition, the GKTL estimate is compared to a second direct estimate obtained with a similar computational cost as the GKTL run. It shows that the GKTL algorithm allows for the estimation of the SCGF over a larger range. However, we illustrate that, similarly to the direct estimates computed as (5.4), the validity of the GKTL estimate is limited to values $k \leq k_c^{alg}$, where k_c^{alg} is a critical value of the bias k past which the GKTL estimate becomes linear.

5.1.2.1 *Setting the parameters*

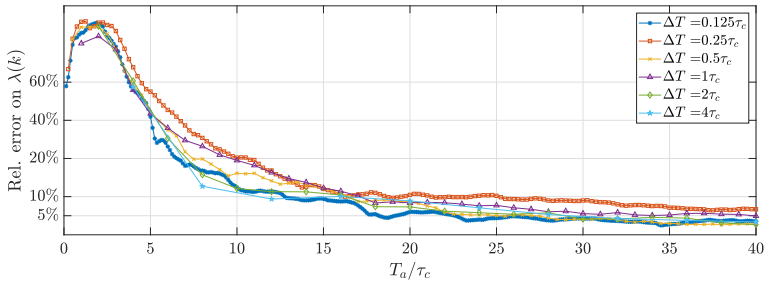
The GKTL algorithm depends on four parameters: the duration of the trajectories T_a, the cloning period τ, the number of trajectories N_c and, for deterministic systems, the perturbation amplitude ϵ required for clones to separate over time, see 4.2.1. Recall that the algorithm can be shown to yields the value of the SCGF $\lambda(k)$ in the double limit $T_a \to \infty$ and $N_c \to \infty$. See chapter 4 for more details about the algorithm.

THE DURATION OF THE TRAJECTORIES T_a At first sight, a natural value for T_a is the estimate of the large time limit found in the previous section, $\Delta T^\star$. As a matter of fact, the algorithm first undergoes an equilibration period T_{eq} before the series of cloning stages effectively results in importance sampling the biased distribution $\mathcal{P}_k$. Only when this stationary regime is attained is the estimate of the SCGF converged. This transient regime is illustrated in figure 5.4, where it can be seen that T_{eq} is actually much longer that the large-time limit $\Delta T^\star$ estimated in the previous section. For an accurate computation of the SCGF, one must therefore set $T_a \gtrsim T_{eq}$.

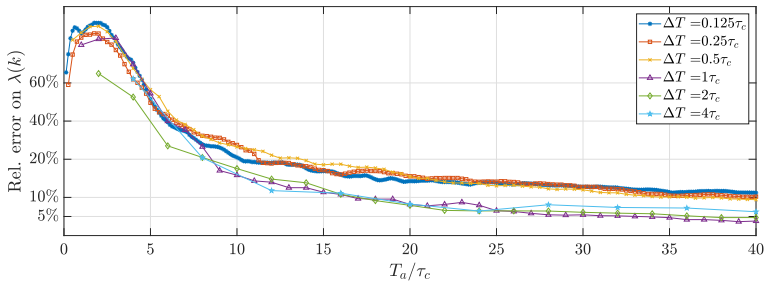
THE NUMBER OF COPIES N_c With T_a fixed, N_c determines the overall computational cost of the algorithm. The objective of this section is to compare the results from the algorithm to a reference solution computed by means of direct sampling. In this case, such reference solution was computed from a timeseries spanning $10^6\tau_c$. The number of copies is then chosen so that $N_c \times T_a \ll 10^6\tau_c$. More precisely, we perform an experiment with $N_c = 256$. As a comparison, a second experiment is carried out with $N_c = 1536$.

THE CLONING PERIOD τ It determines the frequency at which selection is imposed over the ensemble of copies. Recall from chapter 3 that fluctuations of the averaged drag over long durations result from independent short-lived fluctuations, occurring over roughly one correlation time τ_c of the drag. The role of the GKTL algorithm is to capture these short-lived fluctuations, in order to sample trajectories with an unusually high, or low, average over T_a. In addition, recall that the correlation time τ_c can be thought of as the timescale over which the drag decorrelates from its previous values, and is therefore close to the time over which two clones decorrelate, following the introduction of a perturbation in the flow. These aspects are discussed in more details in chapter 4, on page 101. In order to guide the choice of τ, we describe two limit cases. First, let us choose τ such that $\tau \ll \tau_c$. In this case, clones do not separate before the selection stage is performed. This actually result in loss of information, as good candidates can be discarded from the ensemble before they achieve a large fluctuation over an interval τ_c. On the other hand, choosing $\tau_c \gg \tau_c$ leads to a poor sampling of fluctuations over durations of the order of τ_c, and therefore to a poor sampling of extreme fluctuations of the averaged drag F_T. A rule of thumb for the choice of the cloning period τ is therefore $\tau \approx \tau_c$ [59].

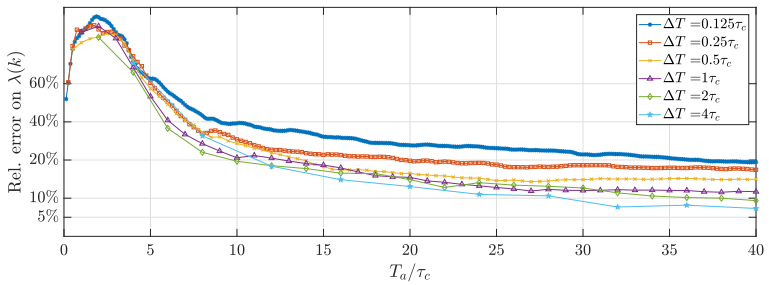
Following the cloning stage, a small perturbation is introduced for each clone that has been discarded and that restarts on the state of another copy. This is required so that they explore different regions of phase space over time. Choosing a perturbation with a large amplitude can however result in spurious fluctuations that would not occur in unperturbed dynamics. On the other hand, a perturbation of very small amplitude results in an increased separation time of the clones. The perturbation amplitude ϵ should therefore be chosen so that clones separate over a duration close to the cloning period τ. A way to check that the perturbation does not significantly alters the statistics is to perform a control simulation of the flow, introducing a perturbation with a period τ. Statistics computed over this perturbed control run must not differ from statistics computed over unperturbed dynamics.



(a) $\epsilon = 0.002$



(b) $\epsilon = 0.02$



(c) $\epsilon = 0.06$

Figure 5.4: Convergence of the estimate of the SCGF from the GKTL algorithm as the length of the trajectories T_a in increased, for several choice of the cloning period τ and perturbation amplitude ϵ. These plots illustrates the equilibration period the algorithm goes through before converging towards a stationary regime. As ϵ is increased, the estimates based on a low cloning period ΔT provide less accurate results.

5.1.2.2 *Description of the numerical experiment*

The objective of the present experiment is to assess the validity and efficiency of the estimation of the SCGF from the GKTL algorithm. To do so, the GKTL estimate will be compared to the reference solution computed on the basis of a very long timeseries. This computation is described in section 5.1.1. Recall that this reference estimate is only valid for $0 \leq k \leq 0.02$, and that its computational cost is $C_{ref} = 10^6 \tau_c$.

We perform 50 independent GKTL runs for 50 equally spaced values of the bias k in the interval $[0.005; 0.02]$. For $k \leq 0.005$, the SCGF is very well approximated by the Gaussian estimate—see figure 5.3—and its estimation poses no challenge. With the exception of k, the different parameters are equal among the GKTL runs. Consistently with figure 5.4, the total duration of the trajectories is set to $T_a = 40\tau_c$. In addition, the number of copies is set to $N_c = 256$. The computational cost of each GKTL run is therefore $C_{alg} = N_c \times T_a \approx C_{ref}/100$. The cloning period τ is set to $\tau = \tau_c/2$. The amplitude of the perturbation introduced for new copies is $\epsilon = 0.002$.

In order to assess the efficiency of the GKTL algorithm, we also estimated the SCGF from a timeseries of duration $C_{alg} = C_{ref}/100$, using the direct estimator (4.4). This computation is identical to the computation of the reference estimate presented in the previous section, except for the total duration of the timeseries, that is 100 times shorter. In practice the control timeseries, spanning $T_{tot} = 10^6 \tau_c$, can be split into 100 independent shorter timeseries of duration $10^4 \tau_c$, on the basis of which a direct estimate of the SCGF can be computed.

Figure 5.5 displays the estimates $\hat{\lambda}(k)$ for the 50 GKTL runs, as a function of k, in the domain of validity of the reference estimate. Each GKTL estimate has a computational cost C_{alg}. The accuracy of such estimates can be assessed by comparison with the reference estimate, computed with a computational cost $C_{ref} = 100 \times C_{alg}$. Last but not least, figure 5.5 displays two independent direct estimates from shorter timeseries, with a computational cost $C_{alg} = C_{ref}/100$. The latter estimates clearly illustrate the *linearization effect*, discussed in section 5.1.1. The linear behaviour of the estimate as k in increased is an artefact resulting from the finiteness of the timeseries. The slope of this tail varies from one estimate to another, as its value correspond to the highest fluctuation sampled in the underlying timeseries. See section 5.1.1 for a discussion of direct estimates of the SCGF and the linearization effect. Figure 5.5 illustrates that the GKTL algorithm allows for an estimation of the SCGF over a much larger range, with respect to a direct estimation with the *same computational cost*.

IMPORTANT REMARK One could argue that, because we performed 50 independent runs, the computation of the GKTL estimate over the *whole* domain $k \in [0.005; 0.02]$ has a computational cost of $50 \times C_{alg} \approx C_{ref}$. While this is true, we stress that, in order to compute

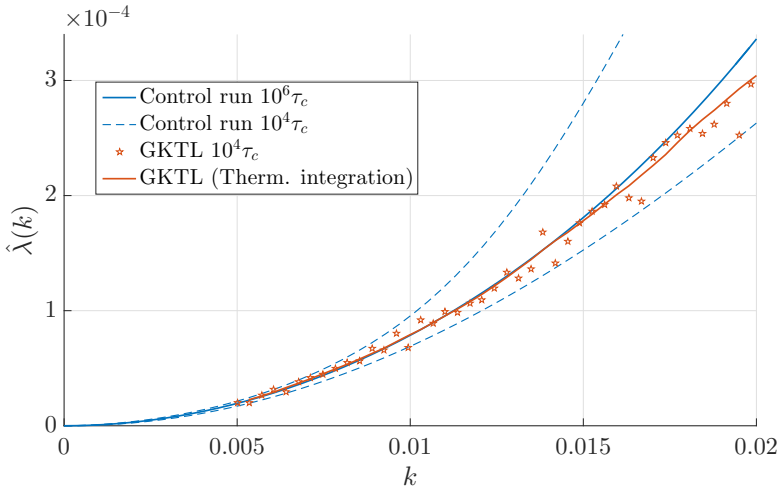


Figure 5.5: Comparison of the estimation of the SCGF from either the GKTL algorithm or a direct estimation from a timeseries of similar computational cost. The parameters for the GKTL experiments are $N_c = 256$, $T_a = 40\tau_c$, $\tau = \tau_c/2$ and $\epsilon = 2 \times 10^{-3}$. Each orange star correspond to a single GKTL run for a specific value of the bias k. Each run has a computational cost $C_{alg} = N_c \times T_a \approx 10^4\tau_c$. The continuous blue line represents the reference solution, computed from a time series of computational cost $C_{ref} = 10^6\tau_c$. The dashed blue lines correspond to two independent realisations of the direct estimation of the SCGF with a cost $C_{alg} = 10^4\tau_c$. They are computed on the basis of two independent chunks of duration $10^4\tau_c$ extracted from the reference timeseries of duration $10^6\tau_c$. The continuous orange line corresponds to estimate of the SCGF based on the GKTL runs, computed through thermodynamic integration, see note 2. This figure illustrates that, for a similar computational cost, the GKTL algorithm yields an accurate estimate of the SCGF over a larger range compared with direct estimation from a timeseries.

a direct estimate for a particular value $k = k^\star$, the SCGF must also be estimated for $k \leq k^\star$. By contrast, the GKTL algorithm directly targets $k = k^\star$, yielding a single estimate $\hat{\lambda}(k = k^\star)$. As an example, consider the estimation of the SCGF for $k^\star = 0.02$. As discussed in section 5.1.1, the minimum duration of the timeseries on which a reliable direct estimate can be computed is $T_{tot} = 10^6\tau_c$, so as to avoid the linearization effect. By contrast, figure 5.5 illustrates that this value can be approached—up to errors of roughly 10%— by the GKTL algorithm with a computational cost 100 *times lower*.

Note 2. Thermodynamic integration

Let F_{T_a} be the time averaged drag over a duration T_a. Recall from page 92 the following property of the SCGF:

$$\mathbb{E}_{\mathcal{P}_k}[F_T] \underset{T\to\infty}{\approx} \lambda'(k), \tag{5.11}$$

where $\mathbb{E}_{\mathcal{P}_k}[.]$ denotes the expectation value with respect to the biased path measure resulting from the GKTL algorithm. The SCGF can therefore be written as

an integral over previous runs for lower values of k. In particular, taking $T = T_a$ on can write

$$\lambda(k^\star) = \int_0^{k^\star} \frac{1}{N_c} \sum_{j=1}^{N_c} F_{T_a}^{(j,k)} \mathrm{d}k, \tag{5.12}$$

where the expectation value $\mathbb{E}_{\mathcal{P}_k}[F_{T_a}]$ is approximated as the empirical average over the trajectories sampled by the algorithm. More precisely, $F_{T_a}^{(j,k)}$ denotes the time average over the whole trajectory j in a GKTL run with bias k. Provided that one has performed several GKTL runs for values $k \leq k^\star$ with a relatively small spacing $\mathrm{d}k$, equation (5.12) provides a less noisy estimator for the SCGF at $k = k^\star$ [94, 103]. Note that this approach, referred to as thermodynamic integration, is a common tool for computing thermodynamic quantities, such as free energies, in molecular dynamics simulations [34].

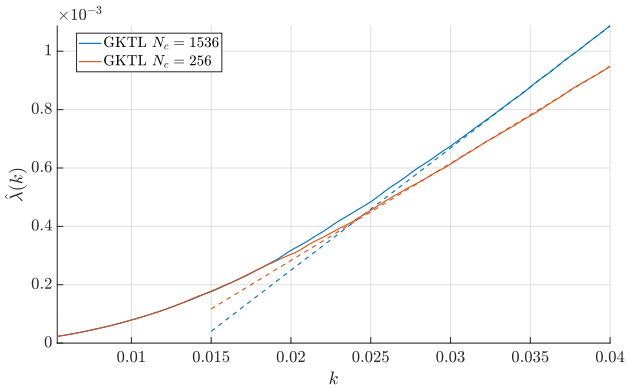
5.1.2.3 *How far can the GKTL algorithm go ? Limit of the GKTL estimate*

In the previous discussion we concluded that the GKTL algorithm leads to a better estimate of the SCGF with respect to a direct estimation from a timeseries. However, it can be seen in figure 5.5 that the GKTL estimate slightly departs from the reference estimate for $k \gtrsim 0.015$. Figure 5.6a illustrates the GKTL estimate displayed in figure 5.5 on a wider range of values of the bias k (continuous orange line). It shows that, similarly with direct estimates from timeseries, the GKTL estimate becomes linear as k in increased. Recall from chapter 4, equation (4.22) that the derivative of the SCGF can be written as:

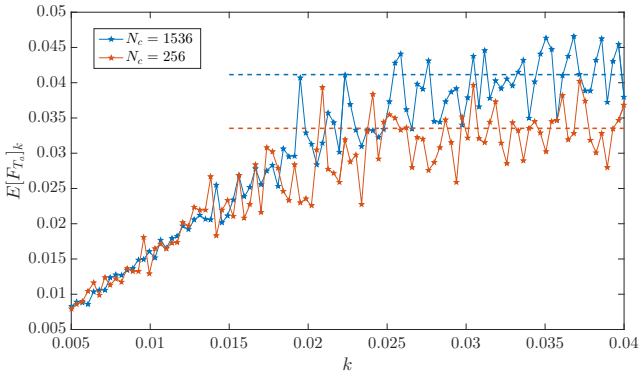
$$\mathbb{E}_{\mathcal{P}_k}[F_T] \underset{T\to\infty}{\approx} \lambda'(k) \tag{5.13}$$

where F_T denotes the time average over a timeseries of duration T and $\mathbb{E}_{\mathcal{P}_k}[F_T]$ the expectation value of F_T with respect to the biased measure $\mathcal{P}_k$ sampled by the algorithm. From figure 5.6a there exists a critical value of k_c past which $\hat{\lambda}'(k) \approx C, \forall k \geq k_c$ with C a constant independent of k. As a result, the value of $\mathbb{E}[F_T]_{\mathcal{P}_k}$ saturates as k is increased: $\mathbb{E}_{\mathcal{P}_k}[F_T] \approx C, \forall k \geq k_c$. Such a behaviour could be explained by the existence of an upper bound C to the values of F_T. However, this hypothesis is ruled out by a second GKTL experiment in which the number of copies is increased to $N_c = 1536$. Figure 5.6a illustrates that, for this experiment, the GKTL estimate of the SCGF also displays linear tails. However, the corresponding slope has a different value. Furthermore, the onset of linearity is attained for further values of the bias k. This result suggests that the linear behaviour of the GKTL estimates for high values of k is artificial: it is due to the algorithm, instead of the physics of the flow.

Both estimates displayed in figure 5.6a have been computed from 100 *independent* GKTL runs at equally spaced values of the bias k in



(a) GKTL estimate for $\lambda(k)$ for different N_c



(b)

Figure 5.6: **(a)** Estimation of the SCGF over a large range of values for the bias k. The direct estimate (dashed) is only valid for $k \leq 0.02$. Past this value, its linear behaviour is an artefact resulting from the finiteness of the timeseries. The continuous lines represent two estimates of the SCGF using the GKTL algorithm with 256 copies and 1536 copies, respectively, other parameters being equal. The total duration of the trajectories is $T_a = 40\tau_c$, $\tau = \tau_c/2$ and $\epsilon = 0.002$. both estimates result from the thermodynamic integration of 100 GKTL runs for equally spaced values of the bias k in the interval $[0.005; 0.04]$. One can see that both estimates become linear as k is increased. **(b)** Average value of the time-averaged drag over the N_c sampled trajectories, for each GKTL run. The time average is computed over the whole trajectories: $F_{T_a}^{(j,k)} = \frac{1}{T_a}\int_0^{T_a} f_d^{(j,k)}(t)\mathrm{d}t$ where $\{f_d^{(j,k^\star)}(t)\}_{0\leq t\leq T_a}$ denotes the drag timeseries for copy j of the GKTL run with $k = k^\star$. The expectation value is approximated by an empirical mean over the N_c trajectories, *i.e.* $\mathbb{E}[F_{T_a}]_k \approx \frac{1}{N_c}\sum_{j=1}^{N_c} F_{T_a}^{(j,k)}$. This approximation is discussed in section 5.2.4 further below.

the interval $[0.005; 0.04]$. The continuous lines result from a thermodynamic integration based on those 100 runs. See note 2 for a discussion of thermodynamic integration. Figure 5.6b displays the value of $\mathbb{E}_{\mathcal{P}_k}[F_T]$ as a function of k, where T is chosen as the total duration of the trajectories, *i.e.* $T = T_a$. In the following we therefore use the notation $\mathbb{E}_{\mathcal{P}_k}[F_{T_a}]$. It is displayed in figure 5.6b for each of the 100 independent runs, for the two experiments with $N_c = 256$ and $N_c = 1536$, respectively. For each run, the value of $\mathbb{E}_{\mathcal{P}_k}[F_{T_a}]$ is approximated by the empirical average of F_{T_a} over the N_c trajectories, see the caption of figure 5.6b. As can be seen in figure 5.6b, the estimate of $\mathbb{E}_{\mathcal{P}_k}[F_{T_a}]$ fluctuate from one run to another, as each run is independent from the other. However, it shows a clear tendency: for lower value of k, the value of $\mathbb{E}_{\mathcal{P}_k}[F_{T_a}]$ increases as a function of k. This is expected, as the algorithm imposes a stronger selection, sampling higher values of the averaged drag with a higher importance. By contrast, for higher values of the bias k, figure 5.6b illustrates that the value of $\mathbb{E}_{\mathcal{P}_k}[F_{T_a}]$ saturates and fluctuates around a limit value independent of k. Moreover, this limit value, which we called C in the above, depends on the number of copies N_c.

To sum up, figure 5.6 suggests that, for a given GKTL experiment with N_c and T_a, there exist a critical value of the bias $k_c = k_c(N_c)$ past which the amplitude if the averaged drag fluctuations sampled by the algorithm saturates. Moreover, this limit amplitude depend on the choice of the number of copies: $C = C(N_c)$. The higher N_c, the further the GKTL can go. We stress that this behaviour is *systematic*, as opposed to *statistical*. Note that this limit amplitude C may also bear dependence on the other parameters of the algorithm, such as the duration T_a, the cloning period τ or the perturbation amplitude ϵ. This dependence was not investigated in this work and is left to further studies.

5.2 ANALYSING GKTL DATA

In the previous section, we applied the GKTL algorithm to the evaluation of the large deviation rate function describing the probability of rare drag fluctuations on the square cylinder embedded in test flow (2). By comparison with a direct evaluation of similar computational cost, we illustrated that the GKTL algorithm leads to a far more efficient estimation of the rate function. Large deviations rate functions, as well as Scaled Cumulant Generating Functions and Probability Density Functions are useful tools to describe the *statistics* of observables. Indeed, they tell us how frequent a typical fluctuation can be expected to be, or how unlikely is an extreme fluctuation with respect to typical events. However, such statistical quantities do not provide insight into the *dynamics* leading these fluctuations. Yet, understanding of the

physical mechanisms underlying extreme fluctuations in turbulent flows is of just as much interest as the description of their statistics. In this section we discuss the study of the dynamics behind extreme drag fluctuations using the GKTL algorithm.

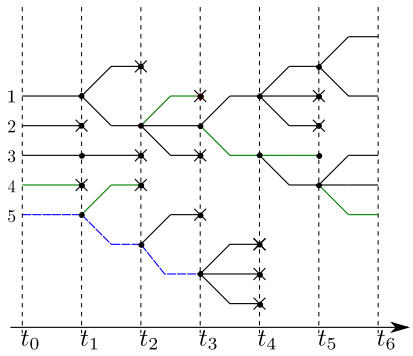
As described in 4, the GKTL algorithm relies on the effective simulation of the underlying dynamical process. In the case of this work, this corresponds to solving the Lattice Boltzmann Equation in the computational domain, see section 2.1. In addition to the estimation of the SCGF, it results in an ensemble of N_c simulations of the flow, where N_c denotes the number of copies involved in the algorithm. In principle, each one of these simulation correspond to the simulation of an extreme event, that can be visualised and analysed like any other numerical simulation of fluid flows. The GKTL algorithm therefore appears like a formidable tool to investigate the physics of extremes in turbulent flows.

5.2.1 *Discontinuous and reconstructed trajectories*

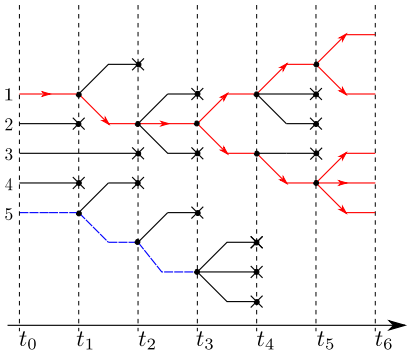
We now make an important remark. The algorithm does not directly outputs the sampled trajectories corresponding to rare events. Instead, it leads to *discontinuous* paths that result from the branchings resulting from the cloning stages along the course of the algorithm. This aspect is better understood through the visualisation of the algorithm presented in figure 5.7.

Following a cloning stage, discarded copies—marked by a cross in figure 5.7—continue their evolution through the next evolution stage starting from the final state of another copy. In order to compute expectation values over the original stationary PDF $\mathcal{P}_0$, based on the biased sampling introduced by the algorithm, one must compute the integral of the observable over the continuous trajectories effectively sampled from $\mathcal{P}_k$. This can be seen from equation (5.1), and will be explained in more details in section 5.2.4 below. The continuous paths are sketched by red lines with arrows in figure 5.7. In the following we refer to the computation of the continuous trajectories from the discontinuous paths as *the reconstruction* of trajectories. Continuous trajectories are referred to as *reconstructed trajectories*. In section 5.2.2, we give a precise definition of the reconstruction procedure, for any observable of the dynamics.

Naturally, the reconstruction of the continuous trajectories is crucial to the study of atypical trajectories sampled through the GKTL algorithm. In practice, it can take many different forms, depending on the data that is required. For instance, one could only need the values of the average drag over the continuous trajectories, so as to compute expectation values as will be explained in section 5.2.4. One



(a) Green lines: discontinuous path for copy 4



(b) Red lines: Continuous reconstructed trajectories

Figure 5.7: Visualisation of the GKTL algorithm in terms of discontinuous and continuous paths in phase space. The M iterations of the evolution/cloning procedure are separated by vertical dashed lines. Within a particular iteration m, continuous lines denote the evolution of the four copies over $[t_{m-1}; t_m]$. Dots mark the cloning stages. Therefore, to each iteration m correspond $N_c = 4$ *partial trajectories* $\{\mathbf{X}_{(j)}(t)\}_{t_{m-1} \leq t \leq t_m}$, $1 \leq j \leq 4$. The starting point of the algorithm is $t_0 = 0$ from which copies evolve until $t_1 = \tau$. At this point, copies $j = 2$ and $j = 4$ are discarded and branched on copies $j = 1$ and $j = 5$ that result in two clones. This procedure is repeated $M - 1 = 5$ times until the last evolution step from $t_5 = 5\tau$ to $t_6 = 6\tau$ after which the algorithm terminates. Figure 5.7a displays the discontinuous paths for copy $j = 4$ as green lines. Discontinuities result from the different cloning stages. In contrast, figure 5.7b displays the continuous reconstructed trajectories (red lines with arrows) for all four copies. As a consequence of the various cloning stages, all reconstructed trajectories originate from $j = 1$ and strongly overlap over a significant part of their history. To each trajectories, either discontinuous or continuous, can be associated a timeseries for a given observable O. In section 5.2.3, we discuss different methods of obtaining the reconstructed trajectories in figure 5.7b from the discontinuous paths shown in 5.7b. In a first method, the whole discontinuous paths are written on disk. This approach is straightforward to implement but leads to unnecessary storage. For instance, one can see in figure 5.7a that the whole branch originating from copy $j = 5$ is not involved in any of the reconstructed trajectories. A straightforward alternative is not to write the partial timeseries for clones that are discarded. However, some timeseries will be written even though they are irrelevant to the reconstructed trajectories. This the case for example of the dashed partial timeseries coloured in blue in both figures above. A solution for automatically erase irrelevant partial timeseries is *reference counting*, see note 3. A third method consists in saving the trajectories in memory along the iterations, and overwrite partial trajectories for discarded copies. In this way reconstruction is achieved on-the-fly.

could also want to compute the timeseries for an observable along the reconstructed trajectories, for instance the drag acting on the obstacle, or visualise the dynamics of the flow fields. In section 5.2.3 we discuss several ways of performing the reconstruction in practice, depending on the situation.

Finally, in section 5.2.5 we illustrate the efficiency of the GKTL algorithm based on the sampling of trajectories leading to extreme drag fluctuations. We show that, for equal computational costs, the GKTL algorithm provides a greater number of realisations of extreme events with respect to direct sampling. Moreover, we illustrate that it allows for the study of the dynamics of extreme events that would be unreachable by a direct approach. This is complementary to the illustration of the efficient evaluation of the large deviation rate function in section 5.1.2. However, we point out that most trajectories overlap at early times

5.2.2 *Reconstruction of the continuous trajectories*

In the following we denote by $T_a = M\tau$ the total length of the trajectories generated by the algorithm, with M the number of stages and τ the cloning period. Recall that an *iteration* refers to the parallel temporal evolution of all the copies over a duration τ, followed by a cloning stage in which some are discarded an others duplicated. In this section, we consider that the algorithm terminates by an evolution stage, that is, the Mth cloning stage is not actually performed. By omitting the last cloning step, we assure that trajectories *at least* differ over the interval $[T_a - \tau; Ta]$, see figure 5.7. Please refer to chapter 4 for further details about the algorithm itself.

Let $\mathbf{X}(t)$ denote the state of the flow, corresponding to the solution of the numerical model, at time t. In the case of the Lattice Boltzmann Method, $\mathbf{X}(t)$ represents the ensemble of mesoscopic populations at each node of the lattice, see chapter 2, section 2.1 for a discussion of the Lattice Boltzmann Method. Furthermore, let us denote by $\{\tilde{\mathbf{X}}_j(t)\}_{0\leq t\leq T_a}$ the discontinuous path for copy j over the course of the algorithm. An example is given in figure 5.7a. In addition, let $\{\mathbf{X}_j(t)\}_{0\leq t\leq T_a}$ be the continuous trajectory corresponding to copy j. In the following we label by i the iterations of the algorithms, so that the cloning times are $\{t_i\}_{0\leq i\leq M-1}$ with $t_i = i\tau$. Note that we view the $t_0 = 0$ as a cloning time, even though no actual cloning occurs. Finally, let us denote by $p_j^{(i)}$ the *parent* of the copy j at iteration i. The parent refers to the copy from which the copy j pulled its initial condition following cloning stage i. As the algorithm terminates by an evolution stage, there is actually no cloning stage M and we define $p_j^{(M)} = j$.

The reconstruction consists in tracing back the history of the copies from the last evolution stage for $i = M$ to the first, for $i = 0$. More precisely, it reads

$$\{\mathbf{X}_j(t)\}_{t_i \le t \le t_{i+1}} = \{\tilde{\mathbf{X}}_{p_j^{(i)}}(t)\}_{t_i \le t \le t_{i+1}}, \quad 0 \le i \le M-1, \quad 1 \le j \le N_c \tag{5.14}$$

The reconstruction is illustrated in figure 5.7. In practice, we are not interested in the states $\mathbf{X}$ themselves, but on physical observables that depend on the current state visited by the flow. For instance, the velocity field in the computational domain, or the drag acting on an obstacle. In the following section we discuss several ways of implementing the reconstruction, either on-the-fly over the course of the algorithm, or as a post-processing step.

5.2.3 *Implementation(s) of the reconstruction in practice*

In this section we discuss several ways of performing the reconstruction (5.14), depending on the observable(s) of interest. Let O denote a given observable. For instance it could be the drag acting on the obstacle ($O \equiv f_d$) or the velocity field on the computational domain: $O \equiv \mathbf{u}$. In the following we refer to a *timeseries* as any sequence of measurements of O at equally spaced points in time, whether O is a scalar or a scalar field.

5.2.3.1 *Method 1: Writing the full discontinuous timeseries on disk*

We begin by discussing the most straightforward way, implementation-wise, of carrying out the reconstruction (5.14). It consist in a brute force approach in which the observable of interest is written on disk during the evolution stages. The reconstruction is performed as a post-processing step, after the algorithm is terminated.

Recall from chapter 4 that the simulation of each copy during the evolution stages is carried out by N_p processes over which the N_c copies are dispatched. In this first method the current value of O along the evolution of each copy is periodically written on disk by each process. This corresponds to directly writing O on disk along the discontinuous paths, of which one example is given in figure 5.7a. A second ingredient is to keep track of the label $p_j^{(i)}$ of the parent copy for each copy after the cloning stage. In this way, the timeseries for O can be reconstructed by tracing backwards the algorithm: first $\{O[\mathbf{X}_j(t)]\}_{T-\tau \le t \le T_a} = \{O[\tilde{\mathbf{X}}_j(t)]\}\}_{T_a-\tau \le t \le T_a}$, then $\{O[\mathbf{X}_j(t)]\}_{T-2\tau \le t \le T_a-\tau} = \{O[\tilde{\mathbf{X}}_{p_j^{(i)}}(t)]\}(t)\}_{T_a-2\tau \le t \le T_a-\tau}$ ... and so on.

Clearly, this method is not optimal, neither in terms of memory management nor in terms of the time spent writing data on disk. In addition, it entails that the whole N_c discontinuous timeseries for O

must be written on disk, leading to unnecessary data storage. Indeed, only a small part of this data is actually involved in the reconstructed timeseries. However, it has the advantage of being straightforward to implement, as processes perform the same operations and do no not have to communicate with one another. Furthermore, debugging is facilitated as the reconstruction procedure is totally separated from the algorithm itself. In the following we discuss a variant of the reconstruction procedure in which the timeseries for O is temporarily saved in memory before being written on disk. Note that in the case where O is a observable consisting of a large number of real numbers, such as the velocity field with a high spatial resolution, it might not be possible to store the timeseries of the field in memory. In this case there may be no choice but to write fields directly on disk. We stress however that it is not the case of this work. For instance, a snapshot of the velocity field for test flow (2) amounts to roughly 500KB of data.

5.2.3.2 *Method 2: On-the-fly reconstruction*

A different way of obtaining the reconstructed timeseries is to perform the reconstruction on-the-fly over the course of the algorithm. In this approach the whole timeseries $\{O_j(t)\}_{0\leq t\leq T_a}$ is stored in memory. As an example, consider the case say in which copy j is discarded and replaced by a clone of copy l, following cloning at stage i. As a consequence, the timeseries for copy j up to time $t_i = i\tau$, $\{O_j(t)\}_{0\leq t\leq t_i}$, is overwritten by the timeseries for copy l. The obvious advantage of this method is that the GKTL directly yields the reconstructed timeseries, bypassing the need for an additional reconstruction step. It also mitigates the number of disk writings as timeseries are stored in memory and are written on disk only once the algorithm terminates. However, it does not mitigate disk usage as, in the end, just as much data is written on disk as in the case of the previous method. As illustrated in figure 5.7, trajectories overlap over a significant part of their history. As a result, most reconstructed timeseries will contain the same data.

This method is actually most useful when computing the time integral of O along the reconstructed trajectories, without having to actually store the reconstructed timeseries for O. Along evolution stage i, from t_i to t_{i+1}, the time-integral of O is updated as follows:

$$\int_0^t O[\tilde{\mathbf{X}}(\tau)]\mathrm{d}\tau = \int_0^{t_i} O[\tilde{\mathbf{X}}(\tau)]\mathrm{d}\tau + \int_{t_i}^t O[\tilde{\mathbf{X}}(\tau)]\mathrm{d}\tau \tag{5.15}$$

If copy i is discarded and replaced by a clone of copy l, the final value of the integral after stage j, $\int_0^{t_i} O_j[\tilde{\mathbf{X}}(\tau)]\mathrm{d}\tau$, is overwritten by the value for copy l.

5.2.3.3 *Method 3: Minimum file I/O*

We finally present a third way of performing the reconstruction, that minimises data storage and file Input/Output. Method 1 relies on continuously writing the timeseries for O on disk over the course of the algorithm and perform the reconstruction as a post-processing step. In contrast, in method 2 the whole timeseries for O is saved in memory and each time a copy is discarded from the ensemble, its timeseries is overwritten.

Method 3 sits in between method 1 and 2. In this approach, the partial timeseries $\{O_j(t)\}_{0\leq t\leq t_i}$ is saved in memory along evolution stage i. Following the corresponding cloning stage at iteration i, this partial timeseries is written on disk only if copy j is not discarded. Reconstruction is performed as a post-processing step, similarly to method 1. Note that this indeed mitigates the amount of data written on disk, but does not reduces it to its minimum. Indeed, when a copy is discarded, its whole trajectory becomes irrelevant to the reconstruction procedure. This can be seen on figure 5.7: the partial timeseries corresponding to the dashed blue lines are written on disk, although they are not involved in any of the reconstructed trajectories, depicted by red lines in figure 5.7b. A way to reduce the amount of disk usage to its minimum is to employ *reference counting*, described in note 3. Its use has been mentioned in the application of the AMS algorithm (see chapter 6) to molecular dynamics simulations, facing the same kind of memory management issue [5]. In this way the partial timeseries $\{O_{(j)}(t)\}_{0\leq t\leq t_i}$ that are no longer relevant to the reconstructed timeseries are automatically erased from disk.

Note 3. Reference counting

In computer science, reference counting is a common approach to implementing garbage collection [175], that is automatic reclaiming of memory allocated for objects that are no longer needed by the program. The situation of the GKTL algorithm is analogous, as one wants to automatically get rid of partial timeseries irrelevant to the reconstructed timeseries. The GKTL algorithm can be visualised as an array of $N \times N_c$ partial timeseries, illustrated in figure 5.7. A partial timeseries is label by a couple (i, j), where i denotes the current iteration and j the index of the copy. Reference counting consists in associating to each partial timeseries the number of reconstructed trajectories in which it is involved, referred to as the reference count. This can be done on the fly over the course of the algorithm. Indeed, if a copy is discarded after cloning stage i, its history is traced back through stages $i-1$, $i-2$.. following the corresponding parent labels $p_j^{(i)}$ and decrementing the reference count of the corresponding partial timeseries. If the reference count of a partial timeseries reaches 0, it is erased.

5.2.3.4 *Flexible reconstruction*

In the previous paragraphs we presented three different methods for computing the reconstructed timeseries for an observable O, on

the basis on the discontinuous path effectively computed by the algorithm. As a matter of fact, one may be interested in several different observables. Furthermore, the observables of interest must be chosen *before* the algorithm is performed.

In order to gain flexibility in the post-processing, the algorithm can be implemented in a way that the state $\mathbf{X}$ of each copy is written on disk before each evolution stage. Because the dynamics are deterministic, the reconstruction procedure described above can be adapted to re-simulate the dynamics, in a piece-wise manner, over the evolution stages $[T_a - \tau; T_a]$, $[T_a - 2\tau; T_a - \tau]$, $[T_a - 3\tau; T_a - 2\tau]$, etc.. In this way one is able to compute any observable along the reconstructed trajectories, as a post-processing step. This can be useful, for instance, if one is just interested in visualising the flow fields for only a subset of trajectories. Consistently with method 3, states at iteration i can be saved in memory and written only if the copy is not discarded. Additionally, reference counting can be applied to erase irrelevant states over the course of the algorithm.

5.2.4 *Computation of expectation values over the reconstructed ensemble*

Reconstructed trajectories are sampled according to the biased measure $\mathcal{P}_k$. Because the importance ratio between $\mathcal{P}_k$ and the original dynamical measure $\mathcal{P}_0$ is known, statistics for $\mathcal{P}_0$ can be computed from the knowledge of events sampled according to $\mathcal{P}_k$.

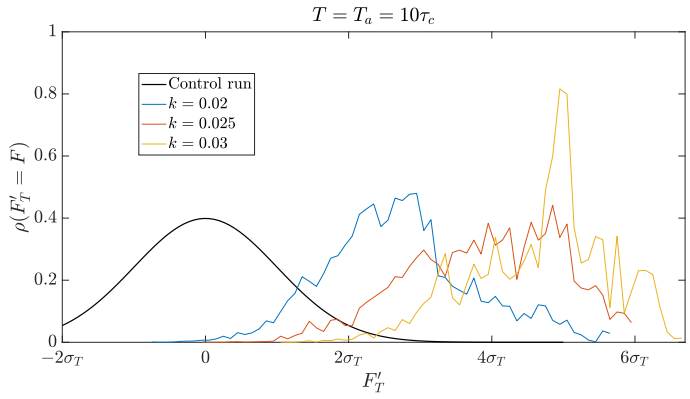
Following (5.1), each sampled trajectory is attributed a weight

$$Z(k, T_a) e^{-k \int_0^{T_a} f_d(\mathbf{x}(t)) \mathrm{d}t}$$

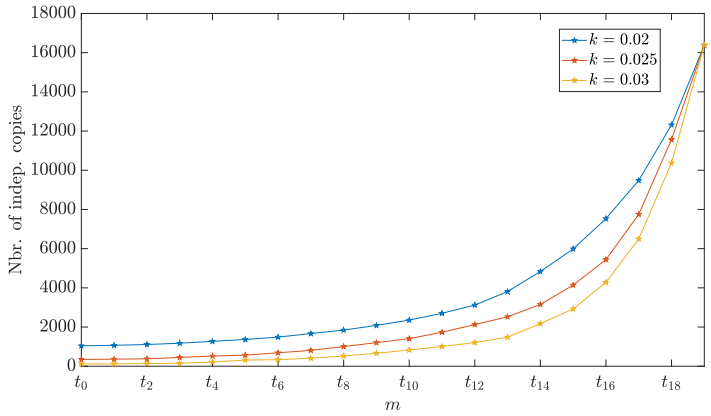
with $Z(k, T_a)$ computed as (4.19) on page 92. For any dynamical observable depending on a trajectory of the model, $O_{T_a}(\{\mathbf{X}(t)\}_{0 \le t \le T_a})$, an expectation value can be computed as an empirical average over the reconstructed trajectories

$$\begin{aligned} &\mathbb{E}_{\mathcal{P}_0}\left[O_{T_a}\left(\{\mathbf{X}(t)\}_{0 \le t \le T_a}\right)\right] \\ &\quad \underset{N_c \to \infty}{\sim} \frac{1}{N_c} \sum_{n=1}^{N_c} O_{T_a}\left(\{\mathbf{X}_n(t)\}_{0 \le t \le T_a}\right) \mathrm{e}^{-k \int_0^{T_a} f_d(X_n(t)) dt} Z(k, T_a), \end{aligned} \tag{5.16}$$

where the $\{X_n\}_{1 \le n \le N_c}$ are the reconstructed trajectories. Recall that reconstructed trajectories are not independent, and therefore the previous result does not simply results from the law of large numbers. However, it can be shown that (5.16) holds true as $N_c \to \infty$ [117].



(a) Illustration of importance sampling



(b) Independence of trajectories

Figure 5.8: **(a)**: Importance sampling for the time-averaged drag acting on the obstacle embedded in test flow (2). The drag is averaged over $T = 10\tau_c$. The GKTL algorithm is used with $N_c = 16384$ copies and $T_a = T = 10\tau_c$ for three increasing values of the bias k. Additionally, the cloning period is $\tau = \tau_c/2$, resulting in $M = 20$ iterations of the evolution/selection procedure. Note that in this particular example the last cloning stage at t_{19} is not performed. As k is increased, the sampled PDF is shifted towards extreme values. Along the X-axis we display the fluctuation $F'_T = F_T - \overline{F_T}$, where the average $\overline{F_T}$ and standard deviation σ_T have been computed over a timeseries spanning $10^6\tau_c$. One can see that the distribution for $k = 0.03$ is peaked for a particular value of the fluctuation. This indicates that a significant part of the sampled trajectories actually overlap over the major part if their history. A a result, the average drag only differ by a small amount. **(b)**: Number of independent copies among the ensemble of sampled trajectories as a function of the stage n. This illustrates that a major part of the trajectories overlap. This effect is illustrated in a different way in figure 5.7b.

5.2.5 *Importance sampling extreme drag fluctuations*

In section 5.1.2.2 we computed the rate function describing the *statistics* of extreme fluctuations of the time-averaged drag. However, it does not give information concerning the *dynamics* corresponding to rare events. In this section we address the question of the applicability of the GKTL algorithm to the investigation of the dynamical aspects of extreme events in turbulent flows.

We first make an important remark. Recall that the GKTL algorithm samples trajectories according to a biased path measure defined as

$$\mathcal{P}_k\left(\{\mathbf{x}(t)\}_{0<t<T_a}\right) = \frac{1}{Z(k,T_a)} \exp\left(k\int_0^{T_a} f_d[\mathbf{x}(t)]\mathrm{d}t\right) \mathcal{P}_0\left(\{\mathbf{x}(t)\}_{0<t<T_a}\right) \tag{5.17}$$

We stress that relation (5.17) is valid for any choice of T_a. Technically, the large time limit $T_a \to \infty$ is only required if one is interested in computing the large deviation properties of the observable, such as the SCGF or the rate function. In this section, we are only interested in sampling trajectories $\{\mathbf{X}(t)\}_{0\leq t\leq T_a}$ that exhibit large values of the averaged drag F_T, with $T \leq T_a$. As a result, in the following we consider durations that are shorter than the time T_{eq} it takes for the algorithm to yield a converged estimate of the SCGF. See figure 5.4 for an illustration of this convergence.

During each cloning stage of the GKTL algorithm, several trajectories are discarded from the ensemble and replaced by clones of other trajectories. As a result, most trajectories in the *reconstructed* ensemble are not independent, and overlap over a significant part of their history. This effect is illustrated in figure 5.7b on page 130. In this context, it is not clear if the *reconstructed* ensemble of trajectories sampled by the GKTL algorithm is useful to the investigation of the dynamics of rare events. Indeed, one has to check that the reconstructed ensemble contains more *independent* events than the number of fluctuations sampled in a long simulation of the flow with the same computational cost.

We therefore performed three GKTL experiments, corresponding to three different values of the bias k, all other parameters being equal. On the one hand, as k is increased, we expect the algorithm to sample rarer events. On the other hand, we expect the number of independent trajectories to decrease. In order to illustrate that the algorithm can be used with shorter durations $T_a \lesssim T_{eq}$, we set $T_a = 10\tau_c$. Note that choosing smaller durations T_a may lead to difficulties. Indeed, as pointed in the previous chapter, the cloning period τ should be set to a value of the order of the correlation time, in order for cloned trajectories to separate over time. As a consequence, choosing a duration T_a too close to the correlation time would result in a

small number of cloning stages, and therefore a poor sampling of rare trajectories. In addition we consider the time-averaged drag F_T defined in (5.2) with $T = T_a = 10\tau_c$. Therefore, equation (5.2) becomes

$$F_T = F_{T_a} = \frac{1}{T_a}\int_0^{T_a} f_d(t)\mathrm{d}t. \tag{5.18}$$

In the following experiments we apply the GKTL the algorithm with a high number of copies: we set $N_c = 16384$. Each experiment therefore has a computational cost of $N_c \times T_a = \mathcal{O}(10^5\tau_c)$. Finally, we set the cloning period $\tau = \tau_c/2$ and the perturbation amplitude $\epsilon = 0.02$.

Figure 5.8a illustrates the amplitude of the fluctuations of F_{T_a} corresponding to the sampled trajectories for each three experiments. More precisely, it illustrates the estimate of the biased drag PDF ρ_k in each case:

$$\begin{aligned}\rho_k(F) &= \mathbb{E}_{\mathcal{P}_k}[\delta(F_{T_a}[\{\mathbf{x}\}_{0\le t\le T_a}] - F)] \\ &\approx \frac{1}{N_c}\sum_{j=1}^{N_c}\delta(F_{T_a}^{(j)}[\{\mathbf{x}\}_{0\le t\le T_a}] - F).\end{aligned} \tag{5.19}$$

Moreover, figure 5.8a displays an estimate of the unbiased drag PDF $\rho_0(F) = \mathbb{E}_{\mathcal{P}_0}[\delta(F_{T_a} - F)]$. Note that this figure is analogous to figure 4.1 on page 86. One can see that, as k is increased, the biased PDF is shifted towards extreme values. However, the experiment with $k = 0.03$ does not seem to increase the sampling of rare event with respect to the experiment with $k = 0.025$. Actually, the shape of the estimate of the biased PDF shows that the reconstructed trajectories concentrate around a single value of the fluctuation $F'_{T_a} = F_{T_a} - \overline{F_{T_a}} \approx 5\sigma_{T_a}$, where $\overline{F_{T_a}}$ and σ_{T_a} denote the average and standard deviation of F_{T_a}, respectively. This suggests a strong overlap of trajectories in the reconstructed ensemble, which we address next.

Figure 5.8b illustrates the overlap of trajectories in the reconstructed ensemble. Along the X-axis, we represent the $M = T_a/\tau$ iterations of the evolution/selection procedure. For each iteration m we plot the number of independent trajectories, *i.e.* the number of trajectories that exhibit different paths $\{X(t)\}_{(m-1)\tau\le t\le m\tau}$ over the interval $t \in [(m-1)\tau; m\tau]$. In practice we trace back the history of the trajectories backward in time starting from the final time $T_a = M\tau$, in a way very similar to the *Method 1* presented in section 5.2.3. Figure 5.8b illustrates that, at early times, the vast majority of the trajectories in the reconstructed ensemble overlap. This is also illustrated, in a different way, by the red trajectories in figure 5.7b. Say N trajectories of the reconstructed ensemble overlap over the interval $[0; m\tau]$, with $1 \le m \le M$. Among these N trajectories, only one trajectory j actually

Overlap	σ	2σ	3σ	4σ	5σ
0%	759	258	32	5	0
25%	942	372	62	9	0
50%	1594	799	155	22	0
$N_{N_c}^{(gauss)}$	2599.408	372.738	22.117	0.519	0.005

(a) $k = 0.02$

Overlap	σ	2σ	3σ	4σ	5σ
0%	305	179	52	5	0
25%	451	310	144	39	4
50%	1019	834	521	198	27
$N_{N_c}^{(gauss)}$	2599.408	372.738	22.117	0.519	0.005

(b) $k = 0.025$

Overlap	σ	2σ	3σ	4σ	5σ
0%	100	80	33	8	1
25%	186	166	110	46	7
50%	539	510	391	205	36
$N_{N_c}^{(gauss)}$	2599.408	372.738	22.117	0.519	0.005

(c) $k = 0.03$

Table 5.1: Number of independent trajectories corresponding to fluctuations $F'_{T_a} \geq a$ with a=$\sigma, 2\sigma$... in the reconstructed ensemble, for the three experiments $k = 0.02, 0.025, 0.03$. Parameters of the GKTL experiments are $N_c = 16384$, $T_a = 10\tau_c$, $\tau = \tau_c/2$ and $\epsilon = 0.02$. In each case we count as independent trajectories with an overlap of 0% (zero overlap), 25% and 50%, respectively. In addition, $N_{N_c}^{(gauss)}$ is the average number of fluctuations $F'_{T_a} \geq a$ on can expect from N_c realisations obtained from a direct sampling approach. This number is based on the fact that the PDF of F_{T_a} is Gaussian. It is defined in (5.20).

originates from $t = 0$. Indeed, the $N - 1$ others have been discarded following the selection stage of a given iteration $\tilde{m} \geq m$, and replaced by a clone of trajectory j over the interval $[0; \tilde{m}\tau]$. In the following we call *fully independent trajectories* the trajectories that do not overlap at all over $[0; T_a]$, *i.e.* trajectories that do not overlap over the first period $[0; \tau]$. From figure 5.8b one can see that the number of fully independent trajectories is very small compared to the total number of trajectories N_c.

In this context, can the GKTL yield more *independent* events than a direct sampling based on a long simulation ? If yes, how much more ? Table 5.1 displays the number of independent trajectories that correspond to fluctuations $F'_{T_a} \geq a$ with $a = \sigma_{T_a}, 2\sigma_{T_a}, 3\sigma_{T_a}$..., for

each of the three experiments. In each case, we count the number of trajectories according to the allowed degree of overlap. For instance, allowing a 25% overlap, we consider trajectories that overlap over $[0; T_a/4]$ as independent. In order to compare with a direct sampling approach, we could perform a very long simulation of computational cost $N_c \times T_a$, divide it into N_c realisations F_{T_a} and count the number of fluctuations $F'_{T_a} \geq a$ above a given threshold value a. However, we found that the PDF describing the statistics of fluctuations F'_{T_a}—with $T_a = 10\tau_c$— is very close to a Gaussian PDF. As a result, the expected number of fluctuations $F'_{T_a} \geq a$ among N_c realisations is

$$N_{N_c}^{(gauss)} \approx N_c \times \mathcal{P}(F'_{T_a} \geq a) = N_c \times \frac{1}{\sqrt{2\pi\sigma_{T_a}}} \int_a^\infty e^{-x^2/2\sigma_{T_a}^2} \mathrm{d}x. \quad (5.20)$$

Considering small fluctuations $F'_{T_a} \geq a$ with $a = \sigma$ and $a = 2\sigma$, one can see in table 5.1 that the number of sampled independent trajectories actually decreases as k is increased. This results from the fact that the overall number of independent trajectories decreases as k is increased, as illustrated in figure 5.8b. More importantly, considering fluctuations above 3σ, the GKTL experiments for $k = 0.025$ and $k = 0.03$ provide a larger number of fluctuations than a direct sampling approach with similar computational cost. For instance, table 5.1b shows that the experiment with $k = 0.025$ provides 5 fully independent trajectories corresponding to fluctuations $F'_{T_a} \geq 3\sigma$. This is ten times more than what can be expected from a direct sampling. Notice however how drastic is the reduction of the number of independent trajectories as the allowed degree of overlap is diminished. Allowing for a 50% overlap, the reconstructed ensemble for the experiment $k = 0.025$ contains roughly 200 independent trajectories corresponding to fluctuations $F'_{T_a} \geq 3\sigma$. Generally speaking, we observe in all three experiments that the number of independent trajectories is very small compared to the total number of trajectories $N_c = 16384$. For instance, table 5.1c shows that, allowing a 50% overlap between independent trajectories, 521 trajectories can be used to study the dynamics leading to fluctuations $F'_{T_a} \geq 3\sigma$. This represents $521/N_c \approx 3\%$ of the overall computational cost of the experiment. We stress that, in spite of this rather poor yield, the GKTL still results in a larger amount of extreme fluctuations, with respect to a direct sampling approach. However, there is clearly room from improvement.

5.3 DISCUSSION

In this chapter, the use of the GKTL algorithm was illustrated in two ways. First, the computation of the SCGF describing the large-time large deviations of the averaged drag was presented. More precisely, using the GKTL algorithm—following the implementation discussed in the previous chapter—we were able to compute the *large deviation*

rate function describing the probability of observing rare drag force fluctuations, far above the average drag level acting on the obstacle. We showed in figure 5.5 that for a similar computational cost, the GKTL algorithm allows for the estimation of the rate function over a wide range of fluctuations, *that could not be computed from a direct sampling approach*. However, we highlighted in figure 5.6b that the computational gain from the GKTL is limited by the number of copies. Indeed, we observed that, past a given value of k, the estimate of the SCGF becomes linear. Furthermore figure 5.6b illustrates that, for two different values of the number of copies, the slope of the linear tail of the SCGF is different, and that statistical errors do not overlap. As a result, we conclude that this linearization is an *systematic* artefact resulting from the algorithm.

In a second part, we addressed the practicality of the GKTL algorithm for the study of the *dynamics* corresponding to extreme fluctuations of the drag force acting on the obstacle embedded in test flow (2). As a matter of fact, the GKTL algorithm has seldom be used to study the dynamical aspects of rare events, as the majority of previous works using the algorithm mostly address the computation of statistical quantities, such as large deviation rate functions. However, in the context of the study of turbulent flows, the investigation of the dynamics of rare events is particularly useful to gain insight into the physical mechanisms leading to extreme fluctuations of the quantity of interest. We performed three GKTL experiments for different values of the bias k, with a high number of copies. First of all, we illustrated that the GKTL algorithm does sample a higher number of extreme events than a direct approach of similar computational cost. However, because trajectories in reconstructed ensemble overlap, the number of *actually independent* trajectories sampled by the algorithm is very low with respect to the total number of trajectories N_c. As a result, a very high number of copies is required to sample extreme events. This requirement can potentially hinder the application of the GKTL to the simulation of more computationally demanding flows.

In a way of conclusion, we find that the GKTL algorithm significantly increase the sampling of extreme fluctuations of the drag force acting on the obstacle immersed in test flow (2) . However, we observe that, for a fixed number of copies, there is a critical value of the bias k past which the majority of trajectories in the reconstructed ensemble overlap over a significant part of their history. Furthermore, the experiments discussed in this chapter suggest that a rather large number of copies is required to sample extremes using the GKTL algorithm. Is is not clear whether similar sampling performance can be achieved by performing several independent experiments with a reduced number of copies.

Generally speaking, we observe that the efficiency of the sampling of extreme drag fluctuations is limited in a way that have not been reported in previous applications such as [57, 130]. The precise reasons for these limitations is not clear yet.

In this work we considered the flow past a square cylinder in channel. Notice that this flow is convective, *i.e.* the fluid structures responsible for the extreme drag fluctuations are advected by a mean flow. In our application of the GKTL algorithm we based the sampling on the value of the drag acting on the square, that is depending on the flow configuration in the vicinity of the obstacle. However, because fluid structures are advected downstream, a high value of the drag does not last more than a few turnover times. In other words, a high value of the drag at a given instant t cannot be considered a precursor of high drag fluctuations in the future. By contrast, consider the case of an OU process defined as

$$\dot{x}(t) = -x(t) + \sqrt{2}\eta(t), \tag{5.21}$$

for which we showed in chapter 4 that the GKTL algorithm yields significant efficiency. Starting from a rare value $x^\star$, a copy can actually *persist* in taking rare values in the vicinity of $x^\star$ with suitable realisations of the noise $\eta(t)$. Trajectories with such realisations of the noise can be sampled by the GKTL algorithm.

By contrast, the phenomenology of the fluctuations of the drag acting on the obstacle immersed in test flow (2) is different. Indeed, there is not such idea of *persistence*. Starting from a state corresponding to an unusually large value of the drag $f_d^\star$, the system will rapidly relax towards common drag values, as the fluid structures responsible for the fluctuations will be swept by the mean flow.

These remarks pave the way to further improvement of the GKTL algorithm. For instance, an alternative approach could rely on a sampling based on the occurrence of flow structures at a given distance of the obstacle.

6

THE ADAPTIVE MULTILEVEL SPLITTING FOR THE SIMULATION OF EXTREME DRAG FLUCTUATIONS

So far in this thesis, rare events leading to extreme values for the drag around the obstacle have been sampled by means of importance sampling. It has been implemented at the level of the trajectories, in order to favour the occurrence of paths associated with an atypically high (or low) value of the average drag on the obstacle over the trajectory. The choice of the importance ratio was inspired from large deviation theory related to the asymptotics for the probability of the large-time averaged drag. We illustrated that the resulting biased measure could be numerically sampled using a population dynamics algorithm. Note that this case is rather exceptional: in general it is very difficult to sample numerically a good importance ratio that favours the rare events of interest.

The GKTL algorithm [57, 120] allows for the estimation of the statistics of extreme fluctuations of an *averaged* observable of the dynamics, thanks to large-deviation theory. Furthermore, the sampled trajectories also yield the detailed dynamics of the system leading to the large fluctuations of the observable. In addition to fluctuations of the time-averaged drag, fluctuations of the instantaneous drag certainly are of interest. That is, we would like to numerically sample flow trajectories over which the drag acting on the obstacle goes beyond a given threshold. However, there is no general result for the asymptotic form of the probability for the instantaneous drag. In this chapter we introduce the Adaptive Multilevel Splitting (AMS), a rare event sampling method that is not restricted to time-averaged observables.

An alternative approach to importance sampling is *multilevel splitting*. It does not rely on sampling a modified distribution, but instead in decomposing a rare event into a sequence of consecutive events. As such, intermediate events are much less rare, conditioned on the realisation of the previous events in the sequence.

To clarify this idea, let us consider the example in figure 6.1. Let X_t be a Markovian $\mathbb{R}$-valued stochastic process with typical value x_0, and $a \geq 0$ a given threshold for which a fluctuation $X \geq a$ is very rare. Say we are interested in computing the probability $q = \mathcal{P}_{x_0}(X \geq a)$ that the random process is larger than a, starting from the initial condition x_0. A multilevel splitting approach consists in decomposing the event $X \geq a$ into a sequence of J events $X \geq Q_j$ with

$$x_0 = Q_1 < Q_2 < ... < Q_{J-1} < Q_J = a$$

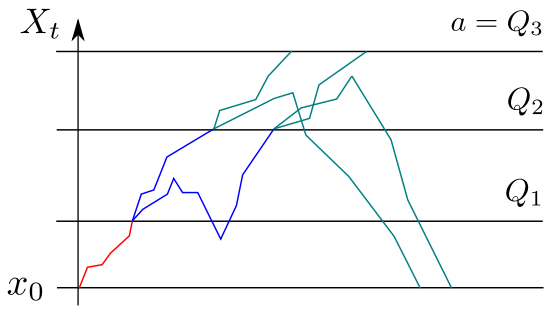


Figure 6.1: Illustration of the *multilevel splitting* strategy. From initial condition x_0, there is a very little probability that the process X_t will reach the level a before falling back to the typical region around x_0. A straightforward way of computing such a probability is to compute many realisations and count how many succeed to reach a. Naturally, as the event $X \geq a$ is very rare, this would require a tremendous amount of realisations. Instead, splitting methods suggest to decompose the fluctuation into a finite number of intermediate fluctuations. First $X \geq Q_1$, then $X \geq Q_2 | X \geq Q_1$ and finally $X \geq Q_3 | X \geq Q_2$, where $A|B$ denote the realisation of A conditioned on the realisation of B. In this way, each intermediate event is much more likely to occur and its probability easier to compute. Generally, the process is duplicated upon reaching a given *level* Q_i. The choice of the levels and the number of replicas depends on the specific splitting method in use.

and to compute p as

$$\mathbb{P}(X \geq a) = \mathbb{P}(X \geq Q_1) \prod_{j=1}^{J-1} \mathbb{P}(X \geq Q_{j+1} | X \geq Q_j) \tag{6.1}$$

In this way, each intermediate conditional probability is expected to be much greater than the target probability $q = \mathcal{P}(X \geq a)$. The splitting approach is sketched in figure 6.1. Similarly to the GKTL algorithm, the numerical computation of each conditional probabilities relies on the simulation of an ensemble of copies of system. Copies that successfully reach a given threshold are duplicated, while the others are discarded from the ensemble.

The original idea of splitting can be traced back to Kahn and Harris in 1951 [73]. It was somewhat forgotten until the 90s when it underwent a revival initiated by the publication of the RESTART method, to compute probabilities of failure in communication systems [171–173]. The main incentive for the application of splitting methods in this field was the apparent simplicity compared to importance sampling, for which it often proves difficult to sample a modified probability. The multilevel splitting strategy was first studied theoretically, in a simplified framework, by Glasserman *et al* [60, 61] and a few years later by Lagnoux [95]. Both works result in expressions for the variance of the estimator for the target probability depending on the choice

of the thresholds $\{Q_j\}_{1\leq j\leq J}$, and the number of duplicates that are spawned at each steps.

However, the biggest drawback of multilevel splitting is that good knowledge of the system is required to set the thresholds and the number of duplicates in order to reduce the variance. In its original form, multilevel splitting is therefore unsuitable for complex systems, for which analytical computations cannot be carried out. In response to this, Cerou and Guyader proposed a splitting algorithm [27] in which the thresholds are not set beforehand, but rather set in an *adaptive* manner over the course of the algorithm. As a consequence, the method is referred to as Adaptive Multilevel Splitting (AMS) and is now the state-of-the-art approach for rare event sampling by means of splitting.

The objective of the work presented in this chapter is to assess the efficiency and practicality of the AMS for rare event sampling in the context of turbulent flows. More precisely, we address the sampling of trajectories resulting in extreme fluctuations of the *instantaneous* drag f_d acting on an obstacle surrounded by a turbulent flow. In fact, we will use a modified version of the AMS: the Trajectory Adaptive Multilevel Splitting (TAMS). We developed this variant in order to compute the probability that a trajectory of fixed duration results in a observable of the dynamics going beyond a given level. As will be discussed in chapter 7, this probability is useful to compute return times of rare events, that is the typical timescale at which they occur. A return time is a very useful quantity for applications.

In this chapter we do not compute return times but rather apply the TAMS with the aim of generating rare trajectories of the flow resulting in a very large drag. This study will be based in test flow (2) presented in chapter 2.

The structure of this chapter is as follows. First, the AMS is presented in section 6.1. We outline the algorithm and describe the main theoretical results concerning the resulting estimator for the probability of the rare event of interest. The TAMS is then discussed in section 6.2.1. We first give a practical outline of the algorithm in section 6.2.1, before we highlight its connection with the classical AMS in section 6.2.2. The TAMS is illustrated on a simple on case in section 6.3. We show that for a one-dimensional stochastic process the algorithm leads to a tremendous improvement of the sampling of rare fluctuations. In section 6.4, we present the application of the TAMS to the sampling of extreme drag fluctuations in test flow (2). We show that the sampling suffers from a naive definition of the *score function*, that quantifies the performance of the replicas with respect to the realisation of the fluctuation.

6.1 THE AMS ALGORITHM

The Adaptive Multilevel Splitting (AMS) algorithm has originally been designed [27] to efficiently and accurately estimate probabilities of rare events of the type $\mathbb{P}_{x_0,t_0}(\tau_{\mathcal{B}} < \tau_{\mathcal{A}}) \in (0,1)$: the probability that a Markov process $(X_t)_{t \geq t_0}$, initialised with $X_{t_0} = x_0$, hits a set $\mathcal{B}$ before hitting a set $\mathcal{A}$ (with $\mathcal{A} \cap \mathcal{B} = \emptyset$), where $\tau_{\mathcal{C}} = \inf\{t > t_0; X_t \in \mathcal{C}\}$ is the hitting time of a set $\mathcal{C}$. In typical applications sets $\mathcal{A}$ and $\mathcal{B}$ are metastable states, from which the dynamics can transition to the other one under the effect of random perturbations.

In addition to the sets $\mathcal{A}$ and $\mathcal{B}$, a crucial ingredient of the AMS algorithm is a *score function* ξ that quantifies the distance from a state of the system to either $\mathcal{A}$ or $\mathcal{B}$. In many application, it is designed so that $\xi(x) = 0$ if $x \in \mathcal{A}$ and $\xi(x) = 1$ if $x \in \mathcal{B}$.

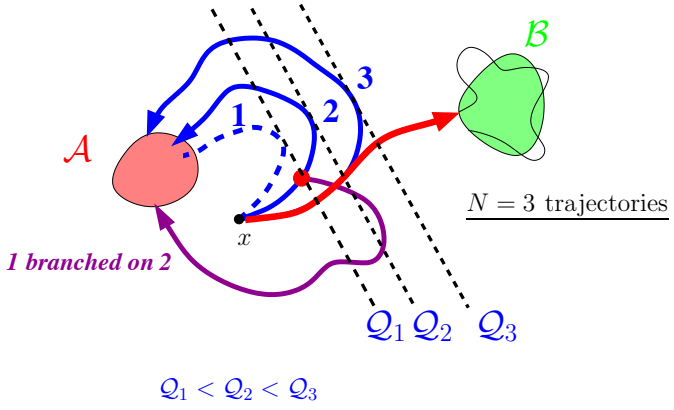


Figure 6.2: Illustration of an iteration of the AMS algorithm. In this sketch, three trajectories are computed from the initial condition x located within the basin of attraction of $\mathcal{A}$. They are computed until they either reach $\mathcal{B}$ or fall to $\mathcal{A}$. The levels $\{\mathcal{Q}\}_{1 \leq i \leq 3}$ correspond to the maximum value of the score function $\xi(x)$ over the corresponding trajectory. They quantify how close each trajectory got from $\mathcal{B}$. Having the lowest level among the ensemble, trajectory 1 (dashed) is discarded from the ensemble of trajectories. Trajectory 2 is then randomly selected and copied until it reaches the maximum of 1. This point is referred to as the *branching point* and is represented by a red dot in the figure. From there, dynamics are integrated until it either reach $\mathcal{A}$ or $\mathcal{B}$.

The AMS algorithm is illustrated in figure 6.2 and can be outlined as follows. First, N_c trajectories are simulated starting from $x = x_0$ with x_0 located in the basin of attraction of $\mathcal{A}$. Each trajectory is computed independently from the others until it either reaches $\mathcal{B}$ or falls back to $\mathcal{A}$. Next, the performance of the trajectories is assessed by computing their *level* Q_n, that is the maximum of the score function over the

trajectory, see figure 6.2. Then, we compute the lowest level $Q^\star$ among the trajectories:

$$Q^\star = \min_{1 \leq 1 \leq N_c} Q_n \tag{6.2}$$

The corresponding trajectory is discarded from the ensemble. It can be seen as the copy that went the least far away from $\mathcal{A}$. Then, a trajectory is uniformly chosen among the $N_c - 1$ remaining and duplicated until the point where it goes beyond Q. This is referred to as the *branching point* and is identified by a red dot in figure 6.2. In practice the discarded trajectory is overwritten by the selected trajectory until it reaches Q. From that point on, it is freely simulated until it reaches $\mathcal{B}$ or falls back to $\mathcal{A}$. This procedure is referred to as a *selection-mutation* step.

In typical AMS applications, it is repeated until the N_c trajectories reach $\mathcal{B}$. If it is repeated J times, the AMS estimator $\hat{q}$ for $q = \mathbb{P}_{x_0,t_0}(\tau_\mathcal{B} < \tau_\mathcal{A})$ is [27]

$$\hat{q}_{N_c} = \left(1 - \frac{1}{N_c}\right)^J \tag{6.3}$$

One of the main properties of the AMS algorithm is the following unbiasedness result. For every N_c, for every score function ξ, $\hat{p}_{N_c}$ is an unbiased estimator of q:

$$\mathbb{E}[\hat{p}_{N_c}] = p. \tag{6.4}$$

Thus only the statistical error $\mathrm{Var}(\hat{p}_{N_c})$ depends on the choice of N_c. For a proof, as well as more general statements and a discussion on the influence of the time discretisation of the Markov dynamics, see [17, 18]. This result has an important practical consequence [17] : instead of performing a single AMS experiment with M copies, one can perform K independent realisations with $N_c = M/K$ copies and compute the resulting estimator as an empirical mean over of the realisations. Because their are independent, the K realisations can be performed in parallel. Additionally, the choice of the score function can be shown to have an important impact on the statistical error, see [17, 141]. Moreover, it has been proved, in different contexts, see [19, 28], that $\hat{q}_{N_c}$ is a consistent estimator of q: the convergence $\hat{q}_{N_c} \underset{N_c \to \infty}{\to} q$ holds true, in probability. More precisely, it is proved in [28], that the estimator $\hat{q}_{N_c}$ satisfies a Central Limit Theorem,

$$\sqrt{N_c}(\hat{q}_{N_c} - q) \underset{N_c \to \infty}{\to} \mathcal{N}(0, \sigma^2(\xi, q)), \tag{6.5}$$

with an asymptotic variance $\sigma^2(\xi, q) \in [-q^2 \ln q, 2q(1-q)]$. The minimal variance $-q^2 \ln q$ is obtained when choosing

$$\xi(x) = \overline{\xi}(x) \equiv \mathbb{P}_{x,t_0}(\tau_{\mathcal{B}} < \tau_{\mathcal{A}}). \tag{6.6}$$

The optimal score function $\overline{\xi}$, is referred to as the *committor* function. In practice, it is of course not known, as it is what the algorithm aims at estimating: $p = \xi(x_0)$. Nevertheless, a crucial point to implement the AMS algorithm is to choose a score function that provides a good approximation of the committor.

The AMS was first applied in the probability and statistical physics communities to rather simple systems, the subject of investigation being the algorithm itself more than the dynamics[17, 27, 141]. In 2015, the AMS was applied to compute reactive trajectories of the stochastic 1D Allen-Cahn equations, showing that the AMS is able to provide good results for stochastic dynamics with a large number of degrees of freedom [139]. Recently, the AMS was introduced in the fields of biophysics and biochemistry. It was coupled with Molecular Dynamics (MD) simulations and was reported to yield satisfying results both for a simple test case [5] and the simulation of a more complex biochemical process: the dissociation of a protein-ligand complex [161]. In fluid mechanics, the AMS was used to investigate multi-stability in stochastic models of transitional wall flows [138]. The common point between these works is that they all aim at computing transition pathways between meta-stable states. Here we are instead interested in computing the probability that a trajectory with a *fixed* duration T_a goes beyond a given level. This is motivated by the computation of return times, discussed in chapter 7. To do so, we introduce a modified AMS algorithm, called the Trajectory Adaptive Multilevel Splitting (TAMS) algorithm, presented next.

6.2 THE TRAJECTORY ADAPTIVE MULTILEVEL SPLITTING ALGORITHM

In the previous section, we presented the AMS algorithm. It introduces a modified sampling based on the splitting of trajectories according to a score function that quantifies the performance of the trajectories with respect to realisation of a rare event. The AMS eventually yields the probability that trajectory reaches a target set $\mathcal{B}$ before a set $\mathcal{A}$, which we denoted by $\mathbb{P}(\tau_{\mathcal{B}} < \tau_{\mathcal{A}})$. Additionally, it yields an ensemble of trajectories that effectively reach the target $\mathcal{B}$. Note that these trajectories have different durations. The AMS was originally designed in order to compute rare transition rates between metastable states. In the following we are interested in rare events in which a given observable reaches a threshold a. In this context a natural quantity is the probability $\mathbb{P}(\tau_{\mathcal{B}} < T_a)$ that the fluctuation is achieved before

a finite time T_a. For instance it is useful to compute return times of rare events, that is the average waiting time for a the realisation of a given event. In chapter 7, we show that return times can be easily deduced from the probability $\mathbb{P}(\tau_{\mathcal{B}} < T_a)$. However, this probability is different from the one that is computed by the AMS, and additional computations are required to deduce $\mathbb{P}(\tau_{\mathcal{B}} < T_a)$ from $\mathbb{P}(\tau_{\mathcal{B}} < \tau_{\mathcal{A}})$.

In this section we propose a novel version of the AMS, that yields directly the probability $\mathbb{P}(\tau_{\mathcal{B}} < T_a)$. We call this variant the Trajectory Adaptive Multilevel Splitting (TAMS). It is a special case of the AMS. In the TAMS, trajectories all have the same duration T_a. This greatly simplifies the algorithm as well as its application.

We start by outlining the algorithm in section 6.2.1. Then, in section 6.2.2 we discuss its connection with the original AMS algorithm. We show that the TAMS can be expressed within the original AMS framework by defining the sets $\mathcal{A}$ and $\mathcal{B}$ for an auxiliary process. This allows the TAMS to directly benefit from the mathematical properties of the AMS presented in section 6.1.

6.2.1 *Description of the TAMS algorithm*

We consider a continuous time Markov model $x(t)$ able to generate trajectories. It can be either a stochastic process, for instance a diffusion, or a chaotic deterministic dynamical system. Let us now describe the algorithmic procedure.

We start by simulating N_c independent trajectories, denoted by $\{x_n^{(0)}(t)\}_{1\leq n\leq N_c}$, for a fixed duration T_a. To each of these trajectories, we associate a weight $w_0 = 1$. The iteration $j \geq 1$ of the algorithm starts with the evaluation of the performance of all replicas $\{x_n^{(j-1)}(t)\}_{1\leq n\leq N_c}$ at the previous iteration $j-1$, measured by the maximum of the score function ξ over the whole trajectory:

$$\mathcal{Q}_n^{(j)} = \sup_{0\leq t\leq T_a} \xi(t, x_n^{(j-1)}(t)). \tag{6.7}$$

We select the trajectories corresponding to the lowest $\mathcal{Q}_n^{(j)}$: let us denote $\mathcal{Q}_j^\star = \min_{1\leq n\leq N_c} \mathcal{Q}_n^{(j)}$ and $n_{j,1}^\star, \ldots, n_{j,\ell_j}^\star$ the indices such that:

$$\mathcal{Q}_{n_{j,1}^\star}^{(j)} = \cdots = \mathcal{Q}_{n_{j,\ell_j}^\star}^{(j)} = \mathcal{Q}_j^\star. \tag{6.8}$$

One might expect intuitively that $\ell_j = 1$. This is not necessarily the case, as explained in [17]: because of the discretisation of the dynamical equations in the numerical model, two or more trajectories may yield the same level $\mathcal{Q}_n^{(j)}$. This effect is illustrated in figure 6.4.

We then proceed to the mutation step. For each trajectory $x_{n_{j,\ell}^\star}^{(j-1)}$ $(1 \leq \ell \leq \ell_j)$, we choose a trajectory $x_{n_\ell}^{(j-1)}$ $(n_\ell \neq n_{j,1}, \ldots n_{j,\ell_j})$ randomly

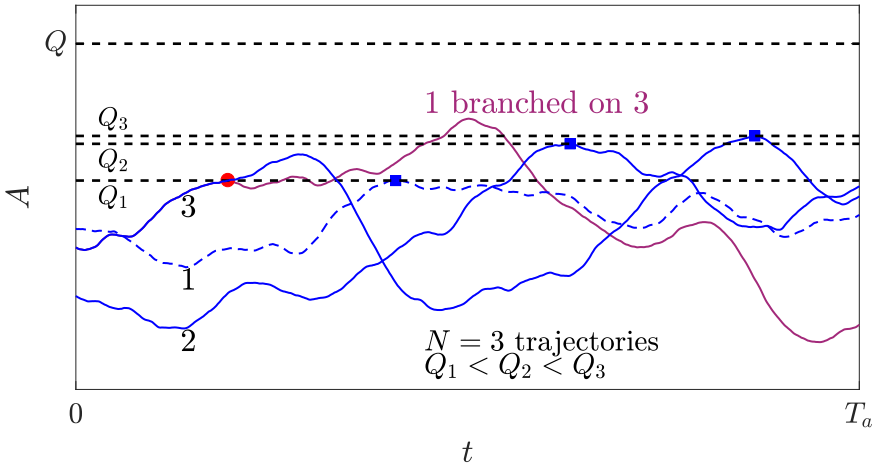


Figure 6.3: Illustration of one selection-mutation step in the AMS algorithm for the computation of the probability that an observable $A : \mathbb{R}^d \to \mathbb{R}$ reaches values larger than Q over a trajectory of duration T_a.

among the $N_c - \ell_j$ remaining trajectories, and define the time $t_{j,\ell}$ defined as the smallest time t such that $\xi(t, x_{n_\ell}^{(j-1)}(t)) > \mathcal{Q}_j^\star$. Finally, we define the new replica $x_{n_{j,\ell}^\star}^{(j)}$ by copying the trajectory $x_{n_\ell}^{(j-1)}$ from t_0 to $t_{j,\ell}$, and simulating the rest of the trajectory, from $t_{j,\ell}$ to T_a. For a Markov process, for instance a diffusion, a new realisation of the noise is used in order to simulate the new trajectory from t_j to T_a. For a chaotic deterministic system, a small amplitude noise is added to the initial condition at time t_j. The other trajectories are not modified: $x_n^{(j)} = x_n^{(j-1)}$ for $n \neq n_{j,1}^\star, \ldots, n_{j,\ell}^\star$. The selection-mutation process is illustrated on figure 6.3. We associate to the trajectories $x_n^{(j)}$ forming the ensemble at step j the weight w_j given by [17, 27, 29]:

$$w_j = \prod_{i=1}^{j} \left(1 - \frac{\ell_i}{N_c}\right) = \left(1 - \frac{\ell_j}{N_c}\right) w_{j-1}. \tag{6.9}$$

Note that we could mutate more replicas at each step by selecting an arbitrary number of levels $\mathcal{Q}_n^{(j)}$, instead of just the minimum $\mathcal{Q}_j^\star$ as described above. The particular case described above is sometimes referred to as the *last particle method* [151].

The selection-mutation process is iterated J times (two possible definitions of J are given below). The number of resampled trajectories is given by $\tilde{J} = \sum_{j=1}^{J} \ell_j$. Note that $\tilde{J} \geq J$, but the two need not necessarily coincide. In the end, the algorithm generates $M = N_c + \tilde{J}$ trajectories, given explicitly by the set $\{x_n^{(0)}\}_{1 \leq n \leq N_c} \cup \{x_{n_{j,\ell}^\star}^{(j)}\}_{1 \leq \ell \leq \ell_j, 1 \leq j \leq J}$, or equivalently, the set $\{x_n^{(J)}\}_{1 \leq n \leq N_c} \cup \{x_{n_{j,\ell}^\star}^{(j-1)}\}_{1 \leq \ell \leq \ell_j, 1 \leq j \leq J}$. Each trajectory has an associated weight, given by the iteration until which it was

a member of the ensemble: w_J for the final trajectories $\{x_n^{(J)}\}_{1\leq n\leq N_c}$, and w_{j-1} for the trajectories $\{x_{n_{j,\ell}^\star}^{(j-1)}\}_{1\leq \ell\leq \ell_j, 1\leq j\leq J}$ mutated at iteration $1 \leq j \leq J$. Let us relabel these trajectories and their associated weights as $\{(x_m, w_m)\}_{1\leq m\leq M}$. Normalising the weight with $W = \sum_{m=1}^{M} w_m$, we obtain the probabilities $p_m = w_m/W$ associated with the trajectories.

Note that instead of just one realisation of the algorithm, one may carry out K independent realisations, thus yielding $M = \sum_{k=1}^{K}(N_c^{(k)} + \tilde{J}_k)$ trajectories with the associated weights, where $N_c^{(k)}$ and $\tilde{J}_k$ denote the number of initial trajectories and resampled trajectories for realisation k, respectively. The probabilities for the trajectories are computed following equation (6.3).

In addition, for any observable $O[x(t)]$, we can define an estimator based on our sampling of trajectory space:

$$\hat{O}_M = \sum_{m=1}^{M} p_m O[x_m(t)]. \tag{6.10}$$

6.2.1.1 *Computational cost of an TAMS run*

The number of iterations J can be set to be a prescribed integer. In that case, the stopping criterion for the algorithm is simply $j = J$. Alternatively, it can be a random number such that all the trajectories in the ensemble reach a threshold level $\mathcal{Q}$. The stopping criterion is then $\mathcal{Q}_n^{(j)} > \mathcal{Q}$ for all $1 \leq n \leq N_c$. The latter case is more common in existing AMS implementations, however both cases are covered by the general framework developed in [17], and give consistent results. The two possible choices are further discussed in the case of the application of the TAMS for the computation of return times in chapter 7.

Let us now estimate the computational cost of a TAMS run. The number of trajectories generated by a TAMS run is $M = N_c + \tilde{J}$, as pointed out above. Each resampled trajectory is not simulated over the whole duration T_a, but over $\tau < T_a$, with τ a random number depending on the branching point. We thus define $\gamma \in [0, 1]$ so that $\mathbb{E}[\tau] = \gamma T_a$ is the average duration of the resampled part of a mutated trajectory. Performing K identical and independent realisations of the TAMS algorithm, the average computational cost associated with a given experiment is then approximately

$$\mathcal{C} = K \times (N_c + \gamma J) T_a. \tag{6.11}$$

6.2.1.2 *Extinction*

As mentioned in [17], and illustrated in figure 6.4, several trajectories can yield the same level, due to the discreteness of the numerical model. As a matter of fact, both the parent and the resampled trajectory are likely to end up with the same level as the threshold $Q_j^\star$ increases. Indeed, for high thresholds, the resampled trajectory is very

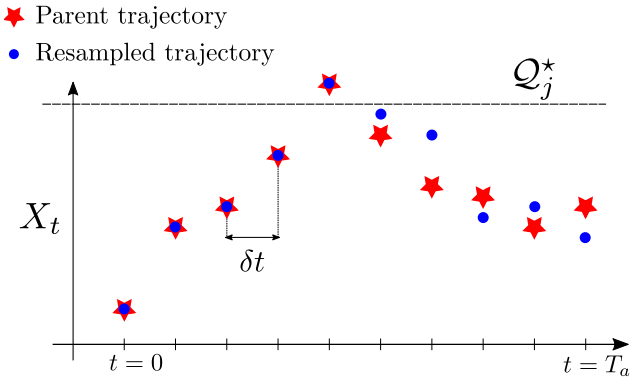


Figure 6.4: Sketch of a resampling step resulting in two trajectories having the same level. At iteration j, the resampled trajectory is overwritten by the parent trajectory until the first point for which it is above the threshold $\mathcal{Q}_j^\star$. From this point, the resampled trajectory quickly falls back below the threshold. However, from the point of view of the discretised dynamics, the blue and red trajectories now have the same level. Both must therefore be resampled at iteration $j+1$.

likely to fall back right after the branching point. If this branching point is the maximum of the parent trajectory, as it is likely the case for high values of $Q_j^\star$, then both trajectories must be resampled at the iteration $j+1$. This situation is illustrated in figure 6.4. If the threshold $Q_j^\star$ is very high, there is a good chance that the resampling of the two trajectories will fail in the same way, resulting in the resampling of three trajectories, and so on and so forth. In the event that the resampling fails N_c times, no parent trajectory is available, and the algorithm stops. This is referred to as *extinction*. The exact critical value of $Q_j^\star$ for which extinction happens varies from one realisation of the TAMS to the other. However, its order of magnitude depends on the number of trajectories as well as their duration.

6.2.2 *Connection with the AMS for time-dependent observables*

In this section, we describe the connection between the Trajectory Adaptive Multilevel Splitting (TAMS) algorithm and the classical AMS algorithm. The aim is to deduce the mathematical properties of the TAMS algorithm from the known ones for the AMS algorithm. For instance, a conclusion is that the optimal score function is the committor function (6.14).

As stated in section 6.1, the Adaptive Multilevel Splitting (AMS) algorithm was originally designed [27] to estimate probabilities of rare events of the form $\mathbb{P}_{x_0,t_0}(\tau_\mathcal{B} < \tau_\mathcal{A})$. In this section, we show how the problem of estimating the probability that the maximum value of a time-dependent observable over a trajectory is higher than a given threshold falls within the scope of the AMS algorithm. In this way, the

TAMS algorithm directly benefits from the theoretical properties of the AMS algorithm outlined in section 6.1.

We consider a $\mathbb{R}^d$-valued Markov process $(X_t)_{t\in[0,T_a]}$, with continuous trajectories, for some fixed final time T_a, and a time-dependent observable $O[X,t]$: this is a time-dependent functional of the process X, taking value in $\mathbb{R}$. It may be defined for times belonging to a subset of $[0,T_a]$, but for simplicity we shall still denote T_a the final time. The aim is to estimate the probability that the observable reaches a threshold a at some point of the trajectory, i.e.

$$q(a) = \mathbb{P}_{x_0,0}\left[\max_{0\le t\le T_a} O[X,t] > a\right]. \tag{6.12}$$

The notation $\mathbb{P}_{x_0,t_0}$ refers to the probability over realisations of the Markov process with initial condition $X_{t_0} = x_0$. The AMS algorithm provides an estimator $\hat{q}(a)$ for this quantity. Indeed, the event

$$\left\{\max_{0\le t\le T_a} O[X,t] > a\right\}$$

can be identified with the event $\{\tau_{\mathcal{B}} < \tau_{\mathcal{A}}\}$ for an auxiliary Markov process Y_t, with an appropriate definition of the sets $\mathcal{A}$ and $\mathcal{B}$, as follows:

$$\begin{aligned} Y_t &= (t, O[X,t]) \in [0,T_a]\times\mathbb{R}, \\ \mathcal{A} &= \{(T_a,z);\ z\le a\}, \\ \mathcal{B} &= \{(t,z);\ t\in[0,T_a], z > a\}. \end{aligned} \tag{6.13}$$

Note that Y is not necessarily a time-homogeneous process. In section 6.2.1, we described the TAMS algorithm that gives a procedure to sample the process Y to provide a good estimate of $q(a)$, based on a score function ξ, which measures the distance between $\mathcal{A}$ and $\mathcal{B}$ (in many implementations of the AMS, $\xi(\partial\mathcal{A}) = 0$ and $\xi(\partial\mathcal{B}) = 1$). In the specific case of the TAMS algorithm, the optimal score function (the committor $\mathbb{P}[\tau_{\mathcal{B}} < \tau_{\mathcal{A}}]$) depends on time and takes the form:

$$\bar{\xi}(t,x;T_a,a) = \mathbb{P}_{x,t}\left[\max_{t\le s\le T_a} O[X,s] > a\right], \tag{6.14}$$

The corresponding estimator $\hat{q}(a)$ will be very useful to compute return times of rare events, and is described in chapter 7, section 7.3.1.

It then follows from the above discussion that the convergence properties of the TAMS algorithm are a direct consequence of the known results for the AMS algorithm, described in section 6.1.

6.3 APPLICATION OF THE TAMS TO THE ORNSTEIN–ULHENBECK PROCESS

In the previous section, we introduced a modified version of the Adaptive Multilevel Splitting (AMS) algorithm, referred to as the Trajectory Adaptive Multilevel Splitting (TAMS) algorithm. Its definition is motivated by the estimation of probabilities of trajectories with a fixed duration T_a, from which *return times* of extreme fluctuations can easily be deduced. The computation of return times is discussed in chapter 7.

In this section, we illustrate the TAMS algorithm by addressing the sampling of extreme fluctuations of a random variable x following an Ornstein–Ulhenbeck (OU) process:

$$\dot{x}(t) = -\alpha x(t) + \sqrt{2\epsilon}\eta(t) \tag{6.15}$$

where α and ϵ are numeric constants and η a Gaussian white noise. The Probability Density Function (PDF) describing the fluctuations of x can be shown to be Gaussian, with a standard deviation $\sigma = \sqrt{\epsilon/\alpha}$. In this case, using trajectories of length T_a, the TAMS algorithm ultimately yields the probability that x reaches a threshold a at some point of the trajectory:

$$q(a) = \mathbb{P}_{x_0}\left[\max_{0\leq t\leq T_a} x(t) > a\right] \tag{6.16}$$

where $\mathbb{P}_{x_0}$ indicates the probability over realisations of the OU process with initial condition $X_{t_0} = x_0$. In the following x_0 is always drawn from the stationary distribution of the process. Furthermore, the TAMS also yields an ensemble of $M = N_c + \tilde{J}$ trajectories where N_c in the number of initial trajectories and $\tilde{J}$ the number of resampled trajectories. Recall that the last N_c resampled trajectories all correspond to fluctuations $x > a$. We choose the score function ξ as the observable itself: $\xi \equiv x$. This is motivated by the fact that the dynamics is one-dimensional. See appendix D for further discussion of the choice of the score function. In the following we show that the TAMS is capable of sampling very rare trajectories with a computational cost much lower than the one required by a direct sapling approach. To do so, we discuss the computational cost associated with the sampling of fluctuations of a given amplitude with the TAMS algorithm. We compare it with the typical return time of such a fluctuation and illustrate that the computational gain induced by the TAMS with respect to direct sampling grows rapidly with the amplitude of the fluctuation. We then illustrate the estimation of the probability $q(a)$ and show that the TAMS algorithm is capable of computing accurate estimates of the probability for very rare fluctuations.

6.3.1 *Efficient sampling of very rare trajectories*

The TAMS is described by two parameters: the number of initial trajectories N_c and their duration T_a. As mentioned in section 6.1, the AMS estimator for the probability $q(a)$ is unbiased, *i.e.*, $\mathbb{E}[\hat{q}(a)] = q(a)$. As a consequence, it was suggested in [17] that a good practice is to use a relatively low number of initial trajectories, and perform several independent realisations of the algorithm. In the following we set $N_c = 32$.

The duration of the trajectories T_a must be set larger than the correlation time τ_c of the process, so that an excursion far from the typical value has sufficient time to occur. On the other hand, setting $T_a \gg \tau_c$ may not provide a significant advantage. Indeed, resampling long trajectories can lead to new fluctuations, however decorrelated from the one the resampling is based on. These fluctuations would therefore occur independently from the TAMS, a situation that is tantamount to direct sampling. In the following we set $T_a = 5\tau_c$.

The efficiency of the TAMS algorithm is analysed as follows. To each iteration j of the algorithm, we associate a computational cost C_j corresponding to the *cumulative duration* of the resampled trajectories for iterations $1 \leq l \leq j$. It reads

$$C_j = (N_c + \sum_{l=1}^{j} \sum_{m=1}^{l_j} \alpha_{lm}) \times T_a \tag{6.17}$$

where l_j is the number of resampled trajectories at iteration j and $\alpha_{lm} T_a$ is the duration over which the resampled trajectory is effectively simulated. Recall that for $0 \leq t \leq \alpha_{lm} T_a$, it is simply copied from its parent trajectory. The computational cost C_j corresponds the the overall computational effort spent computing the dynamics up to iteration j.

Following the notations of section 6.2.1, let $\mathcal{Q}_j^\star$ be the level at iteration j, that is the maximum value of $x(t)$ over the selected trajectory *before* it is resampled. Recall that, by definition, $\mathcal{Q}_n^{(j)} \geq \mathcal{Q}_j^\star$, $1 \leq n \leq N_c$, where $\mathcal{Q}_n^{(j)}$ is the maximum of $x(t)$ over the trajectory n at iteration j.

At iteration j, each member of the ensemble of N_c trajectories displays a fluctuation of amplitude $a \geq \mathcal{Q}_j^\star$. This ensemble of trajectories has been computed with a cost C_j. In the following, we compare C_j to the typical computational cost required to sample fluctuations $a \geq \mathcal{Q}_j^\star$ from a direct simulation of the process (6.15), without algorithm. Roughly speaking, this cost corresponds to the typical *return time* of the fluctuations. That is the timescale of occurrence of fluctuations having an amplitude at least equal to $\mathcal{Q}_j^\star$. Proper definition and sampling of return times is addressed in chapter 7. The efficiency of the algorithm can be quantified by comparing the cost of the TAMS algo-

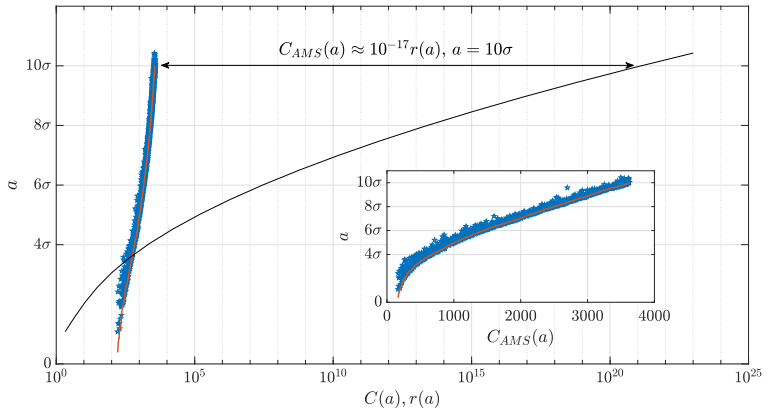


Figure 6.5: Illustration of the efficiency of the TAMS algorithm with respect to direct sampling. The amplitude of fluctuations for the OU process (6.15) are represented along the Y-axis. Along the X-axis is represented the typical computational cost required to sample fluctuations of the corresponding amplitude in both the TAMS and direct sampling. In the latter, the typical computational cost is simply the return time $r(a)$. The computational cost associated to a level a is the TAMS is denoted by $C(a)$. The orange line represents the evolution of the level $\mathcal{Q}_j^\star$ as a function of the cost $C(a)$. The blue stars represent the maximum of $x(t)$ over the resampled trajectory *after* it is resampled, that is at iteration $j+1$. Finally, the solid black line is the analytical solution for the return time of amplitude a [105].

rithm at iteration j to the return time of the corresponding fluctuations $\mathcal{Q}_j^\star$.

We perform a numerical experiment with the TAMS algorithm with $N_c = 32$ and $T_a = 5$. The initial condition for the N_c initial trajectories is drawn from the stationary PDF of the process (6.15). The stopping criterion for the algorithm is chosen based on the value of the level: $\mathcal{Q}_j^\star \geq \mathcal{Q}$ with $\mathcal{Q} = 10\sigma$. At each iteration j we record the current level $\mathcal{Q}_j^\star$. Additionally, we record the simulation time of the resampled trajectories $\alpha_{lm} T_a$ in order to compute the computational cost C_j following (6.11). In the following we drop the dependence in the iteration j and simply call $C(a)$ the computational cost associated to a level $\mathcal{Q}_j^\star = a$ in the TAMS.

Figure 6.5 displays the fluctuation amplitude a as a function of the computational cost $C(a)$. In addition, it also features the corresponding return time $r(a)$, that can be computed analytically [105]. It clearly illustrates the gain from the TAMS algorithm: from $a \geq 4\sigma$, extreme fluctuations are sampled within the TAMS at a much lower computational cost. As an example, figure 6.5 shows that the typical return time of fluctuations $a \geq 10\sigma$ is roughly $10^{21}\tau_c$. This means that, in order to sample typically one of such fluctuations, the process must be simulated over a duration of the order of $10^{21}\tau_c$. In contrast,

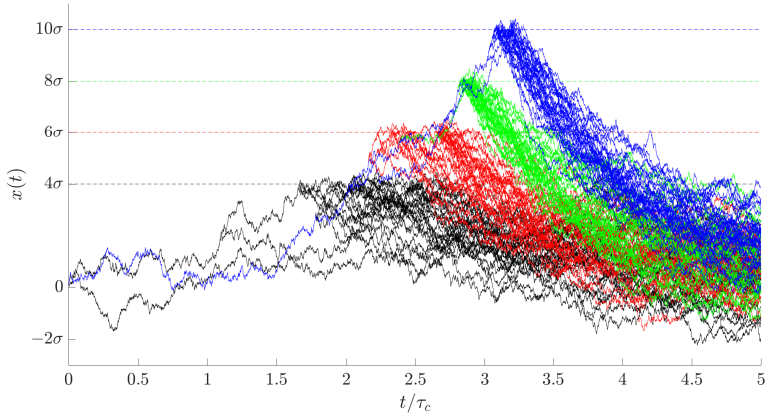


Figure 6.6: Ensemble of $N_c = 32$ trajectories at four different iterations j, for which $\mathcal{Q}_j^\star > a$ with $a = 4\sigma, 6\sigma, 8\sigma$ and 10σ.

fluctuations $a \geq 10\sigma$ are sampled by the TAMS for an overall cost of roughly $5 \times 10^3 \tau_c$.

Figure 6.6 illustrates the ensemble of $N_c = 32$ trajectories at the first iterations j for which $\mathcal{Q}_j^\star > a$ with $a = 4\sigma, 6\sigma, 8\sigma$ and 10σ.

6.3.2 *Estimation of the probabilities of rare fluctuations*

In the previous section, we illustrated that the TAMS successfully samples very rare fluctuations of the OU process defined in (6.15). We showed that it allows for the simulation of the dynamics leading to such fluctuations for a computational cost much smaller than the one required by a direct simulation of the process. In the following, we briefly illustrate the estimation of the probability of such fluctuations.

Following the strategy suggested in [17], we divide the computation into K independent realisations. Each one of them correspond to a TAMS run in which the initial trajectories are drawn in the stationary PDF of the OU process (6.15). Consistently with the experiment described above, these runs are based on $N_c = 32$ trajectories simulated over a duration $T_a = 5\tau_c$. Each one of them is iterated until all trajectories reach a prescribed threshold a, that is until the first iteration j for which $\mathcal{Q}_j^\star > a$. This therefore results in a set of $\{\hat{q}_k(a)\}_{1 \leq k \leq K}$ of realisations of the estimate of the probability $q(a)$ that the process reaches a at some point over $[0; T_a]$. The final estimate is then computed as an empirical mean over the realisations:

$$\hat{q} = \frac{1}{K} \sum_{k=1}^{K} \hat{q}_k \tag{6.18}$$

Figure 6.7 illustrates the convergence of the empirical mean (6.18) as a function of the number of independent realisations K. In addition, it displays the confidence interval $[\hat{q}_K - \Delta\hat{q}_K; \hat{q}_K - \Delta\hat{q}_K]$ associated with

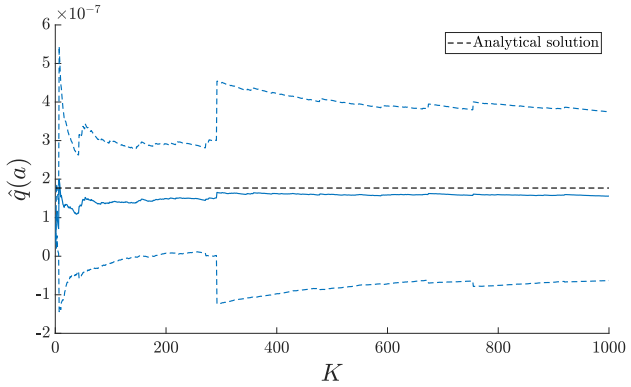


Figure 6.7: Convergence of the TAMS estimate $\hat{q}_K(a)$ as a function of the number of realisations of the algorithm. Each individual TAMS run is based on $N_c = 32$ trajectories with duration $T_a = 5\tau_c$. The threshold a is set to 6σ, where σ is the standard deviation of the OU process (6.15), is this case $\sigma = 1/\sqrt{2}$. The dashed lines represent the bounds of the confidence interval on the estimate, defined by (6.19).

the estimate. It is computed based on the empirical variance over the realisations:

$$\Delta\hat{q}_K = \frac{1}{\sqrt{K}}\sqrt{\frac{1}{K-1}\sum_{k=1}^{K}(\hat{q}_k - \hat{q}_K)^2} \tag{6.19}$$

REMARK The definition of a *symmetric* confidence interval is discutable. Indeed, as the individual estimates $\hat{q}_k$ are lower bounded by 0, their distribution is not symmetric. A more rigorous approach requires the definition of asymmetric error bars, that can be obtained for instance by modelling the distribution of the $\hat{q}_k$ by a log-normal distribution or another type of asymmetric distribution.

6.4 APPLICATION OF THE TAMS TO EXTREMES IN TURBULENT FLOWS

In the previous section, the TAMS algorithm has been tested on one-dimensional Markovian dynamics: the Ornstein–Ulhenbeck (OU) process, defined by (6.15). We illustrated that in this case the TAMS leads to a tremendous improvement in the sampling of very rare trajectories associated with extreme fluctuations of the process. The TAMS therefore appears like a promising tool to investigate extreme fluctuations in dynamical systems. In this sections we discuss the application of the TAMS to turbulent flows. More precisely, we aim at sampling dynamical paths leading to extreme values for the drag acting on the square cylinder embedded in test flow (2).

A crucial step in any application of the AMS is the choice of the score function ξ. In the simplified case of Markovian stochastic dynamics $x(t)$ with one unique attractor, the optimal score function can be shown to be almost independent of time, when the trajectories are much shorter than the typical period of occurrence of the rare event of interest. See appendix D for a discussion of the optimal score function. In this case, the optimal score function is very close to the static committor $\xi_0(a) = \mathbb{P}_{x,0}(\tau_{\mathcal{B}} < \tau_{\mathcal{A}})$. See section 6.2.2 above for the definition of the sets $\mathcal{A}$ and $\mathcal{B}$. In one dimension, this motivates the choice of the score function as $\xi(x,t) = x$. Indeed, an increase of x certainly leads to an increase of the static committor and *vice-versa*, provided that the probability is monotonic. However, this context is rather exceptional. When the dynamics involves more than one degree of freedom, this is not necessarily the case. For practical applications of the AMS and TAMS algorithms on complex dynamics, the choice of the score function should rely on prior knowledge of the system and heuristic considerations about the dynamics of rare events. However, as a result of the complexity of the dynamics, finding an efficient score function close to the committor (6.14) is very difficult.

In this section we assess the application of the TAMS algorithm to turbulent dynamics based on the most naive choice for the score function: it is simply chosen as the observable of interest itself. More precisely, we address the numerical sampling of extreme fluctuations of the drag f_d acting on an obstacle in a turbulent flow, using the drag itself as the cost function. In what follows, the TAMS is applied to test flow (2). Note that, in contrast with the GKTL algorithm, the TAMS is not limited to time-averaged observables. In this section we therefore discuss the application of the TAMS in the two following cases:

- Instantaneous drag: $f_d[\mathbf{X}(t)]$, where $\{\mathbf{X}(t)\}_{0\leq t\leq T_a}$ denotes a trajectory of the numerical model and f_d the corresponding drag timeseries. In this case the score function is defined as $\xi(\mathbf{X}) = f_d$.
- Time-averaged drag: $F_T[\mathbf{X}(t)] = \frac{1}{T}\int_{t-T}^{t} f_d[\mathbf{X}(t)]\mathrm{d}t$ with a prescribed duration T for the averaging window. Note that, in this case, the time-averaged drag timeseries is defined on a different interval than the underlying timeseries, here $[T, T_a]$. In this case the score function is defined as $\xi(\mathbf{X}) = F_T$

We stress that, within this setup, there is *a priori* no reasons for the drag itself to be close to the optimal score function. The objective of the work presented is this section is twofold. First, we want to describe qualitatively the behaviour of the TAMS in this setup. Second, we want to assess the ability of the TAMS to sample trajectories associated to rare fluctuations of the drag for a lower computational cost than the one required by a direct sampling.

In this section we discuss the results of several TAMS experiments with varying trajectories duration T_a and number of trajectories N_c. The efficiency of the TAMS is analysed in a way very similar to section 6.3. To each iteration of the algorithm, we associate a computational cost based on the overall computational effort spent in simulating the flow from the beginning of the algorithm. The computational cost is computed in the same way as in section 6.3, *i.e.* following equation (6.17). See page 151 for a discussion of the computational cost of the TAMS. Note that, in the case of the average drag, resampled trajectories are never re-simulated over $[0;T]$, as the average drag F_T is defined on the interval $[T;T_a]$. Therefore the computation of the computational cost in (6.17) must be slightly modified, considering trajectories with a duration $T_a - T$ instead of T_a.

Let $\tilde{f_d}^{(j)}$ (resp. $\tilde{F}_T^{(j)}$) be the maximum of the drag (resp. averaged drag) along the resampled trajectory following iteration j. In the following, we compare the cost of the TAMS algorithm up to iteration j to the typical return time of fluctuation of amplitude $a \geq \tilde{f_d}^{(j)}$, or $a \geq \tilde{F}_T^{(j)}$. This is very similar to section 6.3, except that, in this case, we consider the maximum over the resampled trajectory, instead of the threshold $\mathcal{Q}_j^\star$. The two approaches are equivalent. In theses cases the return time cannot be computed analytically and is computed on the basis of a very long timeseries of the observable. The computation of return times based on long timeseries is discussed in chapter 7.

6.4.0.1 *Implementation of the TAMS for turbulent flows: the libTAMS library*

Despite its apparent algorithmic simplicity, the implementation of the TAMS for deterministic complex systems can be rather tricky. For instance, for turbulent flows, trajectories cannot be stored in memory. Selected trajectories must therefore be recomputed up to the branching point starting from restart states periodically saved along the dynamics. In addition, prior to the resampling step, the branching state of the resampled copy must be slightly perturbed, in order for the resampled trajectory to separate from its parent. In what follows the perturbation is applied in the same way as for the GKTL algorithm. See chapter 4, section 4.2.1 for more details about the perturbation of the trajectories. Additionally, considering an averaged cost function, such as the averaged drag F_T, the selection step must be done according to the moving average of the instantaneous observable.

As a consequence, the implementation of the TAMS very specific to a given problem, depending on the properties of the dynamics. Even though the structure of the algorithm does not change from a one-dimensional stochastic system to a DNS of a turbulent flows, data structures and implementation of the steps of the algorithm must be modified accordingly. Consequently, code-reuse is error-prone.

Motivated by these observations, we proposed a general object-oriented modelling of a TAMS simulation, independent of the underlying dynamics. This resulted in the development of the `libTAMS` C++ library which aims at facilitating the implementation and analysis of the TAMS applications. Using `libTAMS`, it is possible to write a TAMS code applicable to any kind of dynamics and choice for the choice function, whether it is time-integrated or not. This greatly helps debugging and analysis, as the code can be validated and tested on simple stochastic systems for which computations and post processing are much easier. The `libTAMS` library is presented in appendix E.

6.4.1 *Plan of numerical experiments*

In order to test the TAMS for turbulent flows, we performed several numerical experiments based on test flow (2). The total duration of the trajectories T_a and the number of copies N_c have been chosen following the same remarks as in section 6.3. Recall that the unbiasedness property of the AMS estimator (6.4) suggests to divide a run into K independent realisations, involving a lower number of copies. Consistently with the experiments presented in section 6.3, we first perform tests with $N_c = 32$ trajectories. As a comparison, we perform the same experiments with a number of trajectories that is four times larger, *i.e.* $N_c = 256$. The duration of the trajectories T_a must be set larger than the typical correlation time of the drag, so that an excursion far the typical value has the time to occur. In the following we denote by τ_c the typical correlation time of the drag. It corresponds to the timescale over which the autocorrelation function of the drag vanishes. See chapter 2 for a discussion of the estimation of τ_c for test flow (2). We stress that in the case of the averaged drag, the relevant correlation timescale is not τ_c, but the correlation time of the averaged process. In the following we denote by τ_c^T the correlation time of the averaged drag over a duration T, F_T. The correlation time of the averaged drag is close to the duration of the integration: $\tau_c^T \approx T$. As explained in section 6.3, setting $T_a \gg \tau_c$ or $T_a - T \gg \tau_c^T$ is not expected to provide better results. In what follows we run experiments for the instantaneous drag with $T_a = 5\tau_c$, as well as $T_a = 20\tau_c$ for comparison. In addition, we perform similar experiments for the averaged drag over $5\tau_c$. In this case $\tau_c^T \approx 5\tau_c$. As a consequence, we run two experiments with $T_a - T = 5\tau_c^T$ and $T_a - T = 20\tau_c^T$, respectively.

The N_c trajectories are initialised at $t = 0$ on random initial conditions from which the flow dynamics is computed until $t = T_a$. At each iteration of the algorithm, we record the maximum drag $\tilde{f}_d^{(j)}$, or maximum averaged drag $\tilde{F}_d^{(j)}$, over the resampled trajectory. In contrast with the experiment presented in section 6.3, we do not prescribe a stopping criterion for the algorithm. Instead, the selection/mutation procedure is iterated until all trajectories have reached the same

level. This limit is known as *extinction* and is discussed above in section 6.2.1.2. The objective of this series of experiments is to estimate the maximum gain from the TAMS algorithm with respect to direct sampling, before extinction is reached. It is expected to depend on the values of N_c and T_a. In the following we discuss the results of the experiments for both instantaneous and averaged drag. We illustrate that extinction is reached before significant gain is achieved with respect to direct sampling. Furthermore, for $T_a = 5\tau_c$ we show that at extinction all the trajectories concentrate on the trajectory having the highest maximum in the initial set of trajectory. In addition, we show that for $T_a = 20\tau_c$ too few fluctuations are sampled before extinction to improve the sampling. We illustrate that they are actually unrelated to the branching procedure.

6.4.2 *TAMS for the instantaneous drag*

The TAMS was first applied with $\xi(\mathbf{X}) = f_d(\mathbf{X})$, that is, resampling trajectories based on their maximum for the instantaneous drag f_d. In a way similar to figure 6.5, figure 6.8 displays the maximum $\tilde{f}_d^{(j)}$ over the resampled trajectories as a function of the computational cost C_j of the algorithm at iteration j, computed as (6.17). The difference between the two figures 6.5 and 6.8 is striking. Figure 6.8 illustrates the the TAMS does not lead to any computational gain with respect to direct sampling, unregarding the duration of the trajectories or the number of trajectories. In fact, the TAMS does not lead to new drag events with an amplitude higher that the maximum event among the initial trajectories. As a consequence, iterations of the algorithm are pointless, as resampled trajectories eventually concentrate on a common parent, that was already sampled during the initialisation step. Figure 6.9 illustrates this concentration of resampled trajectories along iterations, until extinction. It clearly shows that the trajectory ultimately responsible for the extinction is present in the ensemble of trajectories from the initialisation step. Increasing the number of copies to $N_c = 256$ and the length of the trajectories to $T_a = 20\tau_c$ leads to similar conclusions.

The early extinction displayed in figure 6.9 can be interpreted as follows. Because the dynamics is deterministic, the resampled trajectory separates from the parent over a timescale of the order of a few correlation times τ_c, resulting from the perturbation introduced at the branching point. We denote by τ_L the separation timescale. Let τ_m be the time at which the maximum of the parent trajectory is attained. Furthermore, let t_b be the time at which the branching occurs. If $\tau_m - t_b \ll \tau_L$ the resampled trajectory will follow the parent trajectory and fall back to typical values of the drag before it later separate. This is well verified in practice, and illustrated in figure 6.9c.

As a result, the selection/mutation procedure introduced by the TAMS does not lead to trajectories with higher fluctuations.

6.4.3 *TAMS for the time averaged drag*

In this case, the TAMS is applied with $\xi(\mathbf{X}) = F_T(\mathbf{X})$. More precisely,

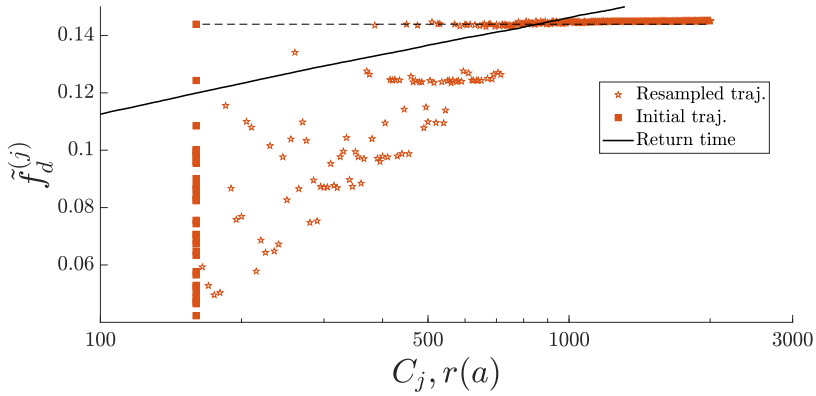
$$\xi(\mathbf{X}(t)) = \frac{1}{T}\int_{t-T}^{t} f_d[\mathbf{X}(t)]\mathrm{d}t \quad \text{for} \quad T \leq t \leq T_a \tag{6.20}$$

The selection is then done according to the maximum values of ξ for each trajectory for $T \leq t \leq T_a$. The TAMS was used with $N_c = 32$ and $N_c = 256$, for trajectories of length $T_a = 5\tau_c^T + T$, and $T_a = 20\tau_c^T + T$, with τ_c^T the correlation time of the averaged drag. In practice, $\tau_c^T \approx T$. Therefore, the duration of the time-averaged drag timeseries over the trajectories is $5\tau_c^T$ and $20\tau_c^T$, respectively. In a way similar to figure 6.8, figure 6.10 displays the maximum drag over the resampled trajectories $\tilde{F}_d^{(j)}$ at iteration j as a function of the computational cost C_j, defined in (6.11). The maximum over the initial trajectories $\{\tilde{F}_{d,n}^{(0)}\}_{1\leq n\leq N_c}$ is also displayed.

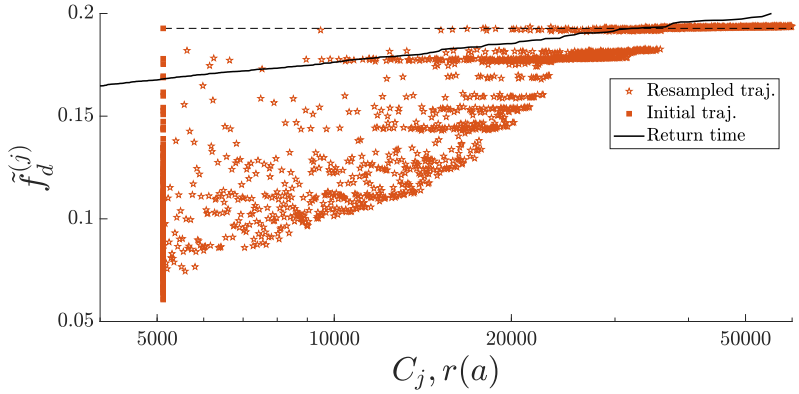
Similarly to figure 6.8, figure 6.10 illustrates that most of the iterations of the TAMS result in fluctuations already sampled in the ensemble of trajectories. This effect can be explained in the same way as in the case of the instantaneous drag. The resampled and the parent trajectories do not separate before the maximum of the parent trajectory is reached. As a consequence, the resampling cannot lead to a higher fluctuation close to the branching point. As a matter of fact, the resampling procedure *can* lead to higher fluctuations if the trajectories are long enough so that the resampled trajectories have time to fully decorrelate from their parent. Is this case their is a probability that the resampled trajectory exhibits a rare fluctuation. Naturally, this probability is very small as it corresponds to the original probability of the fluctuation. An example is illustrated in figure 6.11, in which one can see that a second fluctuation occurs well after the one the resampling step is based upon. As a result, this second fluctuation is uncorrelated from the first one. As in the case of the instantaneous drag, we conclude that the TAMS, *using the averaged drag itself as a score function*, is unable to improve the sampling of rare drag fluctuations.

6.4.4 *Discussion*

The failure of the TAMS can be imputed to the poor choice for the score function, here defined as the observable itself. Let us explain why. A new trajectory is resampled from the threshold a and at a branching time t_b, determined by the TAMS algorithm. The resampling

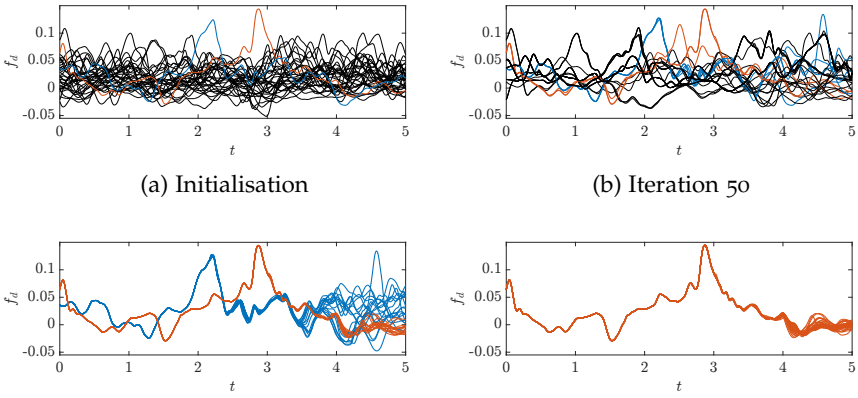


(a) $N_c = 32$, $T_a = 5\tau_c$



(b) $N_c = 256$, $T_a = 20\tau_c$

Figure 6.8: Maximum of the instantaneous drag along the resampled trajectories, denoted by $\tilde{f}_d^{(j)}$, as a function of the corresponding computational cost C_j, defined in (6.17). In both figures, the TAMS is iterated until extinction. The solid black line indicates the typical fluctuation amplitude a as a function of the corresponding *return time* $r(a)$. It is the typical timescale of occurrence of fluctuations above a. Furthermore, the square markers depict the maximum $\tilde{f}_{d,n}^{(0)}$, $1 \leq n \leq N_c$ over the initial trajectories. This figure illustrates that in both experiments the TAMS does not improve the sampling of rare fluctuations. On can see that the maximum values among the resampled trajectories concentrates on values already sampled over the initial trajectories. Eventually, extinction is reached as the algorithm is unable to generate a fluctuation higher than the maximum fluctuation in the initial ensemble of trajectories. This behaviour is illustrated in figure 6.9 on the basis of the drag timeseries involved in the experiment of figure 6.8a.



(a) Initialisation (b) Iteration 50

(c) Iteration 100 (d) Iteration 181

Figure 6.9: Visualisation of the drag timeseries corresponding to the set of trajectories in the ensemble, at different iterations along the algorithm. In this experiment, the TAMS is used with the instantaneous drag f_d as a score function. Trajectories have a duration $T_a = 5\tau_c$ and their number is $N_c = 32$. One can see that, as the number of iterations increases, the number of independent resampled trajectories drastically decreases until extinction at iteration 181. The timeseries corresponding to the two initial trajectories exhibiting the highest fluctuations are displayed in colour. In all four figures, colour is inherited from the parent trajectory. It illustrates that the two strongest initial trajectories quickly become predominant in the ensemble, as the other are discarded. As can be seen in figure 6.9c, trajectories only separate well after the maximum is reached. Therefore, resampling the blue trajectories based on the red trajectories can only change the overall maximum by a very small amount. Consequently, blue trajectories are discarded one after the other, each time resampled into another instance of the red trajectory. This new instance has roughly the same maximum as the others.

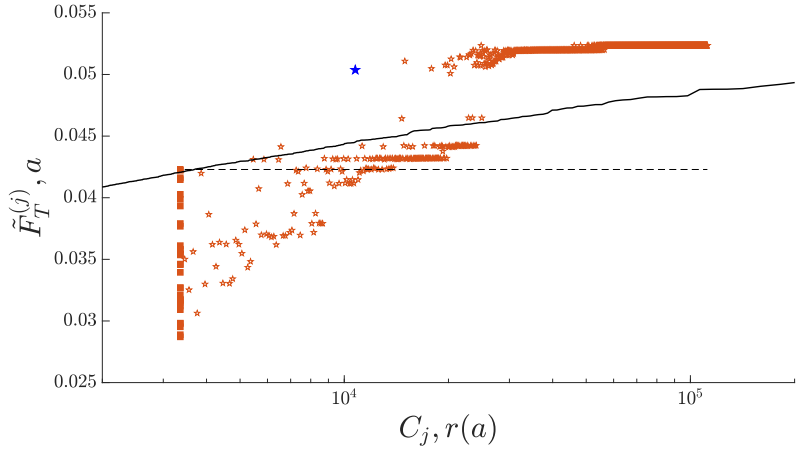


Figure 6.10: Maximum of the instantaneous drag along the resampled trajectories, denoted by $\tilde{F}_T^{(j)}$, as a function of the corresponding computational cost C_j, defined in (6.17). In both figures, the TAMS is iterated until extinction. In this experiment $N_c = 32$ and the trajectories are computed over $T_a = 105\tau_c$. The score function is the averaged drag F_T with $T = 5\tau_c$. Therefore, averaged drag timeseries have a duration $T_a - T = 100\tau_c \approx 20\tau_c^T$ where τ_c^T denotes the correlation time of the averaged drag. The solid black line indicates the typical fluctuation amplitude a as a function of the corresponding *return time* $r(a)$. It is the typical timescale of occurrence of fluctuations above a. Furthermore, the square markers depict the set of maximum values $\{\tilde{F}_{T,n}^{(0)}\}_{1\leq n\leq N_c}$ over the initial trajectories. This figure is similar to both figures 6.8a and 6.8b. It illustrates that, in this experiment, the TAMS does not improve the sampling of rare fluctuations. One can see that the maximum values among the resampled trajectories concentrates on values already sampled in the ensemble. The maximum value over the set of initial trajectory, *i.e.* $\max_n \tilde{F}_{T,n}^{(0)}$, is marked by a dashed straight line. As illustrated in figure 6.11, fluctuations above this value originate from the long duration of the trajectories, which allows resampled trajectories to fully decorrelate from their parent in between the branching point and $t = T_a$. However this does not help in improving the sampling as these fluctuations are sampled according to natural probability, that is very small. In fact, the whole point of using the TAMS is to sample such fluctuations with a higher probability. An example is the resampling step associated to the iteration marked by the blue star in the figure. The timeseries for the parent and resampled trajectories are illustrated in figure 6.11. Eventually, extinction is reached when the branching point is too close from $t = T_a$ so that the resampled trajectory does not have the time to fully separate.

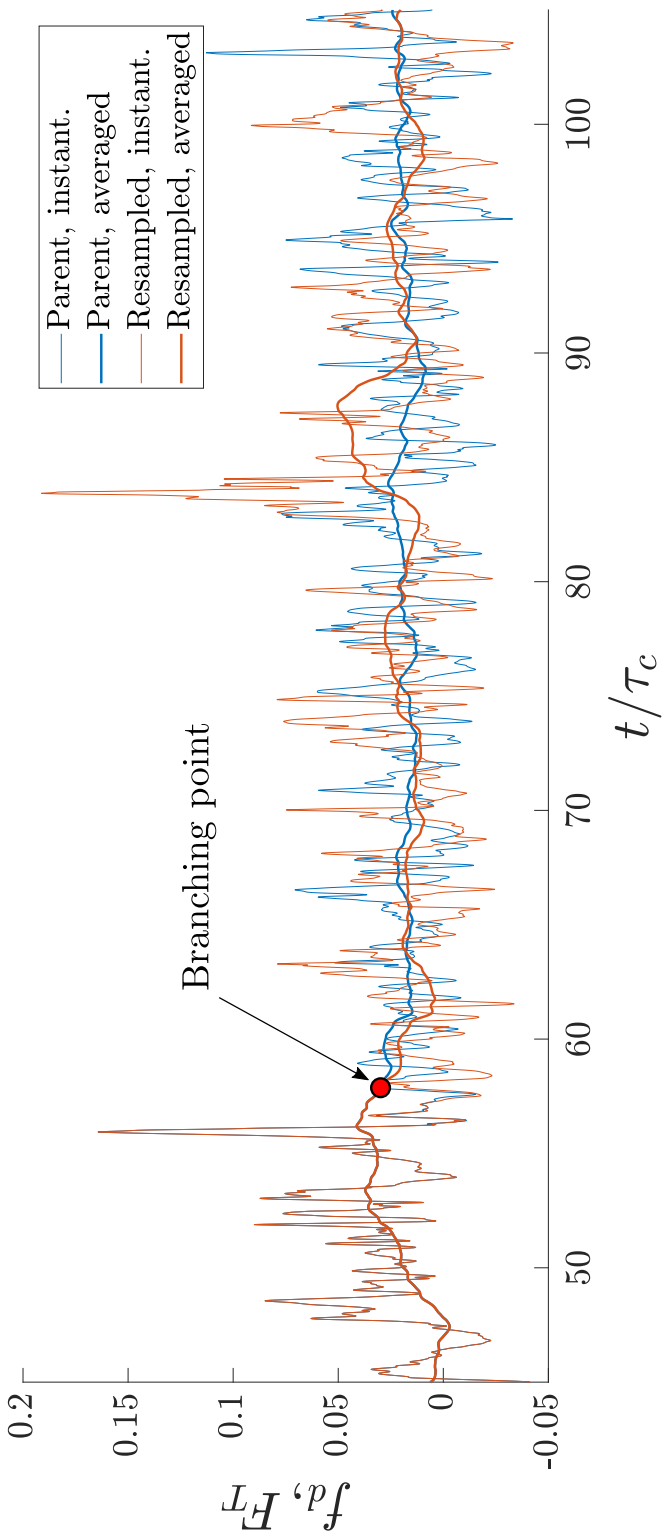


Figure 6.11: Illustration of the resampling step for a particular iteration of the TAMS. In this particular case, the resampled trajectory (orange) display a second, higher fluctuation. The TAMS is used with $N_c = 32$ copies simulated over $T_a = 105\tau_c$. The score function is chosen as the averaged drag (6.20) with an averaging window $T = 5\tau_c$. As a consequence, averaged timeseries have a duration of roughly $20\tau_c^T$ (the correlation time for the averaged timeseries is taken equal to the averaging window). One can see that the highest fluctuations for the resampled trajectory happens well after it has decorrelated from its parent. It therefore occurring independently from the TAMS, with a very small probability.

is initiated by adding a small noise at time t_b (or by resampling over the interval $[t_b, T_a]$ for stochastic dynamics). Let us assume that the parent trajectory reaches an extremum of the observable a_m attained at a time $t_{max} > t_b$. Because the dynamics reduced to the score function is not Markovian, it is very unlikely that a resampling at time t_b is enough to change significantly the dynamics afterwards: changing significantly the dynamics afterwards may require to change part of the history of the trajectory including times that are prior to t_b. A drastic example of this is obtained when $t_{max} - t_b$ is much smaller than the decorrelation timescale of the dynamics, that is close to the correlation time τ_c, or τ_c^T. In this case, a small change at time t_b produces nearly no effect at time t_{max}, and the resampled trajectory is nearly the same as the parent trajectory. Consequently, the maximum a_m is only changed by a small amount.

These results confirm that the efficiency of the TAMS highly depends on the choice of the score function. The design of an efficient score function must rely on *a priori* knowledge of the dynamics leading to the rare events of interest. For complex systems systems such as turbulent flows, it is therefore very difficult to find a good score function.

Generally speaking, the AMS algorithm have been reported to significantly improve the sampling of extreme events in several systems [5, 138, 139]. However, in all these studies, the dynamical equations are intrinsically stochastic. By contrast, the dynamics considered in the present work is deterministic. Moreover, is it extremely complex, with respect to previous applications of the AMS. In this case, it is very difficult to design a score function that is well suited to the particular problem at hand. Indeed, it would require to have *a priori* knowledge of the dynamics of the system, and more specifically about extreme events. Paradoxically, it is precisely because we do not have such knowledge that we want to apply rare event algorithms.

As a consequence, we tested the TAMS algorithm with the most straightforward score function possible: the drag itself. The study presented in this chapter illustrates that, with such a choice of score function, the TAMS cannot improve the sampling of extreme events.

Further studies must be performed in order to identify the reasons of the limitations of the TAMS for complex deterministic dynamics, as well as highlight the difference with previous successful applications of the AMS algorithm. The precise understanding of these limitations will then be useful to address the construction of novel algorithms based on the AMS and the TAMS, suited for complex deterministic dynamics, such as turbulent flows.

7

COMPUTING RETURN TIMES FROM TIMESERIES ANALYSIS AND RARE EVENT ALGORITHMS

In the previous chapters, we presented the application of two rare event algorithms, the Giardina–Kurchan–Tailleur–Lecomte and the Adaptive Multilevel Splitting, to the sampling of extreme fluctuations of the drag acting on an obstacle immersed in a turbulent flow. In chapter 5, we focused on extreme fluctuations of the drag averaged over several correlation times. We showed that the application of the GKTL algorithm allows for a significant improvement in the sampling of extremes, with respect to brute force sampling by means of a simulation of the flow over a long duration. A consequence of this better sampling is better estimation of the probability of rare events. For instance, we showed in chapter 5 that the GKTL algorithm leads to a much improved estimate of the tails of the SCGF describing extreme fluctuations of the drag averaged over several correlation times.

In this chapter we are interested in the average waiting time between for the occurrence of a rare event, referred to as its *return time*. It is a useful statistical concept for practical applications. For instance, insurances or public agency may be interested by the return time of a 10 m flood of the Seine river in Paris. However, the computation of return times for extreme fluctuations cannot be addressed with the sole knowledge of the stationary PDF of the underlying process, as justified in section 7.1. Furthermore, due to their scarcity, reliably estimating return times for rare events is very difficult using either observational data or direct numerical simulations.

For rare events, an estimator for return times can be built from the extrema of the observable on trajectory blocks. In this chapter, we show that this estimator can be improved to remain accurate for return times of the order of the block size. More importantly, we show that this approach can be generalised to estimate return times from numerical algorithms specifically designed to sample rare events. So far, those algorithms often compute probabilities, rather than return times. The approach we propose provides a computationally efficient way to estimate numerically the return times of rare events for a dynamical system, gaining several orders of magnitude of computational costs. We illustrate the method on two kinds of observables, instantaneous and time-averaged, using two different rare event algorithms, for a simple stochastic process, the Ornstein–Uhlenbeck process. As an example of realistic applications to complex systems, we finally discuss extreme values of the drag on an object in a turbulent flow.

7.1 INTRODUCTION

The theoretical framework which has been developed over the last decades in statistical physics to tackle the investigation of rare events is that of *large deviation theory* [35, 42, 49, 165, 174]. Numerical methods have also been developed to efficiently sample rare events, which are not amenable to classical Monte-Carlo methods [6, 96, 106]; see [22, 145] for general references on rare event simulation. Those algorithms can be roughly divided into two main classes: those which work in state space, and evolve a population of *clones* of the system according to selection rules biased to favour the appearance of the desired rare event [27, 56, 68, 118, 139], and those which try to sample directly in path space the histories of the system which exhibit the phenomenon of interest [11, 40, 41, 66, 69, 101]. In chapters 4, 5 and 6 we discussed the application of algorithms of the former class to the study of rare events in fluid mechanics problems. Note that most of those algorithms ultimately compute one-time statistics. Typically, they yield the stationary probability distribution of the system, for which they sample efficiently the tails, or alternatively, large deviation rate functions or scale cumulant generating functions. They can also compute reactive trajectories corresponding to the transition between two metastable states.

For many practical applications however, the most useful information about a rare event is its *return time*: it is the typical time between two occurrences of the same event. For instance, this is how hydrologists measure the amplitude of floods [156]. As a matter of fact, one of the motivations of Gumbel, a founding father of extreme value theory, was exactly this problem [70]. Other natural hazards, such as earthquakes [33] and landslides [127], are also ranked according to their return time. Similarly, climatologists seek to determine how the frequency of given heat waves [109, 131] or cold spells [26] evolves in a changing climate [149]. Public policies rely heavily on a correct estimate of return times: for instance, in the United States, floodplains were defined in the National Flood Insurance Program in 1968 as areas vulnerable to events with a 100-year return time. Such definitions are then used to determine insurance policies for home owners. In the industry as well, return times are the metric used by engineers to design systems withstanding a given class of events. Another property describing rare events is the average time between successive records [62]; here, because of its importance in practical applications, we focus on the return time, i.e. the average time between events of a given amplitude. Just like the extreme values of any observable, the return time of a rare event is very difficult to estimate directly from observational or numerical data, because extremely long timeseries are necessary.

The return time may be estimated heuristically by interpreting it as a *first-passage time*. The *first-passage time* (sometimes also called *first exit time*) is defined as the time it takes for a stochastic process to reach the boundary of a given domain for the first time; the properties of this random variable have been studied extensively in statistical physics [16, 133]. Then, the return time $r(a)$ for an event of amplitude a may be at first sight related to the inverse of the stationary probability p_s: $r(a) = \tau_c(a)/p_s(a)$. We stress that the correlation time $\tau_c(a)$ usually depends on a, but remains bounded when $p_s(a)$ goes to zero. This is true for instance for a system perturbed by a small-noise ϵ at the level of large deviations: $r(a) \underset{\epsilon\to 0}{\asymp} e^{U(a)/\epsilon}$, where the quasi-potential U is defined by $p_s(a) \underset{\epsilon\to 0}{\asymp} e^{-U(a)/\epsilon}$ [49]. However, the return time is only roughly proportional to the inverse of the stationary probability [123]. In order to compute $\tau_c(a)$ one has to go beyond large deviation theory. For instance for gradient dynamics and for first exit time problems, exact formulas exists [53, 98, 134], valid at leading order in ϵ. We stress that different formulas are obtained depending on the hypothesis made on the domain that the particle exits. Generalisations to irreversible non gradient dynamics also exist, see [14] and references therein. From these computations, it appears clearly that $\tau_c(a)$ is not simply related to $p_s(a)$ and that the return time $r(a)$ is a trajectory property, not amenable to a one-point statistics like $p_s(a)$.

From a modelling perspective, it is natural to assume that successive occurrences of a rare event are independent from one another [37, 43, 102]. Then, the average number of events occurring in a time interval is proportional to the length of that interval. This is the definition of a Poisson process. In this case, all the statistics are encoded in a single parameter, the rate of the Poisson process. In the following, we will assume that we are dealing with the simple case of a well identified process that can be described by a single return time or rate. This is often a sufficient framework; indeed the long time behaviour of many systems can be described phenomenologically, or exactly in some limits, as Markov processes described by a set of transition rates describing independent processes, see for instance [49] for systems driven by a weak noise. We note however that many other physical systems are not amenable to such a simple effective Markov processes, for instance structural glasses or amorphous media.

There is thus a need to develop rare event algorithms specifically designed for computing return times, valid also when large deviation estimates are not relevant. This is the aim of this chapter. The approach developed in this work relies on the combination of two observations. First, if one assumes that rare events are described by a Poisson process, then return times can be related to the probability of observing extrema over pieces of trajectories, which are of duration much larger than the correlation time of the system, but typically much smaller than the computed return times. Second, several classes of rare event algorithms

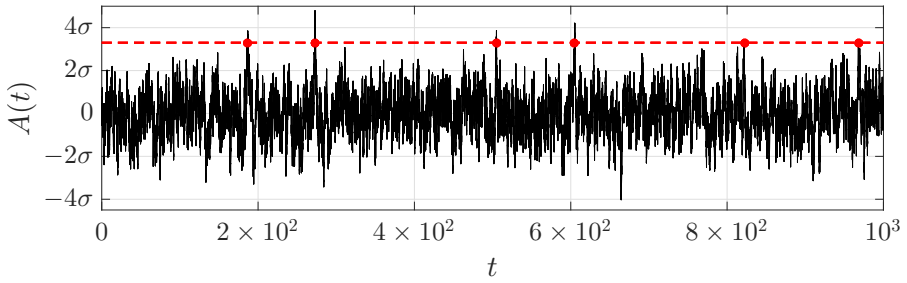
can be easily generalised to compute the probability of extrema over pieces of trajectories, rather than to compute single point statistics. We show that combining these two remarks enables us to build a powerful tool to compute return times in an elementary way with simple and robust algorithms. As a side remark, we also discuss a new way to construct return time plots from a timeseries, which provides an important improvement for return times moderately larger than the sampling time, even when we are not using a rare event algorithm.

We illustrate the method by computing return times, first for an instantaneous observable (one-point statistics) using the Adaptive Multilevel Splitting (AMS) algorithm [27, 29], and second for a time-averaged observable, using both the AMS algorithm and the GKTL algorithm [58]. The computation of return times with the AMS algorithm actually makes use of a generalisation called the Trajectory Adaptive Multilevel Splitting (TAMS) algorithm, introduced in chapter 6, that is more appropriate to this problem. As a matter of fact, we first formulated the TAMS algorithm motivated by the computation of return times. This generalisation has several practical advantages: it computes directly return times $r(a)$ for a full range of return level a rather than a single one, and it avoids the tricky estimation of time scale on an auxiliary ensemble, and the sampling from this auxiliary ensemble. As a test, we first carry out these computations for a simple stochastic process, the Ornstein–Ulhenbeck (OU) process, for which analytical results are available and the accuracy and efficiency of the algorithm can be tested thoroughly. Then, to demonstrate the usefulness of the method in realistic applications, we briefly showcase a problem involving a complex dynamical system: extreme values of the drag on an object immersed in a turbulent flow.

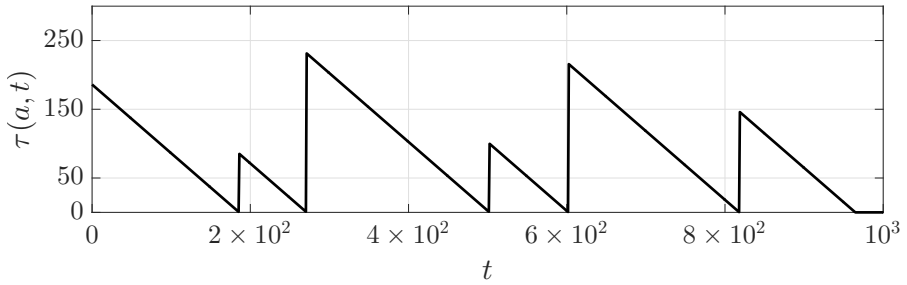
The structure of this chapter is as follows: in section 7.2, we introduce the method to compute return times from a timeseries and from rare event algorithms. We apply the method to compute return times for the instantaneous and time-averaged observables for an Ornstein–Uhlenbeck process using both the GKTL and AMS algorithms introduced in chapters 4 and 6, respectively. In section 7.3 we discuss and illustrate the use of the AMS for return times for fluctuations of instantaneous observables. In section 7.4 we address the computation of return times for time-averaged observables using both GKTL and AMS algorithms. Eventually, we discuss the application to complex dynamical systems in section 7.5. Conclusions are presented in section 7.6.

7.2 RETURN TIMES: DEFINITION AND SAMPLING METHODS

We consider a statistically time homogeneous ergodic process (a stationary timeseries) $\{A(t)\}_{t\geq t_0}$. Typically, $A : \mathbb{R}^d \to \mathbb{R}$ is an observable on a system of interest, considered here as a $\mathbb{R}^d$-valued stochastic



(a) Sample timeseries from an OU process



(b) Waiting time $\tau(a,t)$

Figure 7.1: An example of a random process (a) and the waiting time (b) associated to events reaching a given threshold. **(a)**: Sample timeseries (black line), generated from an Ornstein–Uhlenbeck process (7.1) ($\alpha = 1$, $\epsilon = 1/2$; $\sigma = 1/\sqrt{2}$ is the standard deviation). We are interested in fluctuations which reach a prescribed threshold a (red dashed line). These events are identified by the red dots. **(b)**: Time evolution of the waiting time $\tau(a,t)$ (see (7.2)) associated to the above timeseries: it is a succession of affine parts with slope -1. Note that in principle, there should be small time intervals such that $\tau(a,t) = 0$, corresponding to the duration of the event with $A(t) > a$, separating the triangles. Here, the duration of the events is too small for such intervals to be visible.

process $\left(X_t\right)_{t \geq t_0}$, and we shall denote $A(t) = A(X_t)$. We are interested in the statistical distribution of events in which the observable reaches a prescribed threshold a. The occurrence of such events is illustrated for a sample Ornstein–Ulhenbeck (OU) process, on Fig. 7.1a. The OU process is defined as follows:

$$\dot{x}(t) = -\alpha x(t) + \sqrt{2\epsilon}\eta(t) \tag{7.1}$$

where α and ϵ are numeric constants and η a Gaussian white noise. Note that α and ϵ can be absorbed by a change of timescale.

In a first section we give a precise definition for the *return time* of a rare event. Based on this definition, we describe an efficient approach to the computation of return times from a timeseries of the process of interest in section 7.2.1. In a second part, we extend this approach to the computation of return times based on the output of rare event algorithms.

7.2.1 *Computing return times from a timeseries*

7.2.1.1 *Definition of return times*

We define the return time for a given threshold a as the average time one has to wait before observing the next event with $A(t) > a$. More precisely, we define the waiting time

$$\tau(a,t) = \min\{\tau \geq t \,|A(\tau) > a\} - t. \tag{7.2}$$

As an illustration, the waiting time $\tau(a,t)$ is shown for our sample Ornstein–Uhlenbeck process on Fig. 7.1b. Then, the return time $r(a)$ for the threshold a is defined as

$$r(a) = \mathbb{E}_{x_0,t_0}[\tau(a,t)], \tag{7.3}$$

where $\mathbb{E}$ is the average with respect to realisations of the process X with initial condition $X_{t_0} = x_0$ (hence the notation $\mathbb{E} \equiv \mathbb{E}_{x_0,t_0}$ in that case), or is a time average for an ergodic process. From now on, we shall omit the indices when there is no ambiguity. The return time $r(a)$ is independent of time because the process is homogeneous.

The problem we consider in this section is that of estimating $r(a)$ from a sample timeseries of duration $T_d : \{A(t)\}_{0\leq t\leq T_d}$. The definition leads to an obvious estimator for $r(a)$, the *direct estimator* $\hat{r}_D$ defined by

$$\hat{r}_D(a) = \frac{1}{T_d}\int_0^{T_d} \tau(a,t)\,\mathrm{d}t = \frac{1}{T_d}\sum_{n=1}^{N_d}\frac{\tau_n^2}{2}, \tag{7.4}$$

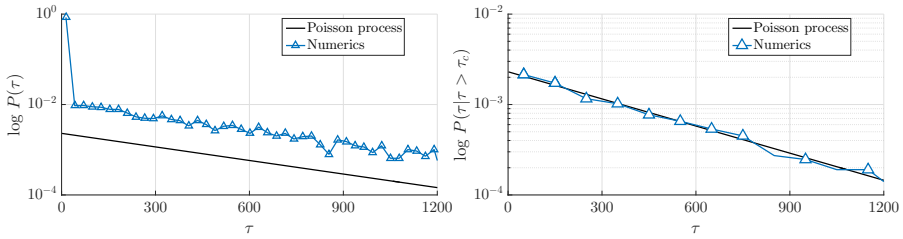
where τ_n is the duration of the successive intervals over which $A(t) \leq a$, and N_d is the number of such intervals. The last identity in (7.4) is illustrated graphically in Fig. 7.1b: the integral of $\tau(a,t)$ is given by computing the total area beneath the triangles.

In the limit of rare events, the return time will also be the average time between two successive independent events. However the definition (7.3) for the return time has the big advantage of not having to deal with the definition of independent events, which is cumbersome when time correlations are not negligible. We explain this further in the following section.

7.2.1.2 *Return times and the distribution of successive events*

Estimating return times using (7.4) implies computing the time intervals τ_n between successive events with $A(t) > a$. When a is large enough, most of the times $A(t) < a$ and very rarely $A(t) > a$. Then we can distinguish two kinds of contributions to the time intervals τ_n. On the one hand, we have correlated events corresponding to fluctuations around the threshold value a, on a timescale of the order

of the correlation time. From our point of view, these correspond to the same event, with a finite duration. On the other hand, there are successive events such as those depicted in Fig. 7.1a, which can be considered as statistically independent events. Therefore, we expect those events to form a Poisson point process, and the corresponding time intervals τ_n should be distributed according to the distribution of time intervals of a Poisson process: $P(\tau) = \lambda \exp(-\lambda\tau)$ [37, 43, 102].



(a) (b)

Figure 7.2: PDF of waiting times between two consecutive fluctuations of amplitude $a = 2.5$, estimated from a timeseries of length $T_d = 10^6$ of the Ornstein–Uhlenbeck process (7.1) with $\alpha = 1$ and $\epsilon = 1/2$ (blue triangles), and assuming the events follow a Poisson process with rate $1/r(a)$, $P(\tau) = e^{-\tau/r(a)}/r(a)$ (black solid line), where $r(a)$ is computed from the timeseries. The correlation time of the Ornstein–Uhlenbeck process is $\tau_c = 1/\alpha = 1$. **(a)** Taking all intervals into account, including those corresponding to oscillations around the threshold. **(b)** Discarding small intervals ($\tau < \tau_c$) linked to oscillations around the threshold.

Figure 7.2a shows the Probability Density Function (PDF) of the time interval between two occurrences of an event $A(t) > a$, drawn from a sample timeseries generated with an Ornstein–Uhlenbeck process. One can see that most of the contributions are indeed small intervals of the order of the correlation time. Discarding all the time intervals below the correlation time, one obtains the PDF displayed in Fig. 7.2b, which coincides with the exponential distribution corresponding to a Poisson point process.

When a is large, $r(a) \gg \tau_c$ where τ_c is the correlation time of the process. Then the contribution of intervals τ_n of duration comparable to τ_c in the formula (7.4) becomes asymptotically negligible compared to the contribution of the time intervals $\tau_n \gg \tau_c$. Graphically, this may be seen as the fact that the sum in (7.4) is dominated by the contribution of very big triangles, while for small a all the triangles have roughly the same area. Then, the return time $r(a)$ coincides with the average time between two statistically independent events exceeding the value a. In other words, rare fluctuations can be considered as independent from one another, their duration can be neglected compared to their return time, and the distribution of such events is well approximated by a Poisson process of rate $\lambda(a) = 1/r(a)$.

Neglecting the duration of the extreme events yields $\sum_{n=1}^{N_d} \tau_n \approx T_d$. As a result, one can check that

$$\frac{1}{T_d}\sum_{n=1}^{N_d}\frac{\tau_n^2}{2} \approx \frac{N_d}{\sum_{n=1}^{N_d}\tau_n}\frac{1}{N_d}\sum_{n=1}^{N_d}\frac{\tau_n^2}{2} \underset{N_d\to\infty}{\to} \frac{1}{2}\frac{\mathbb{E}\left[\tau^2\right]}{\mathbb{E}\left[\tau\right]} = \frac{1}{\lambda(a)} = r(a), \quad (7.5)$$

where the average in this computation is taken with respect to the Poisson process interval PDF $P(\tau) = \lambda \exp(-\lambda\tau)$.

One may be tempted to use the estimator $\hat{r}'_D(a) = \frac{1}{N_d}\sum_{n=1}^{N_d}\tau_n$ instead of the estimator $\hat{r}_D$ defined by (7.4). For an actual Poisson process, that would just give the same result. However this estimator would be more sensitive to the effect of a finite correlation time, since the contributions from time intervals $\tau_n \approx \tau_c$ between successive events will only become negligible linearly in $\tau_c/r(a)$, as opposed to quadratically in formula (7.4).

From now on, we shall assume that the statistics of rare events is Poissonian. This is a reasonable approximation for many dynamical systems as long as there is a well-defined mixing time after which the initial conditions are forgotten. Of course, it would not hold for systems with long-term memory. Note that this assumption is similar to the *Independent Interval Approximation* used in the context of persistence [16]. In the next paragraph, we use this assumption to derive new expressions that allow for accurate and efficient sampling of the return times.

7.2.1.3 *Sampling return times for rare events*

In this section we present an alternative way to compute return times, that provides an easier and more efficient way to draw return time plots for rare events than using the direct estimator (7.4). Let us divide the timeseries $\{A(t)\}_{0\le t\le T_d}$ in M blocks of duration $\Delta T \gg \tau_c$, so that $T_d = M\Delta T$, and let us define the block maximum

$$a_m = \max\left\{A(t)\,|(m-1)\Delta T \le t \le m\Delta T\right\}, \quad (7.6)$$

and $s_m(a) = 1$ if $a_m > a$ and 0 otherwise, for $1 \le m \le M$. This procedure is illustrated in figure 7.3.

For rare events, *i.e.* $r(a) \gg \tau_c$, the number of events

$$N(t) = \sum_{m\le\lceil t/\Delta T\rceil} s_m(a)$$

is well approximated by a Poisson process with density $\lambda(a) = 1/r(a)$. Then, assuming $\tau_c \ll \Delta T \ll r(a)$, the probability $q_m(a)$ that a_m be larger than a is well approximated by $q_m(a) \simeq \Delta T/r(a)$. As $q_m(a)$

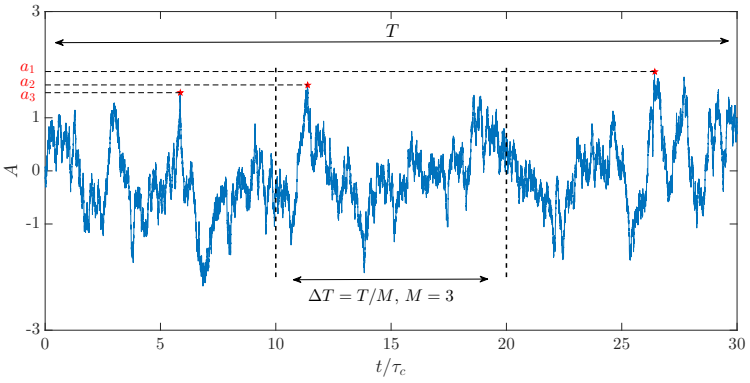


Figure 7.3: Illustration of the computation of return times from a timeseries based on the block-maxima method. The total timeseries is divided into M blocks, which individual duration is long enough so that they can be considered independent realisations of the process. For each of these blocks, the *block-maximum* (7.6) is computed, depicted as red stars in this figure. In the present example, this result in a set of $M = 3$ maxima, which values will be used to compute return times based on (7.8). We stress that this figure is a pedagogical illustration: in practice we use a much longer timeseries, typically of length $T_{tot} = 10^6 \tau_c$ and therefore a much greater number of blocks. The timeseries displayed in this figure originates from a realisation of the OU process (7.1) with $\alpha = 1$ and $\epsilon = 1/2$.

can be estimated by $\frac{1}{M}\sum_{m=1}^{M} s_m(a)$, an estimator of $r(a)$ is the *block maximum estimator*:

$$\hat{r}_B(a) = \frac{T_d}{\sum_{m=1}^{M} s_m(a)}. \tag{7.7}$$

This is the classical method for computing the return time of rare events, valid when $\Delta T \ll r(a)$ [125].

We now introduce a new, more precise estimator, also valid when $\Delta T / r(a)$ is of order one. It is obtained by using $q_m(a) = 1 - e^{-\Delta T / r(a)}$. Then, a better estimator of $r(a)$ is the *modified block maximum estimator*:

$$\hat{r}'_B(a) = -\frac{\Delta T}{\ln\left(1 - \frac{1}{M}\sum_{m=1}^{M} s_m(a)\right)}. \tag{7.8}$$

To compute these estimators in practice, we sort the sequence $\{a_m\}_{1 \le m \le M}$ in decreasing order and denote the sorted sequence

$$\{\tilde{a}_m\}_{1 \le m \le M}$$

such that $\tilde{a}_1 \ge \tilde{a}_2 \ge ... \ge \tilde{a}_M$. Based on (7.7), we then associate to the threshold $\tilde{a}_m$ the return time $r(\tilde{a}_m) = M\Delta T / m$. Indeed, $\sum_{\ell=1}^{M} s_\ell(\tilde{a}_m) = m$, which means that m events with amplitude larger than $\tilde{a}_m$ have

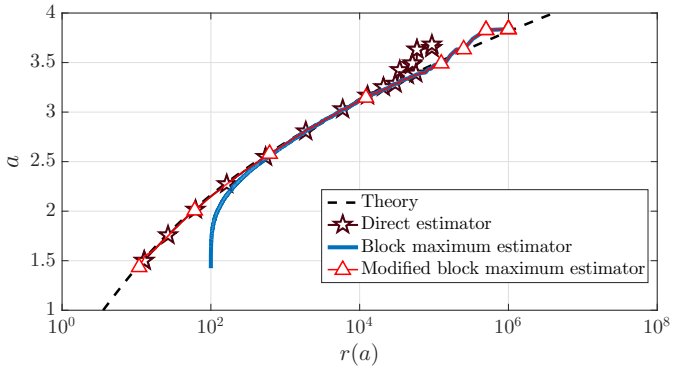


Figure 7.4: Return time plots for the Ornstein–Uhlenbeck process (7.1) with $\epsilon = 1/2$, $\alpha = 1$, estimated from a timeseries of length $T_d = 10^6 \tau_c$ using the direct estimator $\hat{r}_D$ (7.4) (pentagrams), the block maximum estimator $\hat{r}_B$ (7.7) ($\Delta T = 100$, solid blue line), and the enhanced block maximum estimator $\hat{r}'_B$ (7.8) ($\Delta T = 100$, solid red line and white triangles). These estimates are compared to the analytical solution [105](dashed black line).

been observed over a duration $M\Delta T$. Alternatively, using the more precise estimator $\hat{r}'_B$ (7.8) we associate to the threshold $\tilde{a}_m$ the return time $r(\tilde{a}_m) = -\frac{\Delta T}{\log\left(1-\frac{m}{M}\right)}$. The return time plot represents $\tilde{a}_m$ as a function of $r(\tilde{a}_m)$, as illustrated for instance on Fig. 7.4. Let us stress again that formulas (7.7) and (7.8) and this method of plotting the return time are meaningful only if doing block maxima, and for ranges of parameters such that $\tau_c \ll \Delta T \ll r(a)$ for (7.7) or $\tau_c \ll \Delta T$ for (7.8).

Figure 7.4 illustrates the three methods for computing return times from a timeseries: from the definition (7.4) and the two formulas (7.7) and (7.8). The sample timeseries used in this figure is extracted from an Ornstein–Uhlenbeck process, for which the return time curve can also be computed analytically [105]. One can see that both formulas (7.7) and (7.8) lead to the same estimate for events with $r(a) \gg \Delta T$. However, formula (7.7) fails to yield a correct estimate as soon as $r(a) \simeq \Delta T$.

For rare events, plotting return times using (7.7), as is classically done, proves itself much more convenient and efficient than the naive sampling using (7.4). It is important to note however, that the use of (7.7) is valid only after computing maxima over an interval of duration ΔT much larger than τ_c, a remark that not been considered in many previous publications. Moreover, the generalisation (7.8) we propose in this paper is much more accurate for events with a return time of order of ΔT. This procedure to compute return time plots can also be generalised in combination with the use of rare event algorithms, as we shall see in the next section.

7.2.2 *Computing return times from a rare event algorithm*

In section 7.2.1, we defined the return time for a time-homogeneous stochastic process and explained how to efficiently compute it for rare events from a timeseries. However, a major difficulty remains. Indeed, we still have to generate numerically the rare events in the timeseries, which comes at a large computational cost. In the present section, we explain how to apply the above method to the data produced by algorithms designed to sample efficiently rare events instead of direct simulations.

As illustrated in chapters 4, 5 and 6, rare event algorithms provide an effective ensemble of M trajectories $\{X_m(t)\}_{0\leq t\leq T_a}$ $(1 \leq m \leq M)$. Note that the length T_a of the trajectories generated by the algorithm does not necessarily coincide with the length T_d of the trajectory generated by direct sampling: in practice, as we shall see, $T_a \ll T_d$. For each of these trajectories, we compute the maximum of the observable over the time evolution $a_m = \max_{0\leq t\leq T_a}(A(X_m(t)))$. This is similar to the block maximum method described in section 7.2.1.3, with each trajectory playing the role of a block. There is however a major difference: unlike in the block maximum method, the different trajectories sampled by the rare event algorithm do not have identical statistical weight. To each trajectory $X_m(t)$, and thus to each maximum a_m, is associated a probability p_m computed by the algorithm. Hence, rather than just a sequence $\{a_m\}_{1\leq m\leq M}$, rare event algorithms yield a sequence $\{a_m, p_m\}_{1\leq m\leq M}$. The generalisation of the block maximum formula (7.8) to non-equiprobable blocks is straightforward and leads to the estimator

$$\hat{r}_A(a) = -\frac{T_a}{\ln\left(1-\sum_{m=1}^{M} p_m s_m(a)\right)}. \tag{7.9}$$

Of course, we could construct similarly an estimator generalising (7.7), but as we have seen in the previous section, the estimator (7.8) yields better performance.

In practice, to plot the return time curve, we sort the sequence $\{a_m, p_m\}_{1\leq m\leq M}$ in decreasing order with respect to the a_m, and denote the sorted sequence $\{\tilde{a}_m, \tilde{p}_m\}_{1\leq m\leq M}$ such that $\tilde{a}_1 \geq \tilde{a}_2 \geq ... \geq \tilde{a}_M$. We then associate to the threshold $\tilde{a}_m$ the return time

$$\hat{r}_A(\tilde{a}_m) = -\frac{T_a}{\ln\left(1-\sum_{\ell=1}^{m} \tilde{p}_\ell\right)}. \tag{7.10}$$

Indeed, the sum of the weights of the events with amplitude larger than $\tilde{a}_m$ is $\sum_{\ell=1}^{m} \tilde{p}_\ell$. Again, the return time plot represents a as a function of $r\,(a)$.

We stress that the method described here does not depend on the observable of interest, or on the details of the algorithm itself.

In the remainder of this chapter, we provide a *proof-of-principle* for this method, by considering two kinds of observables, sampled by two different algorithms. In section 7.3.1 we first study the return times for instantaneous observables using the Adaptive Multilevel Splitting (AMS) algorithm. Second, in section 7.4 we turn to time-averaged observables using both the AMS and the Giardina–Kurchan–Tailleur–Lecomte (GKTL) algorithm. We show that the method allows to accurately compute return times at a much smaller computational cost than direct simulation. In both cases, we apply the technique to the simple case of an Ornstein–Uhlenbeck process, for which the results are easily compared with direct simulation and theoretical predictions, before illustrating the potential of the method for applications in complex systems in section 7.5.

7.3 RETURN TIMES SAMPLED WITH THE ADAPTIVE MULTILEVEL SPLITTING ALGORITHM

In this section, we present the computation of return times by applying the method presented in section 7.2.2 to the Adaptive Multilevel Splitting, introduced in chapter 6. This algorithm follows the strategy of *splitting methods* for the estimation of rare event probabilities, which dates back to the 1950s [73]. Many variants have been proposed since then. The AMS algorithm can be interpreted as simulating a system $\{x_i(t)\}$ of interacting replicas (instead of independent replicas in a crude Monte Carlo simulation), with some *selection and mutation* mechanism. In chapter 6 we described a modified version of the AMS, referred to as TAMS, specifically designed to compute the probability of trajectories of finite durations that go beyond a fixed threshold. The main motivation for the formulation of the TAMS algorithm is actually the computation of return times through relation (7.9). In section 7.3.1, we show how the algorithm enables us to estimate return times, under the Poisson statistics assumption made in section 7.2. Finally, we illustrate in section 7.3.2 the method by computing the return times for an Ornstein–Uhlenbeck process.

7.3.1 *Computing return times with the TAMS*

Recall that the TAMS algorithm generates an ensemble of M trajectories $x_m(t)$ with associated probability p_m, as explained in chapter 6 on page 151. Additionally, recall that for any observable $O[x(t)]$, we can define an estimator based on our sampling of trajectory space:

$$\hat{O}_M = \sum_{m=1}^{M} p_m O[x_m(t)]. \tag{7.11}$$

It follows directly from (7.11) that an estimator of $q(a)$ is:

$$\hat{q}_M(a) = \sum_{m=1}^{M} p_m s_m(a), \tag{7.12}$$

where $a_m = \max_{0 \le t \le T_a} A(x_m(t))$ is the maximum value for the observable over the trajectory m, p_m the associated probability, see 6, page 151, and $s_m(a) = 1$ if $a_m > a$, 0 otherwise (7.2.1.3).

As explained in 7.2.1.3, the return time is related to $q(a)$ by the hypothesis that these events are Poissonian, and we obtain the estimator for the return time $\hat{r}_M(a) = \frac{T_a}{\ln(1-\hat{q}_M(a))}$ given by (7.9) (alternatively, we could use $\hat{r}_M(a) = \frac{T_a}{\hat{q}_M(a)}$). In essence, to draw return time plots, it suffices to sort the set $\{(a_m, p_m)\}_{1 \le m \le M}$ according to the a_m and use (7.10), as described in 7.2.2. Note that in practice, with the particular choice of score function $\xi(t, x) = A(x)$, storing the levels $\mathcal{Q}_n^{(j)}$ for the killed trajectories directly provides the corresponding values a_m.

By definition, the estimators $\hat{q}_M(a)$ and $\hat{r}_M(a)$ are random variables. In chapter 6, section 6.1, we describe their statistical properties, and how to interpret them in terms of consistency and efficiency of the AMS algorithm. In particular, we show that $\hat{q}_M(a)$ is an unbiased estimator of $q(a)$, study the variance, and show the existence of a Central Limit Theorem.

The total number of generated trajectories is $M = N_c + \tilde{J}$, where N_c is the number of trajectories in the initial ensemble and $\tilde{J} = \sum_{j=1}^{J} l_j$ is the total number of resampled trajectories along the J iterations of the algorithms. The number of resampled trajectories at iteration j is l_j. In the following we discuss two choices for the number of iterations J. First, the TAMS can be used with a fixed number of iterations. Alternatively, as is often seen in the AMS literature, one may decide to iterate the algorithm until all trajectories reach a prescribed threshold a. Then J is a random number. In that case, the threshold a becomes the control parameter for the stopping criterion. Under those circumstances, the estimator $\hat{q}_M$ can be expressed as

$$\hat{q}_M(a) = \prod_{j=1}^{J} \left(1 - \frac{\ell_j}{N_c}\right). \tag{7.13}$$

This formula remains valid in the case where the number of iterations J is prescribed: it suffices to define the threshold a *a posteriori*, by choosing $a = \min_{1 \le n \le N_c} a_n^{(J)}$ the minimum value of the a_m among the final trajectories. The formula could also be used to compute $\hat{q}_M(b)$ with $b < a$, simply by changing the number of iterations required to meet the stopping criterion. In practice, the most convenient approach is to use the expression given in (7.12).

In the above, we have defined the AMS estimators $\hat{q}_M$ and $\hat{r}_M$ based only on the number of trajectories generated by the algorithm. In fact,

the N_c initial trajectories and the $\tilde{J}$ resampled trajectories (generated during the J iterations) are qualitatively different. In practice, the user does not choose the parameter M directly, but rather the number of ensemble members N_c on the one hand, and either the threshold a or the number of iterations J on the other hand. Recall from chapter 6 that the number of initial trajectories N_c governs the convergence of the estimators. Another practical constraint on the choice of N_c is the problem of *extinction*: for some systems, if N_c is too small, all the members of the ensemble become identical after a number of iterations. Extinction is discussed in chapter 6, in section 6.2.1.2. The other parameter (the threshold a or the number of iterations J) selects the type of events we are interested in. Indeed, from (7.13), we obtain an approximate relation between the number of resampled trajectories $\tilde{J}$ and the target return times: we write $\ln \hat{q}_M(a) = \sum_{j=1}^{J} \ln\left(1 - \frac{\ell_j}{N_c}\right)$. For large N_c, this leads to $\ln \hat{q}_M(a) \approx -\sum_{j=1}^{J} \ell_j / N_c \approx -\tilde{J}/N_c$. Targeting rare events with probability $10^{-\beta}$, i.e. return times of order $10^{\beta} T_a$, $\tilde{J}$ is then $\mathcal{O}(N_c \beta)$. This indicates how to choose the number of iterations J in practice. In particular, for rare events, we should often be in the regime $J = N_c \beta$.

To sum up, to compute return time plots $r(a)$, one may either fix the target amplitude a, and run the algorithm for a random number of iterations, until the observable reaches a for all the trajectories. Alternatively, one can fix the target return time $r(a)$, and iterate the algorithm a fixed number of times by choosing $J = N_c \ln(r(a)/T_a)$. In the former case, the prescribed amplitude a needs not correspond to the largest event for which we should estimate the return time, but it will approximately be the case as soon as $N_c \ll J$, i.e. if a is large enough for fixed N_c. Similarly, in the latter case, the largest return time computed by the algorithm will approximately be equal to the prescribed target return time when $N_c \ll J$.

Please note that this method computes the probability to exceed a threshold a, by averaging over trajectories or over K algorithm realisations the sampled value of $q(a)$. This gives an unbiased estimator of $q(a)$, as explained in chapter 6. The standard deviation of this estimator is of order $1/\sqrt{KN_c}$. When computing $r(a)$ through the nonlinear relation $\hat{r}(a) = -T_a / \ln(1 - \hat{q}(a))$, we thus obtain an estimator of $r(a)$ with a bias of order of $1/(KN_c)$ and a standard deviation of order $1/\sqrt{KN_c}$. If however we had made averages over return times among algorithm realisations, then the estimator for each realisation would have been biased with a bias of order $1/N_c$, and the final estimator after K realisations would still be biased with a bias of order $1/N_c$ [105]. Please refer to chapter 6, section 6.1 for a discussion of statistical properties of the AMS and TAMS algorithms.

7.3.2 *Return times for the Ornstein–Uhlenbeck process from the Trajectory Adaptive Multilevel Splitting algorithm*

In this section we consider the Ornstein–Ulhenbeck process $x(t)$ defined in (7.1). Additionally, we define the correlation time τ_c as

$$\tau_c = \lim_{T\to\infty} \int_0^T (\mathbb{E}[x(0)x(T)] - E[x]^2)\mathrm{d}t \tag{7.14}$$

Choosing $\alpha = 1$ and $\epsilon = 1/2$, the correlation time is $\tau_c = 1$ and the variance is $\sigma^2 = 1/2$. We now illustrate the use of the TAMS algorithm for computing the return times $r(a)$ for the variable X_t being larger than a threshold a. This amounts to choose the observable as $A(x) = x$. We use the TAMS algorithm described in chapter 6 with a score function $\xi(x,t) = x$. This choice of score function is motivated by the fact that the optimal score function is nearly independent of time, except on a small boundary layer, as explained on appendix D. Additionally, in one dimension, the level set of x will be the same as the level set of the static committor function.

The algorithm relies on three numerical parameters : the length of the generated trajectories T_a, the maximum threshold value a_{max} and the number of replicas N_c. As explained in chapter 6, section 6.1, the relative error depends on N_c. Additionally, one has to choose $T_a \gg \tau_c$, as explained in section 7.2.1.3. We see empirically that a good trade-off between this requirement and computational burden is to choose trajectories of length T_a equal to a few correlation times.

Figure 7.5 shows the return time plot computed using $N_c = 100$ replicas, $T_a = 5\tau_c$ and $a = 7\sigma$, using the TAMS in conjunction with the methodology described in section 7.3.1. For comparison, figure 7.5 also features the theoretical value, estimated by computing the mean first-passage time [105], as well as the estimate obtained from a direct sampling with the same computational cost as the TAMS run. We see that return times are very well recovered by the TAMS algorithm. Furthermore, figure 7.5 clearly illustrates the computational gain from the TAMS algorithm. Indeed, for the same computational cost as direct sampling, the use of the TAMS algorithm gives access to return times for much rarer events: we can now accurately compute return times on the order of 10^{13}, about seven orders of magnitude larger than direct sampling.

7.4 RETURN TIMES SAMPLED WITH THE GIARDINA-KURCHAN-TAILLEUR-LECOMTE ALGORITHM

In the previous section, we illustrated the use of the TAMS algorithm in conjunction with formula (7.9) to compute return times of large fluctuations of the Ornstein–Ulhenbeck process defined in (7.1). We showed in figure 7.5 that it leads to a significant computational gain

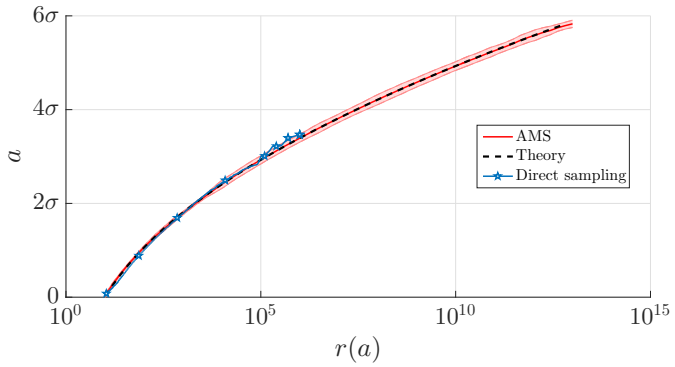


Figure 7.5: Return time plot for a random variable following an Ornstein–Uhlenbeck process (7.1) with $\alpha = 1$ and $\epsilon = 1/2$ ($\sigma = 1/\sqrt{2}$ is the standard deviation). The solid red line represents the estimate obtained using the TAMS with $N_c = 100$ replica, $T_a = 5\tau_c$ and $a = 7\sigma$. The total number of trajectories (both initial and resampled) is $M \approx 2 \times 10^3$ so that the total computational cost is $\mathcal{O}(10^6\tau_c)$. It is compared to the modified block maximum estimator $\hat{r}'_B$ applied to a sample timeseries of length $T_d = 10^6\tau_c$ (blue stars) and to the analytical result, explicited in [105]. The shaded area represents the confidence interval on the estimation of the fluctuation amplitude a, for a fixed value for the return time $r(a)$. It is computed as the empirical mean over the 100 interpolated return time plots originating from the 100 independent realisations of the algorithm.

with respect to the computation of return times based on a long timeseries of the process.

In the present section, we illustrate the computation of return times using the method described in section 7.2.2 for a *time-averaged* observable. Even though it could be done using the TAMS algorithm presented in chapter 6, we instead illustrate the use of a different rare-event algorithm, namely the GKTL algorithm introduced in chapter 4. It is specifically designed to compute large deviations of time-averaged dynamical observables.

7.4.1 *Return times for the time-averaged Ornstein–Uhlenbeck process from the GKTL algorithm*

On the basis of the OU process defined in (7.1), we consider the time averaged position

$$\overline{X}_T(t) = \frac{1}{T}\int_{t-T}^{t} x(s)ds, \ t \in [T, T_a] \tag{7.15}$$

where the position x follows an Ornstein–Uhlenbeck process (7.1) between times 0 and T_a. We denote by σ_T^2 the variance of $\overline{X}_T$ and $\tau_{c,T}$ the correlation time. In this section we illustrate the application of the GKTL algorithm to the computation of the return times $r(a)$

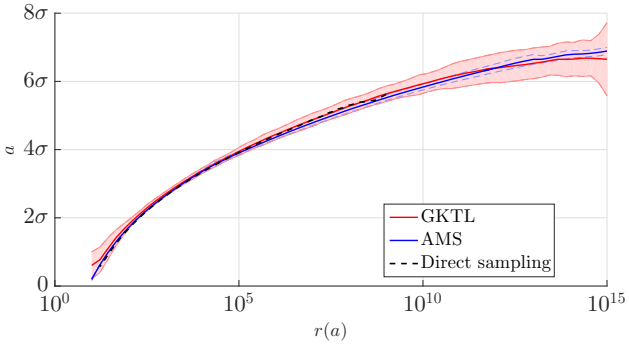


Figure 7.6: Return time plot for the time-averaged Ornstein–Uhlenbeck process $\overline{X}_T$ (7.15) with $\alpha = 1$ and $\epsilon = 1/2$ ($\sigma = 1/\sqrt{2}$ is the standard deviation), estimated from the GKTL algorithm (solid red line) and AMS algorithm (solid blue line). The GKTL algorithm was used with $N_c = 500$ replica, $T_a = 20\tau_c$ and $k = 0.9$. It was repeated $K = 100$ times. The TAMS algorithm was used with $N_c = 100$ replicas, $T_a = 50$ and $a = 6.5\sigma_T$. It was repeated $K = 10$ times. Finally, the dashed black line represents the result of a direct sampling over a timeseries of length $T_d = 10^9\tau_c$. Parameters of both the GKTL and AMS algorithms were chosen so that 100 realisations of the algorithms amount to a computational cost of $\mathcal{O}(10^6\tau_c)$. The cost of the direct sampling is $10^9\tau_c$.The shaded area represents the confidence interval on the estimation of the fluctuation amplitude a, for a fixed value for the return time $r(a)$. It is computed as the empirical mean over the 100 interpolated return time plots originating from the 100 independent realisations of the algorithm.

for $\bar{X}_T$ being larger then a. We make use of the GKTL algorithm with trajectories of duration $T_a > T$, computing the time-averaged position $\bar{X}_T(t)$ for $T \leq t \leq T_a$ as a moving average.

Similarly to the case of the TAMS (see section 7.3.2), the application of the GKTL algorithm depends on three numerical parameters: the number of trajectories N_c, the length of the trajectories T_a and the bias parameter k. The number of trajectories N_c governs the relative error, as explained in chapter 4, and one should use T_a so that $T_a - T \gg \tau_{c,T}$, as explained in section 7.2.1.3. Finally, as for the strength of the selection k, its relation with the amplitude of the generated fluctuations is not known beforehand, and one has to set its value empirically [1].

In Fig. 7.6, we show the return times $r(a)$ for $\overline{X}_T$, with $T = 10\tau_c$, computed from the GKTL algorithm described in chapter 4, following the methodology described in 7.2.2. In order to validate the computation, the estimate obtained from the algorithm is compared to the direct sampling method (7.8). For rare events ($r(a) \gg \tau_{c,T}$), the results from the GKTL algorithm agree well with direct sampling. Further-

1 When the duration of the average is long enough so that a *large deviation regime* is attained, the relation between the value of k and the typical amplitude of the fluctuations generated by the algorithm is known from the Gartner-Ellis theorem. See chapters 1 and 4, as well as Ref. [165] for further details.

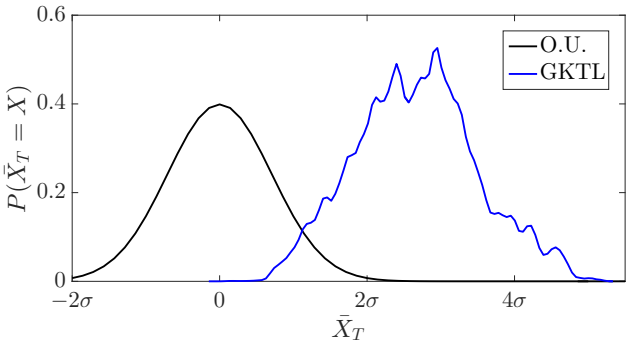


Figure 7.7: PDF of the time-averaged observable $\bar{X}_T$, with $T = 10\tau_c$, for the Ornstein–Uhlenbeck process with $\alpha = 1$ and $\epsilon = 1/2$ ($\sigma = 1/\sqrt{2}$ is the standard deviation): computed from a direct simulation of length $T_d = 10^6$ (black line), and based on the trajectories generated by the GKTL algorithm with 500 replicas, $T_a = 20\tau_c$ and $k = 0.9$ (blue line).

more, the comparison of the computational costs for the two different methods shows the efficiency of the algorithm. Indeed, for direct sampling, the length of the sample trajectory, $10^9\tau_c$ in the case of Fig. 7.6, naturally sets an upper bound on the return times one is able to compute. By contrast, the total cost of the GKTL estimate is $10^6\tau_c$ and one can see in Fig. 7.6 that it allows to reach return times larger by many orders of magnitude. Figure 7.7 shows an estimate of the PDF for $\bar{X}_T$ along the trajectories generated using the GKTL algorithm. Even though importance sampling is performed for the observable $\bar{X}_{T_a}$, the observable averaged over the whole trajectory of length T_a, it better samples the tail of the PDF for $\bar{X}_T$, resulting in better estimation of the corresponding return times.

Figure 7.6 also shows the return time for $\bar{X}_T(t)$ being larger than a computed using the TAMS algorithm, as described in section 7.3.2. We use as a score function the time-averaged observable itself $\xi(t) = \bar{X}_T(t)$, for $T \leq t \leq T_a$. The selection is then done according to the maximum value of $\bar{X}_T(t)$ for each trajectory for $T \leq t \leq T_a$. More precisely, following the notations of section 7.3.2, for iteration j we denote by $\mathcal{Q}_j^\star$ the lowest maximum of $\bar{X}_T$ over the trajectories in the set $\{x_n^{(j)}(t)\}_{0\leq t\leq T_a, 1\leq n\leq N}$. Following the TAMS algorithm described in chapter 6, the l_j new replica are defined by copying the trajectories $x_{n_\ell}^{(j-1)}$ from 0 to the smallest time t such that $\bar{X}_{T,n_\ell}^{(j-1)}(t) > \mathcal{Q}_j^\star$ and simulating the rest of the trajectory from this time to T_a.

The agreement between the two estimates illustrates that the method to compute return times from rare event algorithms proposed in 7.2.2 can be applied to any rare event algorithm suitable for the type of observable under study. Here, while the AMS algorithm allows for computing return times for both the instantaneous and time-averaged observables, the GKTL algorithm is not suited for instantaneous observables.

7.5 APPLICATION: RETURN TIMES FOR EXTREME DRAG FORCES ON AN OBJECT IMMERSED IN A TURBULENT FLOW

In sections 7.3.1 and 7.4, the AMS and GKTL algorithms were shown to greatly mitigate the amount of computational effort required to the computation of return times of rare events for simple one-dimensional stochastic dynamics. A key issue with rare event algorithms is to understand if they are actually useful to compute rare events and their return time for actual complex dynamical systems. In this section, we give a brief illustration that more complex dynamics can be studied. We illustrate the computation of return times using rare event algorithms for a turbulent flow. The possible limitations of rare event algorithms are further discussed in the conclusion.

Unlike simple low-dimensional models, such as the OU process, numerical simulations of turbulent flows of interest for physicists and/or engineers require tremendous computational efforts. As a consequence, direct sampling of rare events based on a long time series is simply unthinkable for such systems. A common practice in the engineering community is to generate synthetic turbulent flows, without resolving explicitly the small scales, to study numerically the physical phenomena of interest [116, 153]. However, the main difficulty is to capture synthetically the correct long-range spatio-temporal correlations of turbulence and such approaches can not capture the essential effects of coherent structures. We suggest here that rare event methods such as the GKTL and the AMS algorithms can be used in order to study extremes in turbulent flows without having to rely on such modelling.

In this section we address the computation of the return time of extreme fluctuations of the drag force applied by a turbulent flow on a immersed obstacle. As explained in chapter 1 and 3, the computation of flow trajectories associated to such extremes is of great interest both for fundamental issues and applied problems, such as reliability assessment for industrial structures. More specifically, we focus here on the averaged drag $F_T(t) = \frac{1}{T}\int_t^{t+T} f_d(\tau)\mathrm{d}\tau$, which corresponds to the averaged sum of the efforts from the flow, projected along the flow direction. See chapter for a mathematical definition of the drag f_d. The length of the averaging window depends on the nature of the application. For instance, it could be related to the typical response time of a material, in order to average out high frequency excitations that have a minor impact on the deformation of the structure. Note that the choice of the observable is arbitrary. For instance, one could choose to study other related physical quantities, such as the lift or torque.

In order to establish a *proof-of-principle* for the computation of return times using rare event algorithms for fluid mechanics problems, we here focus on test flow (2). Recall that this configuration consists in the

flow past a square cylinder in a two-dimensional channel, in which upstream turbulence is generated by the interaction of an incoming steady laminar flow with a grid. Please refer to chapter 2 for a detailed discussion of test flow (2), as well as several illustrations of the flow fields for this configuration. In the following, we used the correlation time, denoted as τ_c, as the unit of time. It is defined as the zero-crossing time of the autocorrelation function of the drag acting on the obstacle. See chapter 2 for a definition and discussion of the correlation time.

7.5.1 *Computation of the reference solution for return times*

As a preliminary step, we compute an estimate of the return times for the fluctuations of the averaged drag F_T acting on the square embedded in test flow (2) from a very long timeseries. This timeseries is computed by mean of a very long simulation of the flow over $T_d = 10^6 \tau_c$. The resulting estimate for the return times will serve as a reference solution. We refer to this estimate as the *direct estimate*. Note that this step is analogous to the computation of a reference solution for the large deviation rate function in chapter 5. The direct estimate is computed based on the method described in section 7.2.1. Note that the maximum return time accessible from a direct estimation, related to the rarest event in the timeseries, is bounded. Indeed, following formula (7.9), the maximum return time accessible by the direct estimate cannot exceed the total duration of the timeseries, here denoted as T_d.

7.5.2 *Computation of return times with the GKTL algorithm and comparison with the reference solution*

Figure 7.8 illustrates the computation of the return times for the drag averaged over $5\tau_c$, using the GKTL algorithm. It shows that the use of the algorithm makes possible the computation of rare events at a much lower computational cost than direct sampling.

More precisely, the algorithm has been applied using $N_c = 128$ replicas simulated over 10 correlation times. The return time plot presented in Fig. 7.8 is based on the data from $K = 10$ repetitions of the algorithm, leading to an overall computational cost of, roughly, 10^4 correlation times. From a direct sampling of similar computational cost, the rarest accessible event has a return time close to the computational cost itself, in this case is $10^4 \tau_c$. Figure 7.8 shows that the use of the GKTL algorithm allows for the computation of return times of much rarer events. The reference estimate has been computed from a time series spanning 10^6 correlation times. For events having a return time close to 5×10^5 correlation times, the computational cost of estimating the return times using the GKTL algorithm is 50 times lower than direct sampling.

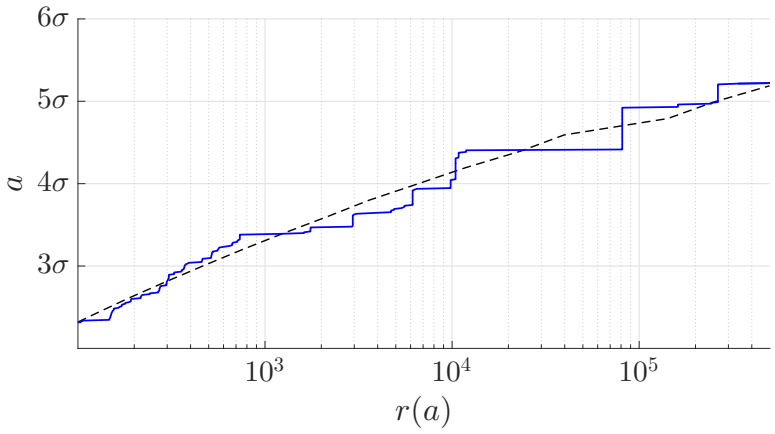


Figure 7.8: Illustration of the computation of return times for the averaged drag over the square obstacle in test flow (2). The averaging window is 5 correlation times. The dashed black line represents the reference return times computed from a timeseries spanning 10^6 correlation times, using (7.8). The solid blue line represents the return times obtained using the GKTL algorithm.

The occurrence of plateaus in Fig. 7.8 is due to the increasing multiplicity of trajectories as the amplitude a increases. Indeed, because of the selection procedure involved in the GKTL algorithm, a subset of trajectories can share the same ancestor. Henceforth, they are likely to differ only by a small time-interval at the end of their whole duration. In such cases, it is common that the maximum over the trajectory is attained in earlier times. As a consequence, this subset of trajectories will contribute the same value to the set of maxima from which return times are computed. This effect is accentuated in the present case of a deterministic system, as it takes some time for copies to separate after being perturbed at a branching point. A straightforward way of mitigating the occurrence of such plateaus is to increase the number of trajectories or/and the number of repetitions of the algorithm. As an illustration, Fig. 7.9 shows the return time plot obtained using 50 repetitions instead of 10 in Fig. 7.8.

7.6 CONCLUSION

In this chapter, we have considered the question of estimating the return time of rare events in dynamical systems. We have compared several estimators, using usual timeseries, generated with direct numerical simulations, as opposed to rare event algorithms. To do so, we generalised the approach relating the return times to the extrema over trajectory blocks. This approach relies on the fact that rare events behave, to a good approximation, like a Poisson process: this allows for the derivation of a simple formula (see (7.7)) for estimating the return times based on block maxima. We slightly improved this for-

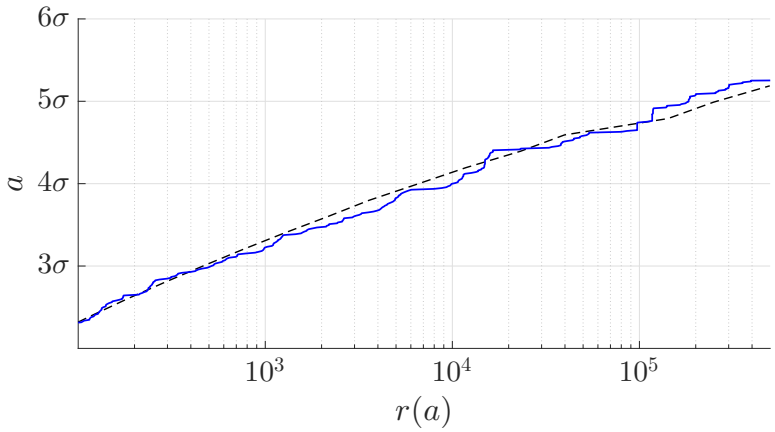


Figure 7.9: Illustration of the computation of return times for the averaged drag over the square obstacle pictured in test flow (2), using 50 repetitions of the GKTL algorithm. The parameters are the same as in Fig. 7.8. This figure illustrates the reduction in the occurrence of plateaus for the return time curve obtained using the GKTL algorithm. The dashed black line represents the reference return times. The solid blue line represents the return times obtained using the GKTL algorithm.

mula (see (7.8)) and further showed that it is possible, provided only minor modifications, to evaluate it with data produced by rare event algorithms. Indeed, the traditional block maximum method consists in dividing a given trajectory in blocks with arbitrary length, larger than the correlation time of the system, and smaller than the return time one seeks to estimate. Moreover, we stressed that there is actually a class of rare event algorithms which yields precisely an ensemble of trajectories exhibiting the rare event more often than direct simulation, together with the probability of observing each member of the ensemble.

Hence, we have generalised the block maximum formula to non-equiprobable trajectory blocks; this allowed us to use directly rare event algorithms, such as the AMS and the GKTL algorithm, to estimate return times for rare events. Using the Ornstein–Uhlenbeck process as an illustration, we showed that the method is easy to use and accurately computes return times in a computationally efficient manner. Indeed, compared to direct sampling, combining the generalised block maximum approach to rare event algorithms allowed for computing return times many orders of magnitude larger, at fixed computational cost. This method does not depend on the dynamics of the system or on the type of observable, as long as a suitable rare event algorithm is selected. This approach paves the way to numerical computation of return times in complex dynamical systems. To showcase the potential of the method, we discussed briefly an application of practical interest: extreme values of the drag force on an object immersed in turbulent

flows. Another example of application given very recently is the study of heat waves [130].

CONCLUSION

Objects immersed in turbulence are subject to randomly varying forces, resulting from their interaction with the surrounding flow. Moreover, such forces can undergo extreme fluctuations, linked to violent variations in the flow, such as abrupt changes of the pressure or velocity. Even though these fluctuations are rare events, their potential impact on the immersed structure make such fluctuations worth be studied.

In this thesis we addressed the numerical simulation of extreme fluctuations of the drag force acting on an immersed object. Although the characterisation of typical drag fluctuations have been addressed in previous works [3, 23, 67, 82, 170], extreme events have rarely been studied.

Direct sampling of extreme drag fluctuations

In a fist part, we applied a *direct sampling* approach. That is, we carried out very long simulations of the interaction of a turbulent upstream flow with a square cylinder, in order to sample extreme fluctuations of the drag force. The flow geometry, introduced in chapter 2, was chosen so that the dynamics could be integrated over a very large number of characteristic times, thus allowing the sampling of a significant number of extreme events.

Extreme fluctuations of the instantaneous drag

From the resulting timeseries, we focused on rare events for which the instantaneous drag f_d reached a given threshold. In practice, we chose the value of the threshold so that the return period of the sampled events is much bigger than the timescale of the typical fluctuations, while still sampling a large number of extreme fluctuations. The simulation and the analysis procedure were implemented so that each fluctuations could be individually re-simulated as a post-processing step, therefore allowing us to look into the details of the corresponding dynamics. We showed that extreme fluctuations of the instantaneous drag are linked to very specific flow configurations, in which vorticity concentrates very close to the downstream side of the square. Such atypical configuration results from the interaction of a large vortical structure with vorticity generated through strong shear along either the top or bottom boundary of the square. The large vortex results from the interaction at earlier times of the upstream turbulent structures with the boundary layer opposite to the boundary where the vorticity is later generated.

Extreme fluctuations of the time-averaged drag

In addition, we also sampled extreme fluctuations of the *time-averaged* drag

$$F_T(t) = \int_t^{t+T} f_d(\tau)\mathrm{d}\tau$$

. Such fluctuations are relevant to many applications, especially when the behaviour of the system under study exhibits a typical response timescale close to T. By contrast with the extremes of the instantaneous drag, we found that the timeseries $\{f_d(\tau)\}_{t\leq\tau\leq t+T}$ corresponding to extreme values of $F_T(t)$ do not display a common structure. In chapter 2, we illustrated that the statistics of the extreme fluctuations of the instantaneous drag f_d are well described by an exponential PDF. Motivated by this observation, in chapter 3 we investigated extreme fluctuations of the time-average for three different stochastic process with different PDF: a Gaussian PDF, a power law PDF and finally an exponential PDF. Furthermore, we designed each process so that it displays a fast decay of the correlation over time. We showed that, for the process with the Gaussian PDF, extreme positive fluctuations of the time average over $10\tau_c$ primarily result from a series of positive independent fluctuations, and very few negative fluctuations. The extreme value of the overall average results from the accumulation of these independent fluctuations. By contrast, we showed that extremes of the time average of the process with the power law PDF exhibit radically different behaviour. Indeed, they mostly result from a few (typically one or two) very high fluctuations of the instantaneous drag f_d, while the rest of the timeseries exhibits typical fluctuations. More importantly, we illustrated that the process with the exponential PDF is a *marginal case*. For this process, we found that extremes of the time-averaged process over $10\tau_c$ can both result from few very high fluctuations or a large number of positive smaller fluctuations.

When investigating extreme events, the direct sampling approach based on very long simulations is limited. Indeed, longer timeseries are necessary to sample a larger number or fluctuations of a fixed amplitude, or rarer events. Moreover, very long integration of the flow dynamics is restricted to simple flows.

Rare events algorithms

In this thesis we explored an alternative approach to rare event sampling in turbulent flows, based on *rare-events algorithms*. We considered the application of two algorithms, the Giardina–Kurchan–Tailleur–Lecomte algorithm [57, 120] and the Adaptive Multilevel Splitting algorithm [27, 140], that are suited to both non-equilibrium

and deterministic dynamics. Both rely on the evolution of a population of trajectories, that can either be replicated or discarded in order to foster the sampling of extreme events. These algorithms have been successfully applied in previous works to various problems involving rather simplified dynamics [21, 54, 57, 158]. In this thesis we addressed their application to turbulent flows, thus representing a great leap in complexity from previous studies. More precisely, we applied the GKTL and AMS algorithms on the basis of the two-dimensional flow for which we studied extreme drag fluctuations from a direct sampling approach. In this way, the results obtained using the algorithms could be compared to very long simulations of the flow.

The Giardina–Kurchan–Tailleur–Lecomte algorithm

In chapter 4, we discussed the implementation of the GKTL algorithm for complex dynamics such as turbulent flows. In addition to its efficiency, the practicality of a rare event algorithm in terms of actual implementation is an important element for applications. In particular, we highlighted that, in the case of complex dynamics, the optimisation of the cloning stage does not lead to a significant increase in performance. Indeed, for cloning periods of the order of the correlation time, the computational cost of simulating the dynamics of the clones overwhelms the computational cost associated with the message passing during the cloning stage.

Using the GKTL algorithm, we computed the *large-deviation rate function* for the drag force acting on a square obstacle. Large deviation rate functions are notoriously difficult to compute with direct approaches, as very long timeseries are required. We showed the GKTL algorithm provides an accurate estimate of the rate function for rare events that otherwise could not be addressed with a direct sampling approach of similar computational cost. However, we highlighted that the efficiency of the algorithm is limited. Indeed, the GKTL estimate of the Scaled Cumulant Generating Function displays linear tails, similarly to direct estimates obtained from finite timeseries. The slope of the tail increases with the number of copies. As a matter of fact, we showed that, as the bias k is increased for a fixed number of copies, the amplitude of the fluctuations sampled by the GKTL algorithm saturates. Moreover, we showed that this limit value increases with the number of copies.

In a last part, we showed that the GKTL algorithm effectively yields a greater number of independent trajectories corresponding to extreme fluctuations of the time-averaged drag, with respect to direct sampling. However, we illustrated that the number of independent trajectories is very small with respect to the total number of trajectories in the population. Furthermore, the number of copies required for sampling a statistically significant amount of independent trajectories grows very rapidly with the bias k. In a way of conclusion, even though the

GKTL algorithm can sample extreme events inaccessible from a direct sampling of similar computational cost, its yield is rather poor.

The Trajectory Adaptive Multilevel Splitting algorithm

The GKTL is restricted to time-averaged observables. In chapter 6, we considered the Adaptive Multilevel Splitting algorithm, in order to sample extreme fluctuations of the *instantaneous* drag. More precisely, we aimed at sampling rare trajectories in which the instantaneous drag f_d reaches a given threshold. As a matter of fact, we developed the Trajectory Adaptive Multilevel Splitting, a modified version of the AMS in which all trajectories have the same *fixed* duration. As a consequence, the TAMS algorithm is particularly easy to use. Its main interest is the computation of *return times*, described in chapter 7, which is considerably simplified with respect to the original AMS algorithm. It yields the probability that an observable reaches a given threshold during a fixed time. On the basis of the Ornstein–Ulhenbeck process, we illustrated that the TAMS is able to probe very rare events for one-dimensional stochastic dynamics. By contrast, we provided evidence that the application of the AMS and TAMS is difficult for complex deterministic dynamics.

Indeed, we showed that choosing the score function as the drag force itself prevents the algorithm to sample new, rarer trajectories. More precisely, duplicated trajectories do not lead to higher fluctuations. Indeed, because the dynamics is deterministic, duplicated trajectories separate from their parent after a finite time of roughly a few correlation time. As a result, when duplicated trajectories are restarted close to a drag maximum, they simply follow the parent trajectory and do not lead to higher fluctuations. Consequently, in this case, the TAMS is unable to generate a diverse ensemble of trajectories that exhibit large fluctuations.

Although the AMS and TAMS algorithms can lead to a significant improvement in the sampling of extremes for stochastic dynamics [5, 138, 139], we showed that its application to complex deterministic dynamics is not straightforward.

The computation of return times

In the last chapter, we addressed the computation of *return times* for rare events, a useful concept for applications. Considering an observable $x(t)$, the return time $r(a)$ was defined in chapter 7 as the the average waiting time for a fluctuation $x \geq a$. We stressed that computing return times cannot be done from the sole knowledge of the probability. As a consequence, we described a methodology to compute return times from long timeseries, based on the *block-maxima* method and the Poisson statistics of rare events. More importantly, we showed that this procedure can straightforwardly be extended to

rare event algorithms such as the GKTL algorithm or the AMS algorithm. Therefore, we illustrated the computation of return times of both instantaneous and time-averaged observables based on the two algorithms and on the Ornstein–Ulhenbeck process. Furthermore, we computed return times for extreme fluctuations of the time-averaged drag acting on a square cylinder using the GKTL algorithm, illustrating that return can be computed for a lower computational cost. However, due to the very few number of independent trajectories, the return time plot obtained with the GKTL estimate displays spurious plateaus. This effect can be reduced by using a larger number of copies, therefore leading to a larger number of independent trajectories.

Perspectives

In this work we laid the groundwork for the development and implementation of rare event algorithms designed for turbulent flows. Using the GKTL algorithm, we computed the statistics and dynamics for extreme fluctuations of the time-averaged drag at a much lower computational cost than the one required by a direct sampling approach. However, our experiments highlight that the efficiency of the sampling might be limited in some cases. Even for values of the bias k for which the algorithm outputs a fairly accurate estimation of the SCGF, we observe that the ensemble of sampled trajectories contains very few independent trajectories, typically less than 5% of the total number of copies. Furthermore, our experiments suggest that a relatively high number of copies is required, with respect for example to the number of copies used in [130].

Even though these limitations have been reported in several applications and analysis of the GKTL algorithm [21, 78, 121, 132], we observed that they particularly alter the performance of the GKTL in our case. By contrast, such limitations have not been reported in [130], which addresses the application of the GKTL algorithm with a simplified model of climate dynamics. The reasons for the difference in efficiency of the algorithm are not clear yet, and will be addressed in further studies. Nonetheless, a possible direction is as follows. The GKTL algorithm can be expected to be efficient for systems in which extreme events are related to the persistence of the system in states for which the observable of interest exhibit an extreme value. In this way, once the system have reached such a state, the GKTL algorithm can select the short lived fluctuations over a cloning period for which the observable of interest persists in taking an extreme value. For instance, this corresponds to the phenomenology in [130]. As a matter of fact, we illustrated in chapter 2 that the qualitative dynamics of the flows concerned in this work is very different. Indeed, flow configurations responsible for extreme values of the drag are swept away by the mean flow. Starting from a flow state in which the drag is extreme, it will

quickly return to typical values. As a result, the flow configuration in a sole region of the computational domain cannot be considered a precursor of extreme fluctuations at future times. consequently, the selection of the trajectories based on the past value of the drag is not a relevant choice.

We believe this is actually why the TAMS fails to sample new trajectories leading to higher fluctuation. The value of the drag force itself at a given instant is a very poor choice of score function. A better score function should probably be a non local function of space, taking both the flow configuration in the vicinity of square and the structure of the upstream turbulence into account. Naturally, the design of such of cost function is very difficult. An alternative route would be to alter the algorithm itself, keeping the drag as a cost function For instance, one could restart resampled trajectories a few correlation times *before* the actual branching point, so that it has time to partially separate until the branching point. If the resampled trajectory actually achieves a higher fluctuations, it is accepted and simulated until the final time. It it does not, it is rejected and another parent trajectory is selected instead. If such a procedure can potentially sample rare trajectories, it probably does not conserve the mathematical properties of the AMS, and further work is necessary in order to design a method which properties can be related to the ones of the AMS.

Part III

APPENDIX

A

THE PIPELBM C++ LIBRARY

This appendix briefly describes the implementation of the numerical simulations for the flow dynamics involved in this work, introduced in chapter 2. It is achieved using a C++ library developed specifically for this project. This appendix does not discuss the details of the implementation. Rather, it describes the motivations behind the development of such a library, its architecture and its main features.

A.1 A SPECIFIC LBM IMPLEMENTATION

The numerical simulation of the test flows presented in chapter 2 is the backbone of this thesis. Throughout this manuscript, it is coupled with rare event algorithms that rely on the integration of the flow over time, as well as the computation of specific observables. Examples include the drag acting on an obstacle immersed in the flow, the velocity, pressure, or vorticity fields at given locations in the computational domain, or the different contributions to the mechanical stresses on the surface of the obstacle. The numerical simulation of the two-dimensional flow in a pipe is an academic problem in computational fluid dynamics, that can surely be solved with the help of available CFD codes.

However, we chose to base the numerical simulations on a code specifically developed for this project. It is specific to the simulation of a two-dimensional flow past obstacle in a channel. The main reason for this choice is the interaction of the simulation of the flow with rare event algorithms. Indeed, this interaction was expected to result in unnecessary difficulties, using complex CFD software. By contrast, the LBM code developed for this work, referred to as `pipeLBM`, was specifically designed to ease the implementation of rare events algorithms.

More precisely, the implementation is *object-oriented*. Object-oriented programming is a programming paradigm which relies on the definition of *objects* that interact with one another [113, 115]. Objects possess both *attributes*, representing data, and *methods* which describe operations on this data. For instance, obstacles in the flow can be described as objects. Examples of possible attributes include the dimension of the obstacle and its position. Furthermore, possible methods are the application of the corresponding boundary conditions for the velocity or the computation of mechanical stresses over the corresponding surface.

Object-oriented programming is particularly adapted to this project, as it leads to the *encapsulation* of the LBM code into objects. From the point of view of the rare event code (or any other code that uses the `pipeLBM` library), the simulation of the flow can be achieved by interacting with objects without having to temper with the actual LBM code. A direct consequence of encapsulation is a clear separation between the code that performs the rare event algorithm and the code that performs the simulation of the dynamics. This leads to better code maintenance, reuse and debugging.

In section A.3, the use of `pipeLBM` is illustrated on a simple example. Then, a validation is presented in two different contexts. First, a convergence analysis is carried out on the basis of the Poiseuille flow. Then, the drag coefficient on a 2D square obstacle is computed and compared to reference values.

A.2 ARCHITECTURE OF THE PIPELBM LIBRARY

The `pipeLBM` library consists of two types of objects, referred to as *classes*. First the `pipeLBM` class is responsible for the implementation of the method, *i.e* the streaming and collision algorithm introduced in chapter 2, section 2.1. Second, the `Obstacle` class describes the implementation of obstacles in the flow, such as square cylinders or grids. It implements the corresponding boundary conditions and provides methods to compute physical observables linked to the obstacle, such as the drag.

A.2.1 *The pipeLBM class*

The `pipeLBM` class describes a particular Lattice Boltzmann model. Is is defined by

- The dimensions of the lattice (or equivalently the lattice spacing).
- The relaxation parameter τ.
- The boundary conditions at the inlet and outlet.
- The forcing amplitude.

The main role of the `pipeLBM` class is to implement the corresponding Lattice Boltzmann algorithm, *i.e.* streaming and collision of the populations. Instances of the `pipeLBM` class provide methods to perform initialisation of the simulation in different contexts, iterations of the dynamics and boundary conditions, as well as several diagnostics on the current state of the flow. The main features of the `pipeLBM` class are outlined below.

A.2.1.1 *Boundary Conditions*

A `pipeLBM` object is responsible for applying the boundary conditions describing the behaviour of the flow along the top and bottom boundaries of the domain, and at both the inlet and outlet. The top and bottom boundaries are modelled as rigid walls with a no-slip boundary condition and are implemented through the halfway bounce-back rule for the boundary populations. Boundary conditions for the inlet and outlet can be set by the user through interaction with the corresponding `pipeLBM` object. By default, the `pipeLBM` class is instantiated with periodic boundaries for the inlet and outlet. For the inlet, a parabolic velocity profile can be chosen instead. It is simply imposed by setting the boundary populations to the corresponding equilibrium value. For the outlet, an open boundary can be setup, based on a linear extrapolation of the velocity filed along the pipe axis (see chapter 2, section 2.3). Additionally, a sponge layer can be defined in order to damp spurious density waves reflected at the boundary and smoothen the flow. It is based on quadratic increase of the viscosity in the vicinity of the outlet.

A.2.1.2 *Initialisation*

A `pipeLBM` simulation can be initialised as follows in several ways :

- Fluid at rest. In this case all populations are set to equilibrium with $\mathbf{u} = 0$ and $\rho = 1$.
- From a binary file describing a given state. This file contains the $D_x \times D_y \times 9$ populations describing the state of the flow.
- From a randomised initial condition. In this case an initial state is generated based on the random superposition of several pre-computed states, using the method described in appendix C. In practice, a `pipeLBM` object must be provided the path to a directory containing a series of binary files corresponding to pre-computed states of the flow.

In addition, a `pipeLBM` object provides a method to perturb the current flow state, based on the method described in chapter 2, section 4.2.1.2, as well as in appendix C. Similarly to the initialisation on a random state, the corresponding method must be given a set of binary files containing pre-computed states of the flow.

A.2.1.3 *Streaming and Collision*

From a user point of view, the streaming/collision/boundary conditions sequence at the core of the Lattice Boltzmann algorithm is encapsulated in a single `pipeLBM` method performing one or several Lattice Boltzmann timesteps. The `pipeLBM` class implements both the

classical Lattice BGK scheme [31] and the central-moment cascaded scheme [143, 144].

A.2.1.4 *Flow diagnostics*

The `pipeLBM` class provides methods to get information on the current state of the flow. Examples include longitudinal and transverse velocities at point, vorticity and stress tensor on obstacles in the flow, see section A.2.2. Flow fields such as velocity and pressure can be written on disk at any time, either in the form of raw binary files or VTK files for visualisation with Paraview.

A.2.2 *The Obstacle class*

This work is concerned about flows past obstacles. In addition to `pipeLBM`, a second class, referred to as `Obstacle`, has been introduced to deal with the definition and interaction of obstacles in the channel with the flow. Its main tasks are :

- Definition of the corresponding solid boundaries
- Application of the boundary schemes
- Computation of efforts applied by the flow on the obstacle, such as the drag or lift.

Naturally, these tasks depend on the nature of the obstacle. As a consequence, `Obstacle` is an *abstract* class that cannot be instantiated as such. However, it provides a structure for further specifications. For instance, classes like `squareObstacle` or `gridObstacle` are inherited from the abstract `Obstacle` class and properly define their own implementation of the boundary conditions or computation of the mechanical stresses. The `Obstacle` and derived classes are independent of the `pipeLBM` class. However, the `pipeLBM` class handles `Obstacle` objects to take the corresponding boundary into account in the LBM scheme. This interaction makes use of *polymorphism*, as `pipeLBM` methods deal with references to Obstacle objects, whether they actually are a square, a grid, or any user-defined shape.

A.3 EXAMPLE : FLOW PAST A SQUARE

This section describes a simple example of an application of the `pipeLBM` library to the computation of the flow around a square obstacle. A parabolic velocity profile is imposed at the inlet and an open boundary condition is implemented at the outlet.

To begin with, the following header must be specified:

```
#include "libpipeLBM.h"
```

A `pipeLBM` object is created, providing the lattice dimension along both space directions:

```
int Dx = 257; // Number of lattice points along the pipe axis
int Dy = 65; // Number of lattice points along the transverse axis
pipeLBM *myLB = new pipeLBM(Dx, Dy); // Instancation of the pipeLBM class
```

By default, the `pipeLBM` class is instantiated with periodic boundaries at the inlet and outlet and parameters leading the a flow with $Re = 10$. This can be changed as follows

```
myLB->setInletBC("poiseuille"); // Parabolic velocity profile at the inlet
myLB->setOutletBC("open"); // Open boundary at the outlet

double Uo = 0.05; // Characteristic velocity, in this case maximum velocity for the inlet profile
dobule tau = 0.501; // Relaxation parameter
double Fo = 0.0; // No volumic forcing in this example
myLB->setParameters(tau, Uo, Fo);
```

Additionally, a sponge layer can be defined. Its definition calls for two values. The first one is the abscissa from which viscosity is progressively increased to a maximum value (typically 10 times the original viscosity of the fluid). From the second value, the viscosity is kept constant at its maximum value until the outlet.

```
int spongeOrigin = (int) 3*(Dx-1)/4+1;
myLB->setSpongeLayer(spongeOrign, Dx); // viscosity is quadratically increased from x=spongeOrigin to the outlet at x=Dx-1.
```

Next, the square cylinder must be defined :

```
int R = 8; // Dimension of the obstacle
int xo = (Dx-1)/2; // x-coordinate for the lower-left corner
   of the square
int yo = (Dy-1)/2 - R/2; // y-coordinate for the lower-left
  corner of the square
squareObstacle *mySquare = new squareObstacle(xo,yo,L);
```

An important step is to define the corresponding geometry in the `pipeLBM` object.

```
myLB->setObstacles(mySquare);
```

Prior to the simulation, the fluid is initialised at rest :

```
myLB->initializeToEquilibrium();
```

Finally, the flow dynamics is integrated. In the following, the drag acting on the square cylinder is computed at each time step and gathered into an array. Additionally, flow fields are periodically written on disk as VTK files for visualisation with Paraview.

```
int To = R/Uo; // Defines the turnover time in units of
  timesteps
int Tf = 10*To // Compute the dynamics over roughly 10
  turnover times

double *drag_timeseries = new double[Tf];
for (int t=0;t<Tf;t++)
{
  myLB->advanceOneTimestep(mySquare); // Performs one LB
  iteration
  myLB->computeStress(mySquare); // Compute various
  mechanical efforts on the square
  drag_timeseries[t] = mySquare.getDrag(); // Gathers the
  current value for the drag on the square

    if(t%100==0)
    {
      myLB->writeVTK(t); // Writes VTK files describing
  velocity and pressure field at iteration t.
    }

  }
```

A.4 TEST CASES

This section illustrates the application of the `pipeLBM` library to two different academic test cases: the Poiseuille flow and the flow past a square obstacle. Results for those configurations can be compared to analytical and experimental solutions, respectively.

A.4.1 *The Poiseuille flow*

The Poiseuille flow denotes the flow in a pipe with periodic boundaries at the inlet and outlet as well as no-slip boundary conditions on the pipe walls. In the laminar regime, the Navier-Stokes equations can be solved exactly. The analytical velocity field is the well-known parabolic profile

$$\mathbf{u}_{th}(y) = -4\frac{U_0}{H^2}(y - H)y\,\mathbf{e}_x, \tag{A.1}$$

with H the diameter of the pipe and $\mathbf{e}_x$ a unit vector aligned with the pipe axis.

The Poiseuille flow was simulated using the `pipeLBM` library for comparison with (A.1) along a section of the pipe. More precisely, we define the error over N grid points as

$$E = \frac{1}{\sqrt{N}}\sqrt{\sum_{i=1}^{N}(\tilde{u}(y) - \tilde{u}_{th})^2}. \tag{A.2}$$

The error can be decomposed into 3 different contributions. First, the lattice Boltzmann method is affected by the usual error coming from the discretisation of time and space: it is second order accurate in both. Second, an additional error term arises from the discretisation of velocity space in the Boltzmann equation. This term scales as the square of the Mach number, defined as the ratio between the typical velocity of the flow and the speed of sound. The Mach number scales like $\Delta x/\Delta t$, where Δt and Δx denote the simulation timestep and lattice spacing, respectively. It leads to

$$E = \mathcal{O}(\Delta x^2) + \mathcal{O}(\Delta t^2) + \mathcal{O}(\Delta x^2/\Delta t^2). \tag{A.3}$$

Please refer to appendix B for a discussion of errors in the LBM. A convergence analysis can then be carried out by refining the grid several times, each time comparing the numerical results to the analytical solution. On order to exhibit the second order convergence in space, the lattice spacing is varied keeping the relaxation time constant (or equivalently, the viscosity in units of the lattice constants Δx and Δt). This approach is often referred to as *diffusive scaling* [92]. Indeed,

when doing so, the timestep scales like Δx^2, and therefore the error in (A.3) reduces to $E = \mathcal{O}(\Delta x^2)$. Figure A.1 displays the results from the convergence analysis. The flow was simulated using the Lattice BGK scheme[31]. The length of the computational domain was set to $3H$ and the relaxation parameter was set to $\tau = 0.55$.

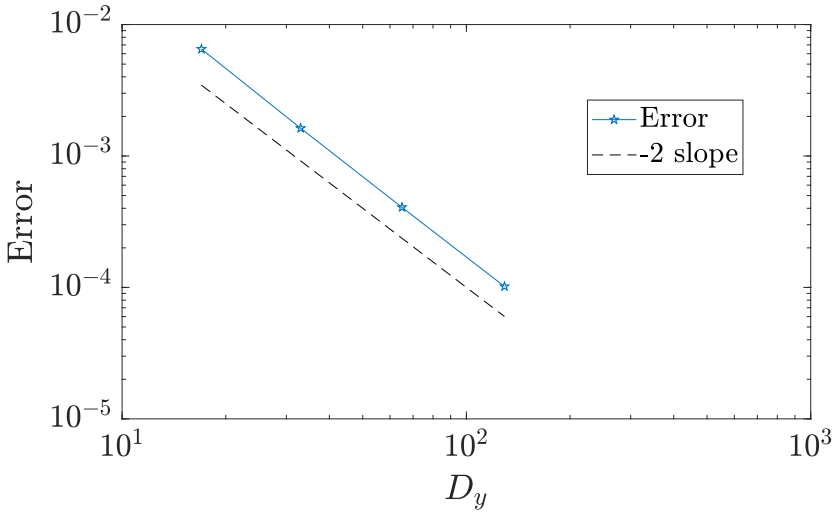


Figure A.1: Error (A.2) computed over a cross section of the channel, for $x = L/2$. The numerical estimate for the velocity field is compared to the analytical profile (A.1).
The error was computed for different numbers of lattice nodes along the transverse direction : $D_y = 17$, $D_y = 33$, $D_y = 65$, $D_y = 129$. As expected, it scales like the square of the inverse of the number of grid points.

A.4.2 *The laminar flow past a square cylinder*

The second test case is the numerical simulation of the laminar flow past a square cylinder immersed in the pipe. For this fairly simple flow, reference values for the drag coefficient of the square are available in the literature []. The flow is solved using the `pipeLBM` library with a parabolic velocity profile imposed at the inlet and open boundary at the outlet. The dimension of the square is denoted as R and the blockage ratio $Bl = H/L$ is set to $Bl = 8$. See figure 2.2 on page 26 for a sketch of the computational domain. Note that in the case of figure 2.2 the boundary conditions at the inlet and outlet of the domain are periodic, which is not the case here. The central-moment cascaded LBM scheme is employed. Finally, the section of the channel is discretised using 129 lattice nodes. The flow is simulated until the stationary regime for several values of the Reynolds number. The velocity is kept constant to $u = 10^{-2}$ (lattice units) and the relaxation parameter τ is varied accordingly.

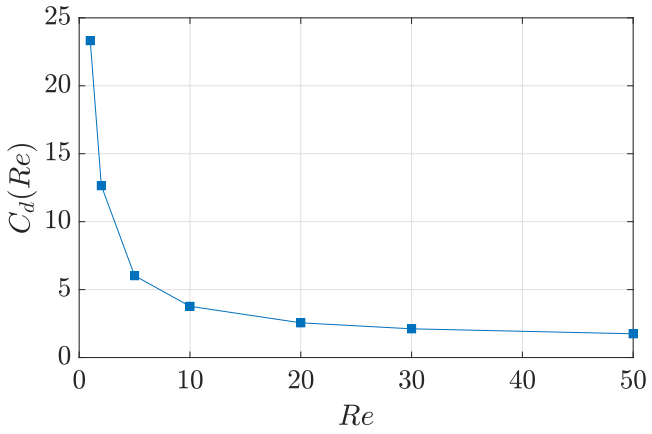


Figure A.2: Drag coefficient as a function of the Reynolds number. The estimate from pipeLBM is compared to reference values from[], showing good agreement. $Re = 50$, $U_0 = 0.01$, $D_y = 129$, $D_x = 385$, $R = 16$.

The drag coefficient for the square is a function of the Reynolds number only and is defined as

$$C_d = \frac{f_d}{\frac{1}{2}\rho R U_0^2}$$

with f_d the drag force acting on the square, ρ the density of the fluid, R the dimension of the square and U_0 the typical velocity of the fluid impacting the obstacle (in this case the inlet velocity). Comparison between the reference values presented in [20] and the result from the `pipeLBM` implementation is displayed in figure A.2, showing good agreement.

B

THE LATTICE BOLTZMANN METHOD

In this appendix extends section 2.1. We give a brief introduction the the Lattice Boltzmann Method (LBM), with a focus on its practical application. In section B.0.1 we give an overview of the genesis of the LBM. Then, in section B.0.2 we introduce the Lattice Boltzmann Equation (LBE), a discrete analog to the Boltzmann equation that describes the dynamics of dilute gases. The main properties of the LBM are then outlined. Finally, in section B.0.3 we discuss practical aspects of LBM simulations: algorithmic procedure, boundary conditions and choice of parameters.

B.0.1 *Lattice Gas Cellular Automaton*

The LBM originates from Lattice Gas Cellular Automata [75, 76]. A Lattice Gas Cellular Automata describes the dynamics of an ensemble of particles evolving on a discrete lattice according to a set of rules. Typically, these rules describe the propagation of the particles from their current location to neighbouring nodes. Additionally, this propagation is constrained by rules describing *collisions*, *i.e.* configurations for which two or more particles are located on the same node. The motivation driving the development of Lattice Gas models was the description of the macroscopic behaviour of fluid on the basis of very simplified models, which simple microscopic rules verify basic properties [50, 135]. Although Lattice Gas Cellular Automata (LGCA) were regarded with great enthusiasm at first, they revealed to be hindered by several limitations and spurious effects, that prevented them from being useful in concrete applications. Nonetheless, the LGCA laid the groundwork for the formulation of the LBM, which emerged from LGCA in the early 90s [80, 81, 108]. For a review of the early developments of the LBM from LGCA, see [154, 155].

B.0.2 *The Lattice Boltzmann Equation*

The LBE can be thought of as a discrete analog of the Boltzmann equation, that describes the dynamics of a dilute gas from a statistical approach. More precisely, time, space as well as velocity space are discretised. In this way, the gas particles propagate on the nodes of the lattice along a discrete set of velocities, illustrated in figure 2.1. More precisely, we denotes by $f_i(\mathbf{x}, t)$ the amount of mass (per unit volume)

carried by the particles moving (with speed $\mathbf{c}_i$) in the i^{ts}-direction at position $\mathbf{x}$ and time t. The LBE writes

$$f_i(\mathbf{x} + \mathbf{c}_i \Delta t, t + \Delta t) = f_i(\mathbf{x}, t) + \Omega_i(\mathbf{x}, t) \qquad \text{for } i = 0, \ ...8. \tag{B.1}$$

where $\Omega_i(\mathbf{x}, t)$ is the *collision operator*, sometimes referred to as the *collision kernel*. The left-hand side of equation (B.1) describes the propagation of particles across the lattice, often referred to as *steaming*. The right-hand side describes the effect of collisions between particles.

Using the LBM terminology, the f_i's are referred to as *populations*. For two-dimensional flows, the most common choice for the discretisation of velocity space is the so-called D2Q9 lattice, illustrated in figure 2.1. In this representation, the state of each lattice node $\mathbf{x}$ at time t is represented by a set of 9 populations $\{f_i(\mathbf{x}, t)\}_{0 \leq t \leq 8}$ that correspond to 9 possible velocity directions. Note that we included the null velocity as $i = 0$, which describes particles that do not move. Similarly to kinetic theory [30], macroscopic quantities such as the velocity field or density field can be recovered as moments over the populations:

$$\rho = \sum_i f_i \tag{B.2}$$

$$\rho \mathbf{u} = \sum_i f_i \mathbf{c}_i \tag{B.3}$$

$$\mathbf{S} = -\frac{1}{2\tau c_s^2 \rho} \sum_i \mathbf{c}_i \mathbf{c}_i (f_i - f_i^{eq}) \tag{B.4}$$

$$\tag{B.5}$$

Where ρ, $\mathbf{u}$ and $\mathbf{S}$ are the density, velocity and strain-rate on the lattice node, respectively.

Importantly, the mesoscopic behaviour of the LBE can be shown to yield the Navier-Stokes equations at a macroscopic level. This can be achieved through a Chapman-Enskog analysis[1] [31, 92], a multiscale method based on the expansion of the populations f_i around local equilibrium. The pressure field $p(\mathbf{x}, t)$ can be computed from, the density field $\rho(\mathbf{x}, t)$ through the ideal equation of state

$$p = c_s^2 \rho \tag{B.6}$$

where c_s may be interpreted as the speed of sound. The Chapman-Enskog analysis recovers the Navier-Stokes equations in the weak-compressibility regime, *i.e.* $Ma = u/c_s \ll 1$ where Ma denotes the Mach number defined as the ratio between the typical velocity and the speed of sound. In this regime, the incompressible Navier-Stokes equation are recovered up to compressibility errors scaling like Ma^2.

Note that the weak-compressibility approximation has an important practical impact: the pressure can be computed locally by summing

1 The Chapman-Enskog analysis was originally published in 1917 [30] as a way to derive macroscopic equations from Boltzmann's equation.

the contribution of the populations $\{f_i\}_{0\leq i\leq 8}$ on a given lattice node. As a result, there is no need for the resolution of a Poisson equation, as in conventional methods [46]. Additionally, the Chapman-Enskog analysis yields an expression for the kinematic viscosity ν, as a function of the properties of the collision operator Ω.

B.0.3 *The LBM in practice*

B.0.3.1 *Lattice Bhatnagar–Gross–Krook*

There are actually many variants of the LBM, depending on the choice for the collision operator in (B.1) Its simplest formulation is referred to as the LBGK operator [129]:

$$\Omega_i = -\frac{f_i - f_i^{eq}}{\tau}\Delta t, \tag{B.7}$$

where τ is a timescale that describes the relaxation towards equilibrium. This equilibrium is described by the *equilibrium populations* f_i^{eq}

$$f_i^{eq} = w_i\rho\left(1 + \frac{\mathbf{u}\cdot\mathbf{c}_i}{c_s^2} + \frac{(\mathbf{u}\cdot\mathbf{c}_i)^2}{2c_s^4} - \frac{\mathbf{u}\cdot\mathbf{u}}{2c_s^2}\right), \tag{B.8}$$

where the $\{w_i\}_{0\leq i\leq 8}$ are a set of weights which value depending the choice of the lattice geometry [155, 176].

In this context, the kinematic viscosity ν can be related to the relaxation time τ [155]:

$$\nu = \frac{1}{3}(\tau - \Delta/2) \tag{B.9}$$

B.0.3.2 *The Lattice Boltzmann algorithm: streaming and collision*

Under the LBGK approximation, the LBE reads

$$f_i(\mathbf{x}+\mathbf{c}_i\Delta t, t+\Delta t) = f_i(\mathbf{x},t) + \frac{\Delta t}{\tau}(f_i(\mathbf{x},t) - f_i^{eq}(\mathbf{x},t)) \qquad \text{for } i = 0, \ldots 8. \tag{B.10}$$

The algorithmic procedure of the LBM is very simple. Indeed, it consists of two distinct parts, described as follows.

COLLISION On each node of the lattice, the collision operator is applied on the populations:

$$f_i^{coll}(\mathbf{x},t) = f_i(\mathbf{x},t) + \frac{\Delta t}{\tau}(f_i(\mathbf{x},t) - f_i^{eq}(\mathbf{x},t)) \qquad \text{for } i = 0, \ldots 8, \tag{B.11}$$

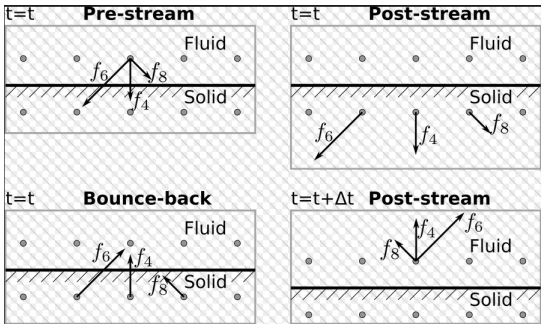


Figure B.1: Illustration of bounce-back rule for no-slip boundaries. Incoming populations are streamed outside the fluid domain, before being streamed back from the direction they came from.

where $\{f_i^{coll}(\mathbf{x}, t)\}_{0 \leq i \leq 8}$ denote the *post-collision* populations on the node $\mathbf{x}$ at time t. Note that this operation is local, indeed, it does not require information from the other node. As such, the collision step is particularly well-suited for parallel architectures.

STREAMING The second step of the LBM is the *streaming* of the post-collision populations to neighbouring nodes, according to (B.10):

$$f_i(\mathbf{x} + \mathbf{c}_i \Delta t, t + \Delta t) = f_i^{coll}(\mathbf{x}, t). \tag{B.12}$$

Iterations of the Lattice Boltzmann Method are therefore amenable to the collision step, followed by the streaming of the population on the lattice.

B.0.3.3 *Boundary conditions*

Within the Lattice Boltzmann framework, boundary conditions can be imposed through specific collision rules for the populations at boundary nodes. For instance, no-slip boundaries are commonly achieved by means of the so-called *Bounce-Back rule*, that is the reflection of a subset of the populations in order to ensure a zero momentum. It is illustrated in figure B.1. Velocity or pressure boundary conditions can be imposed in many ways [100]. However, the different methods are not equal in terms of stability and/or accuracy, and the choice of the boundary schemes therefore depend on the application.

In this work, no-slip boundaries are imposed following the halfway bounce-back rule [155], illustrated in figure B.1. In addition, velocity boundaries are imposed by computing the strain-rate tensor on the boundary nodes, using finite differences. From the knowledge of the strain rate tensor, it is possible to compute a set of *regularised* populations $\{\tilde{f}_i\}_{0 \leq i \leq 8}$ that are consistent with the macroscopic dynamics of the flow [99, 100]. This specific approach is known in the literature as the Finite-Difference regularised boundary conditions [100, 152].

B.0.3.4 *Simulation parameters and lattice units*

A Lattice Boltzmann simulation relies on three parameters: the lattice spacing Δx, the simulation timestep Δt and the relaxation parameter $\tilde{\tau}$. These three parameters are not independent, indeed, relation (B.9) can equivalently be written as

$$\nu = \frac{1}{3}\left(\tilde{\tau} - \frac{1}{2}\right)\frac{\Delta x^2}{\Delta t}, \tag{B.13}$$

where we used that $c_s^2 = (1/3)(\Delta x^2/\Delta t^2)$ [155] and defined the non-dimensional relaxation parameter $\tilde{\tau} = \tau/\Delta t$.

In most cases, it is convenient to work in the system of units in which $\Delta x = 1$ and $\Delta t = 1$. This is motivated by the *similarity principle* in fluid dynamics, which states that two flow having the same geometry and Reynolds number

$$Re = \frac{u_0 L}{\nu} \tag{B.14}$$

have similar dynamics. In the above u_0 and L denote the characteristic velocity and length of the problem, respectively. In lattice units, equation (B.13) reduces to

$$\tilde{\nu} = \frac{1}{3}\left(\tilde{\tau} - \frac{1}{2}\right). \tag{B.15}$$

In many applications the velocity u_0 can be considered an additional parameter of the simulation.

What happens if one chooses $\tilde{\tau} = 1/2$? Infinite Reynolds number ? As the viscosity is decreased, the lattice Reynolds number $u\Delta x/\nu$ is increased, where u denote the typical velocity fluctuations at the scale Δx. As a result, important numerical errors result from the fact large gradients at the lattice scale at not well resolved. Note that this is not specific to the LBM, but is a cross-cutting issue in all numerical simulations of turbulent flows. In the limit $\tau \to 1/2$ the LBM scheme because highly unstable. In practice, this means that small errors can rapidly diverge, *i.e.* the simulation "explodes".

In addition, because of the weak-compressibility approximation, one must ensure that the typical velocity u_0 in the computational domain is small with respect to the Mach number. In lattice units, it writes

$$\tilde{u}_0 \ll 1/\sqrt{(3)} \tag{B.16}$$

ERRORS In the incompressible flow limit, the compressibility error grows like $\mathcal{O}(Ma^2) = \mathcal{O}(\tilde{u}_0)$. For a fixed physical velocity u_0, the lattice viscosity $\tilde{u}_0 = u_0(\Delta t/\Delta x)$ increases with $\Delta t/\Delta x$. Therefore, the compressibility error grow like $\mathcal{O}(\Delta t^2/\Delta x^2)$.

In addition, it can be shown that the spatial discretisation error is $\mathcal{O}(\Delta x^2)$ and that the time discretisation error is $\mathcal{O}(\Delta t^2)$ [92]. As a result, the choice of parameters can be difficult. For instance, reducing the lattice spacing Δx does not necessarily decrease the overall error. Indeed, even though the spatial discretisation error is reduced, the compressibility error is increased.

There are various ways of setting the parameters for a LBM simulation. For instance, one could set the lattice velocity $\tilde{u}_0$ first, ensuring that $Ma \ll 1$. For a given target Reynolds number $\tilde{u}_0 \tilde{L}/\tilde{\nu}$, the choice of Δx (or equivalently the size N of the lattice) constrains the choice of the relaxation parameter $\tilde{\tau} = 3\nu + 1/2$. Indeed,

$$Re = \frac{\tilde{u}_0 (N-1)}{3\nu + 1/2}, \tag{B.17}$$

where N is the number of lattice nodes along the relevant dimension. Equivalently, fixing $\tilde{\tau}$ constrains the choice of Δx (or N). Additionally, the timestep Δt can be expressed in units of the *turnover time* L/u_0:

$$\Delta t = \frac{\tilde{u}_0}{\tilde{L}} \tag{B.18}$$

C

INTRODUCTION OF A PERTURBATION IN THE FLOW STATE WITHIN THE LATTICE BOLTZMANN METHOD

In this appendix we explicit the construction of the perturbation introduced within the GKTL and GKTL algorithms, so that cloned trajectories separate over time. The application of this perturbation is described in chapter 4, section 4.2.1.2.

The numerical model for the flow dynamics is the Lattice BGK equation, presented in chapter 2, section 2.1:

$$f_i(\mathbf{x} + \mathbf{c}_i\Delta t, t + \Delta t) \quad = \quad f_i(\mathbf{x}, t) - \frac{1}{\tau}\left[f_i(\mathbf{x}, t) - f_i^{eq}(\mathbf{x}, t)\right] \qquad \text{for } i = 0, ...8. \tag{C.1}$$

For the sake of simplicity and without loss of generality, we do not introduce any forcing term in (C.1). The equilibrium populations f_i^{eq} are derived from a second order truncation of the Maxwell-Boltzmann equilibrium distribution. It reads

$$f_i^{eq} = w_i\rho + w_i\rho\frac{\mathbf{u}\cdot\mathbf{c}_i}{c_s^2} + \frac{(\mathbf{u}\cdot\mathbf{c}_i)^2}{2c_s^4} - \frac{\mathbf{u}^2}{2c_s^2} \qquad \text{for } i = 0, ...8. \tag{C.2}$$

The lattice weights w_i are numerical constants and $\mathbf{c}_i$ stands for the i^{th} lattice vector, as pictured in figure 2.1. The macroscopic fields ρ and $\mathbf{u}$ are computed as a local average over the populations:

$$\rho = \sum_{i=1}^{8} f_i \tag{C.3}$$

$$\rho\mathbf{u} = \sum_{i=1}^{8} \mathbf{c}_i f_i \tag{C.4}$$

Formally, (C.1) rewrites as

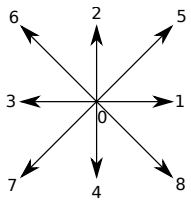


Figure C.1: Sketch of a lattice node

$$f_i(\mathbf{x}+\mathbf{c}_i\Delta t, t+\Delta t) - \left(1-\frac{1}{\tau}\right) f_i(\mathbf{x},t) - \frac{w_i}{\tau}\sum_{k=1}^{8} f_k = \mathcal{O}\left(\frac{\mathbf{u}}{c_s}\right). \quad \text{(C.5)}$$

The left hand side of (C.5) is linear, which entails that a linear combination of solutions is itself solution. The nonlinear part is located in the order one part of the equilibrium populations (C.2) and its amplitude is of the order of Ma. In order to construct a perturbation, the non-linear part of the LBGK equation (C.1) is neglected and the perturbation is built from the linear combination of N pre-computed solutions $\{f_i^{(n)},\ 1 \leq i \leq 8\}_{1\leq n\leq N}$ of (C.1).

$$f_i' = \sum_{n=1}^{N} a_n f_i^{(n)}, \quad \text{(C.6)}$$

where $\{a_n\}_{1\leq n\leq N}$ is a collection of random weights drawn uniformly in $[0;1[$. This is motivated y the fact that, throughout this work, we are interested in incompressible flow for which $Ma \ll 1$. As a result, the error on the perturbed state is expected to be of order one in the Mach number, that is, to scale as u/c_s.

In order to tune the intensity of the perturbation, we introduce the norm $||\cdot||$

$$||f|| = \left(\int |f(\mathbf{x},t)|^2 d\mathbf{x}\right)^{1/2} \quad \text{with} \quad |f| = \left(\sum_{k=1}^{9} f_k^2\right)^{1/2}.$$

A perturbed state $\{\tilde{f}_i(\mathbf{x},t)\}_{1\leq i\leq 8}$ is created from a state $\{f_i^{(0)}(\mathbf{x},t)\}_{1\leq i\leq 8}$ as follows

$$\tilde{f}_i = f_i^{(0)} + \epsilon \frac{f_i'}{||f'||} \qquad \text{for } i = 0,\ ...8. \quad \text{(C.7)}$$

Note that we then have by construction $||\tilde{f} - f^{(0)}||(t=0) = \epsilon$, where we took the origin of time as the instant at which the perturbation was introduced.

The last step consists in renormalizing the populations so that no mass is added in the system. Indeed, at each lattice node, the density perturbation is

$$\rho'(\mathbf{x}) = \sum_{i=0}^{8} \epsilon \frac{f_i'(\mathbf{x})}{||f_i'(\mathbf{x})||} = \frac{\epsilon}{||f'||}\sum_{n=1}^{N}\sum_{i=0}^{8} f_i^{(n)}(\mathbf{x}).$$

The total added mass is therefore

$$\begin{aligned} M' = \int \rho'(\mathbf{x},t)\mathrm{d}\mathbf{x} &= \frac{\epsilon}{||f'||}\sum_{n=1}^{N}\int \underbrace{\sum_{i=0}^{8} f_i^n(\mathbf{x})}_{\rho^{(n)}(\mathbf{x})}\,\mathrm{d}\mathbf{x} \\ &= NM^{(0)}\frac{\epsilon}{||f'||}. \end{aligned} \tag{C.8}$$

The sum under the integral in (C.8) is the density at the lattice node $\mathbf{x}$ corresponding to the flow state $\{f_i^{(n)}(\mathbf{x})\}_{0\leq i\leq 8}$. By conservation of the mass, the mass corresponding to such states must be equal to the mass of the system, as they are solution of the numerical model (2.1). After the perturbation, the total mass of the system is then

$$\tilde{M} = \left(1 + N\frac{\epsilon}{||f'||}\right) M^{(0)}. \tag{C.9}$$

To guarantee mass conservation, the $\{\tilde{f}_i(\mathbf{x})\}_{0\leq i\leq 8}$ must therefore be re-scaled as follows

$$\tilde{f}_i(\mathbf{x}) \longrightarrow \frac{1}{\left(1 + N\frac{\epsilon}{||f'||}\right)}\tilde{f}_i(\mathbf{x}) \qquad \text{for } i = 0,\ ...8. \tag{C.10}$$

D

THE OPTIMAL SCORE FUNCTION

This section is a theoretical discussion of the properties of the optimal score function; it may be skipped by readers who are only interested in the application of the TAMS algorithm for computing return times.

As mentioned in section 6.1, the statistical properties, and in particular the variance of the AMS estimator $\hat{q}(a)$, depend on the choice of the score function ξ. The variance is minimal for a particular choice of the score function, sometimes referred to as the *committor*. In a very generic manner, for the AMS algorithm, it is given by $\bar{\xi} = \mathbb{P}[\tau_{\mathcal{B}} < \tau_{\mathcal{A}}]$. In the specific case of the TAMS algorithm, the optimal score function takes the form:

$$\bar{\xi}(t, x; T_a, a) = \mathbb{P}_{x,t}\left[\max_{t \le s \le T_a} O[X, s] > a\right], \tag{D.1}$$

for all $(t, x) \in [0, T_a] \times \mathbb{R}^d$, where we denote $\mathbb{P}_{x,t}$ the probability over the process initialised at position x at time t, and the threshold a and trajectory duration T_a are fixed parameters. Note that the optimal score function depends both on time and space. Of course, we cannot use this score function in practice, because it is exactly what we are trying to compute. Indeed, as mentioned above, the algorithm ultimately provides an estimate of the probability $q(a) = \bar{\xi}(0, x_0; T_a, a)$. Nevertheless, a crucial point to implement the AMS algorithm is to choose a score function that provides a good approximation of the committor. In practical applications, constructing the score function will often be based on heuristic considerations, but it may also be useful to have theoretical results about the optimal score function.

Here, we want to explain the qualitative properties of the time-dependent committor (D.1) specific to the TAMS algorithm. For simplicity, we shall only discuss the case of an instantaneous observable: $O[X, t] = A(X_t)$. Moreover, for the precision of the discussion, we assume that the stochastic process X solves the stochastic differential equation $dX_t = b(X_t)dt + \sqrt{2\epsilon}dW_t$, where b is a vector field with a single fixed-point $x_\star$. We further assume that the basin of attraction of $x_\star$ is the full phase space. With this hypothesis, the invariant measure of the diffusion is concentrated close to the attractor $x_\star$ when $\epsilon \ll 1$. Let us assume that the set $\mathcal{C} = \{x \mid A(x) \le 0\}$ is a neighbourhood of $x_\star$ on which most of the invariant measure mass is concentrated. We call $\mathcal{C}$ the attractor. The target set $\mathcal{D} = \{x \mid A(x) \ge a\}$ is similarly defined. The hitting times for the sets $\mathcal{C}$ and $\mathcal{D}$ are the random variables given by $\tau_\star = \inf\{t > 0 \mid A(X_t) \le 0\}$ and $\tau_a = \inf\{t > 0 \mid A(X_t) \ge a\}$, respectively, where the process is started from a point x at time $t = 0$,

such that $0 \leq A(x) \leq a$. We finally define the static committor $\xi_0(x,a) \equiv \mathbb{P}_{x,0}[\tau_a < \tau_\star]$. The aim of the following discussion is to explain the relation between the time-dependent committor (D.1) and the static committor $\xi_0(x,a)$.

On the one hand, the time-dependent committor $\bar{\xi}$ satisfies a backward Fokker-Planck equation

$$\frac{\partial \bar{\xi}}{\partial t} = -L[\bar{\xi}], \quad \text{with } L = b_i \frac{\partial}{\partial x_i} + \epsilon \frac{\partial^2}{\partial x_i^2}, \tag{D.2}$$

in the domain $A^{-1}([0,a]) \subset \mathbb{R}^d$ with boundary condition $\bar{\xi}(t,x;T_a,a) = 1$ for $x \in \partial\mathcal{D}$, and final condition $\bar{\xi}(T_a,x;T_a,a) = 0$. This follows directly from the backward Fokker-Planck equation for the transition probability $P(y,s|x,t)$, and the fact that, with an absorbing boundary condition on $\partial\mathcal{D}$, $\bar{\xi}(t,x;T_a,a) = 1 - \int dy P(y,T_a|x,t)$. Note that when $T_a - t \gg r(a)$, $\bar{\xi}(t,x;T_a,a) \approx 1$ everywhere ($\bar{\xi}$ converges to 1). On the other hand, $\xi_0(x,a)$ satisfies $L[\xi_0] = 0$, but with different boundary conditions: $\xi_0(x,a) = 1$ if $x \in \partial\mathcal{D}$ and $\xi_0(x,a) = 0$ if $x \in \partial\mathcal{C}$. In the next paragraph, we argue that when $T_a - t$ is much smaller than $r(a)$, the time-dependent committor $\bar{\xi}(t,x;T_a,a)$ given by (D.1) is well approximated by the static committor $\xi_0(x,a)$, except in two boundary layers: a spatial one of size ϵ for x close to the attractor, and a temporal one of size τ_c for t close to T_a.

Using the notations of chapter 6, the events $\{\tau_{\mathcal{B}} < \tau_{\mathcal{A}}\}$ can be decomposed into the disjoint union of events for which the observable reaches the threshold a before or after hitting 0. The typical time for X to reach $\mathcal{C}$ is the correlation time τ_c. If we assume that $T_a - t \gg \tau_c$, we have the approximation $\bar{\xi}(t,x;T_a,a) \simeq \xi_0(x,a) + [1-\xi_0(x,a)]\bar{\xi}(t,x_\star;T_a,a)$ (we have used here the approximations $\bar{\xi}(\tau_\star,y;T_a,a) \simeq \bar{\xi}(\tau_\star,x_\star;T_a,a)$ for any $y \in \partial\mathcal{C}$, and $\bar{\xi}(\tau_\star,x_\star;T_a,a) \simeq \bar{\xi}(t,x_\star;T_a,a)$). Moreover, when $T_a - t \ll r(a)$, the Poisson approximation $\bar{\xi}(t,x_\star;T_a,a) \simeq (T_a-t)/r(a)$ holds. To sum up, in the limit $\tau_c \ll T_a - t \ll r(a)$,

$$\bar{\xi}(t,x;T_a,a) \simeq \xi_0(x,a) + \frac{T_a - t}{r(a)}[1-\xi_0(x,a)]. \tag{D.3}$$

Let us now introduce the quasipotential V. We note that

$$\xi_0(x,a) \underset{\epsilon\to 0}{\asymp} \exp(-(\inf_{y\in A^{-1}(\{a\})} V(y) - V(x))/\epsilon),$$

while

$$r(a) \underset{\epsilon\to 0}{\asymp} \exp((\inf_{y\in A^{-1}(\{a\})} V(y))/\epsilon).$$

We can thus conclude that $\xi_0(x,a)$ dominates this expression for all x except in a region of size ϵ around the attractor $x_\star$.

As a conclusion, when $T_a - t$ is much smaller than $r(a)$, the time-dependent committor $\bar{\xi}(t,x;T_a,a)$ (D.1) is well approximated by the

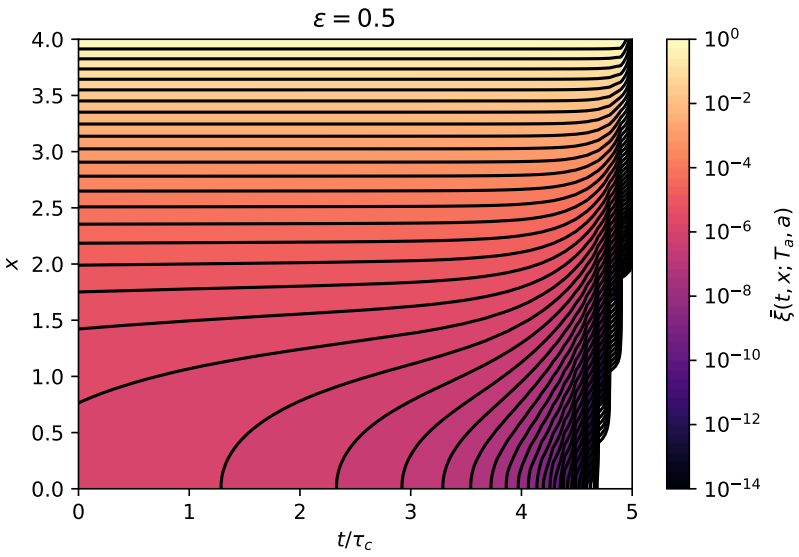


Figure D.1: Contour lines of the time-dependent committor $\bar{\xi}(t, x; T_a, a)$ for the Ornstein–Uhlenbeck process (with $\alpha = 1, \epsilon = 1/2$; in particular $\tau_c = 1$), obtained by solving numerically the backward Fokker-Planck equation (D.2), with $a = 4, T_a = 5$.

static committor $\xi_0(x, a)$, except in two boundary layers: a spatial one of size ϵ for x close to the attractor, and a temporal one of size τ_c for t close to T_a. This is illustrated in Fig. D.1, representing the committor $\bar{\xi}(t, x; T_a, a)$ for the Ornstein–Uhlenbeck process (with $\alpha = 1, \epsilon = 1/2$), obtained by solving numerically the backward Fokker-Planck equation (D.2), with $a = 4, T_a = 5$. Courtesy of Corentin Herbert.

E

THE LIBTAMS LIBRARY

This appendix provides a brief description of the `libTAMS` C++ library, that I developed in the context of this thesis. As of June 2018, the `libTAMS` library is a work in progress, and is in the alpha-test phase. As such, the source code has not been released yet, and the estimated release date of the first beta version is early 2019. Nonetheless, all the TAMS experiments presented in chapter 6 are based on code written using early versions of the `libTAMS` library.

The objective of the `libTAMS` library is to facilitate both the design and writing of codes implementing the TAMS algorithm, introduced in chapter 6. It provides an *object oriented* framework, articulated around several classes. These classes provide an ensemble of methods that can be used to implement TAMS code with any type of dynamics, wether the score function defined as an instantaneous observable or as a time-average. We stress that libTAMS **is not** an implementation of the TAMS algorithm, but rather a framework in which a user can easily write a TAMS code with the dynamics of its own choice.

The development of the `libTAMS` library is motivated by two remarks:

- The TAMS algorithm, described in section 6.2.1 is is completely independent of the dynamics under study. Moreover, it is very simple, and its structure is not changed whether the cost function is based on an instantaneous observable or a time-average.

- Paradoxically, the implementation of TAMS code is specific to the dynamics under study. For instance, for a cost function defined as a time-average of an observable, the parts of the code that perform the computation of the cost function, the selection step, as well as the duplication of the parent trajectory must be modified. In addition, although the implementation of the TAMS algorithm can be straightforward for simple dynamics, its implementation for complex deterministic dynamics is tricky.

As a result, in spite of the great simplicity and generality of the TAMS algorithm, it is very difficult to write a single TAMS code that can be used for different dynamics with different level of complexity. Among other aspects, that is a serious hindrance to validation of TAMS. For instance, a code implementing the TAMS algorithm for a turbulent flow should be easily transposable to the case of an one-dimensional system, for which analytical results are available.

The `libTAMS` library provides a single framework to implement TAMS codes with any type of dynamics, whether it is stochastic, deterministic, low-dimensional or high-dimensional. In addition, it handles both instantaneous and time-averaged score functions. The different steps of the TAMS algorithm, as well as the related data, are encapsulated into objects that can be manipulated in order to write very general TAMS codes. Put differently, the `libTAMS` library is designed in such a way that, from a user point of view, the code is only (very) slightly modified depending on the properties of the dynamics under study.

E.0.1 *Object-oriented modelling of the TAMS algorithm*

The simplicity and and general formulation of the TAMS algorithm makes it very well suite for an *object-oriented* implementation. Object-oriented programming is a programming paradigm which relies on the definition of *objects* that interact with one another [113, 115]. Objects possess both *attributes*, representing data, and *methods* which describe operations on this data.

Recall from chapter 6 that an iteration j of the TAMS algorithm consists of the following steps:

1. Selection of the l_j resampled trajectories which maximum over the score function is $\mathcal{Q}_j^\star$.
2. For each l_j resampled trajectories:
   a) Selection of a parent trajectories among the $N_c - l_j$ remaining trajectories.
   b) Overwriting of the resampled trajectory by the parent, until the branching point where the score function goes beyond $\mathcal{Q}_j^\star$.
   c) Simulation of the dynamics from the branching point to the final time T_a.
3. Computation of the threshold $\mathcal{Q}_j^\star$

As of June 2018, the `libTAMS` consists of three different classes:

- `TAMS` This is the base class, from which others are derived (inherited). It provides a framework for implementing the TAMS algorithm for a score function defined as an instantaneous integral.
- `TAMSAVG` This class overloads several of `TAMS`'s methods to implement the TAMS algorithm with a cost function being defined as the time-integral of some dynamical observable.

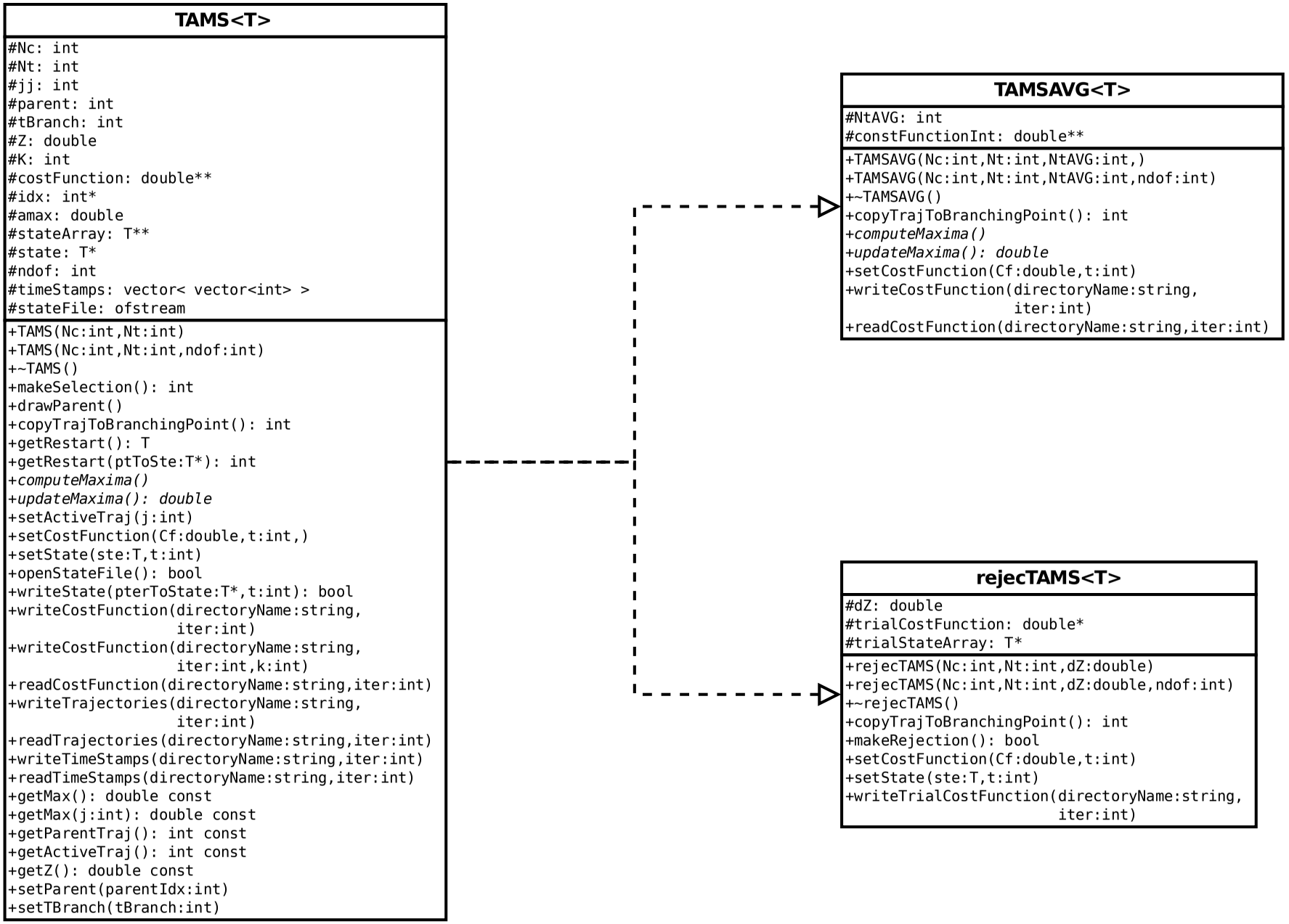


Figure E.1: Class diagram of the `libTAMS` project.

- `rejecTAMS` This class can be used to implement a modified TAMS algorithm in which resampled trajectories are branched from a shifted threshold and accepted/rejected depending on their maximum.

In the following we describe a simple example of application of the `libTAMS` library. In this example we consider the computation of the probability of rare excursions of a simple one-dimensional stochastic process. In section E.2 we discuss the application of the `libTAMS` to complex deterministic dynamics, such as turbulent flows. In this case, the main difficulty originates from the fact that the trajectories cannot be stored in memory. Then, in section E.3 we describe the application of the `libTAMS` library to problems in which the score function is defined as a time-average observable of an observable. Finally, section E.4 addresses the implementation of a modified TAMS algorithm, in which resampled trajectories are simulated from a an earlier time. As of June 2018, this modified TAMS scheme is still under study.

E.1 A SIMPLE LIBTAMS CODE: RARE EXCURSIONS OF AN ORNSTEIN–ULHENBECK

This first section introduces the core libTAMS routines on a simple example. We consider a one-dimensional, real valued variable x following an OU process defined as

$$dX_t = -\alpha X_t dt + \sqrt{2\epsilon} dW_t \tag{E.1}$$

In the following we assume that simulation of such dynamics is managed by a separate code. More precisely, we assume that the simulation of (E.1) is encapsulated in an object `myOU`. As such, this object provides methods such as `advanceOneTimeStep()`, `setInitialCondition()` or `getState()`.

Say we are interested in using the TAMS algorithm to compute the probability $q(a)$ that x goes over a threshold a over a duration T_a. Formally, this probability writes

$$q(a) = \mathcal{P}\left(\max_{0 \leq t \leq T_a} x(t) \geq a \right) \tag{E.2}$$

First things first, one must include the `libTAMS.h` header :

```
#include "libTAMS.h"
```

.

In order to use the libTAMS methods, one must first declare a libTAMS object. In our case, we instantiate the class `TAMS`. That provides

a framework for the implementation of the TAMS algorithm based on a cost function defined as an instantaneous dynamical observable.

```
TAMS<double> myTAMS(Nc, Nt);
```

The use of `TAMS<double>` declares a `TAMS` object for which the state of the dynamical system is represented by a single `double`. The parameters `Nc` and `Nt` represent the number of running trajectories and the number of timesteps they consist in, respectively.

E.1.1 *Initialisation*

The first step of the TAMS algorithm consists in simulating the `Nc` trajectories from 0 to T_a, starting from independent initial conditions. Let's initialise the N_c trajectories by drawing a random state in the stationary measure :

```
for(int j=0;j<Nc;j++)
{
  xo = randNormal(0.0,1./sqrt(2)); //Draw random initial cond.
     in stat. measure
  myOU->setInitialCondition(xo); // Set initial state to xo
  // ....
```

A trajectory is said to be *active* if it is currently being computed, whether it is during initialisation or during a resampling process. To set trajectory `j` as active, we use `setActiveTraj`.

```
myTAMS->setActiveTraj(j);
```

REMARK *Most libTAMS methods require that a trajectory is set as active.*

The next step is to compute the dynamics, in order to record the evolution of the cost function along the trajectories. In this simple case, the cost function is the dynamical variable itself. To set the value for the cost function, `setCostFunction(double Cf, int t)` must be called at each timestep :

```
for(int t=0;t<Nt;t++)
  {
    myOU->advanceOneTimestep(); // Compute state at next
    timestep
    x = myOU->getState(); // Gathers value of the cost
    function (in this case the state)
    myTAMS->setCostFunction(x, t); // Set the cost function at
     time t to be x
```

```
    //....
```

For low dimensional systems, it is possible to keep the full ensemble of running trajectories in memory. This is needed since we do not before hand which trajectory will act as parent, and which one of its `Nt` states will be the branching state. To record the states composing the trajectory, we use `setState` in a similar way we used `setCostFunction` :

```
myTAMS->setState(x,t);
```

Once the N_c trajectories have been computed, one must compute the corresponding maxima. A way to do this is to use the method `myTAMS->computeMaxima()`, that computes the maximum over each running trajectory, and updates the threshold `Z`.

```
myTAMS->computeMaxima();
```

To sum up, a full implementation for the initialisation step using the `libTAMS` library is

```
for(int j=0;j<Nc;j++)
  {
    xo = randNormal(0.0,1./sqrt(2));
    myOU->setState(xo);
    myTAMS->setActiveTraj(j);
    for(int t=0;t<Nt;t++)
      {
  myOU->advanceOneTimestep();
  x = myOU->getState();
  myTAMS->setCostFunction(x, t);
  myTAMS->setState(x, t);
      }
  }
myTAMS->computeMaxima();
```

E.1.2 *Iterations of the TAMS*

Now that the `Nc` trajectories have been computed, the cost function an states recorded, as well as the maxima, it is time to implement the core of the TAMS algorithm. In this example we are interested in computing the probability that x goes over a threshold a between 0 and T_a. The stopping criterion will therefore be that all `Nc` running trajectories go over a.

In TAMS terminology, the *threshold*, noted Z, refers to the current minimum of the `Nc` maxima over each trajectory. An iteration of the TAMS begins with the selection of the trajectories for which the maximum coincide with Z.

```
while(myTAMS->getZ()<=a)
  {
    K = myTAMS->makeSelection();
```

The simple query `getZ` returns the current threshold, that is the minimum of the set of maxima over the running trajectories. The method `makeSelection` computes Z and selects the `K` trajectories for which the maximum is equal to `Z`.

REMARK *Most of the time `K=1` but it can be that `K>1`, as mentioned in chapter 6.*

For each selected trajectories, a parent trajectory must be drawn among the $N_c - K$ non selected trajectories. This is achieved using `drawParent()`

```
for (int k=0; k<K;k++)
  {
    myTAMS->drawParent();
```

Next, the parent trajectory is copied until it reaches the threshold Z :

```
for (int k=0; k<K;k++)
  {
    myTAMS->drawParent();

    myTAMS->setActiveTraj(k);
    tBranch = myTAMS->copyTrajToBranchingPoint();
    // ...
```

The method `copyTrajToBranchingPoint()` returns in integer, representing the *branching time*. It is defined as the first time the parent's cost function is strictly above to the current threshold.

We must now simulate the dynamics from `tBranch` to `Nt`, using the state of the parent trajectory at `t=tBranch` as an initial condition. This particular state is referred to as the *branching state*. To get the branching state, we use the method `getRestartState` :

```
xo = myTAMS->getRestart(); // Get restart state into xo
myOU->setState(xo);
```

The last two steps are the computation of the dynamics and the computation of the new maximum over the resampled trajectory. This is similar to the initialisation step.

```
for (int t=tBranch;t<Nt;t++)
  {
    myOU->advanceOneTimestep();
    x = myOU->getState();
    myTAMS->setCostFunction(x, t);
    myTAMS->setState(x, t);
  }
myTAMS->updateMaxima();
```

The value of the maximum over the (resampled) trajectory `j` is updated using `updateMaxima`. It computes and sets the maximum over the **active** trajectory. It also updates the threshold `Z`.

Last but not least, let's not forget to update the current threshold Z :

```
myTAMS->updateZ();
```

A full libTAMS implementation for the iterations of the TAMS algotihm is

```
while(Z<=a)
  {
    K = myTAMS->makeSelection();
    for (int k=0; k<K;k++)
      {
  myTAMS->drawParent();
  myTAMS->setActiveTraj(k);

  tBranch = myTAMS->copyTrajToBranchingPoint();

  xo = myTAMS->getRestart();
  myOU->setState(xo);
  for (int t=tBranch+1;t<Nt;t++)
    {
      myOU->advanceOneTimestep();
      x = myOU->getState();
      myTAMS->setCostFunction(x, t);
      myTAMS->setState(x, t);
    }
  myTAMS->updateMaxima();
      }
  }
```

E.1.3 *General case*

The previous example illustrated most of the machinery provided by libTAMS. In general, to use the class `TAMS`, one should call

```
TAMS<T> myTAMS(Nc, Nt);
```

The symbol `T` denotes a given data type that represents a state of the system under study. It could be an `int`, `double`, or even a structure. For instance, the state of a system evolving in 3D space could be represented by a structure containing 3 `double` values :

```
struct 3Dpoint{
  double m_x;
  double m_y;
  double m_z;
};

TAMS<struct 3Dpoint>(Nc,Nt);
```

E.2 HIGH DIMENSIONAL DYNAMICS

For dynamics with a large number of degrees of freedom, it is often very inefficient (if even possible) to store the whole ensemble of trajectories in memory. In this case, `libTAMS` provides solutions to read/write states on disk from the running trajectories. In particular, this allows to re-simulate the system, starting from one of these saved states, up to the the branching time, so as to obtain the branching state.

The states written on disk in this way are referred to as *saved states* and the corresponding time is called a *timestamp*.

To enable this feature, the `TAMS` class must be instantiated using two additional parameters :

1. The number of degrees of freedom representing a state of the system : `ndof` In other words, a state of the system can then be represented by an array of size `ndof*sizeof(T)`.
2. The minimal duration between consecutive saved states (in units of model timesteps) : `savingStatePeriod`

```
TAMS<T>* myTAMS = new TAMS<double>(Nc, Nt, ndof,
    savingStatePeriod);
```

Along the simulation of the dynamics, states are recorded through `TAMS::<T>setCostFunction(double costFunctionValue, int time, T* pointerToState)`. The third additional argument is a pointer to the block of memory hosting the current state of the system.

REMARK *To mitigate disk usage, only new global maxima are written, always with a minimal spacing of* `savingStatePeriod` *timesteps.*

Implementation of the TAMS algorithm for high dimensional systems requires to add one additional step to the usual TAMS procedure. Having computed the branching point, one must then find the closest timestamp and re-simulate the dynamics up to the branching point, starting from the corresponding saved state. The saved states are written in a binary file. There is one of file per running trajectory. The modified procedure then reads

1. Select the K trajectories whose maximum is `Z`
2. For each selected trajectories do
   a) Draw a parent at random
   b) Copy its cost function up to tBranch
   c) Browse the recored timestamps for current active trajectory to find the nearest **before** tBranch
   d) Simulate dynamics from the corresponding saved state up to tBranch
   e) Free simulation from `tBranch` to `Nt`

E.2.1 *Writing the states on disk*

First of all, a file must be opened. It is specific to the current active trajectory :

```
myTAMS->setActiveTraj(k);
myTAMS->openStateFile();
```

If a state file was currently opened, `openStateFile` closes it first. The method `openStateFile()` opens the binary file corresponding to the currently active trajectory. If the file exists, it is overwritten.

Next, at some point during the simulation of the dynamics, it is possible to write the current state of the system on disk using `writeState`. It takes two arguments :

- A pointer to an array of size `ndof*sizeof(T)` containing the the current state of the system to be written.
- An integer representing the correspond time

Say one is using an object `Dynamics myDynamics` and that class `Dynamics` provides a method `Dynamics::getState()` to get the current state, `writeState` can be used as follows :

```
myTAMS->setActiveTraj(k);
myTAMS->openStateFile();

T* pointerToState;
for (int t=<tMin;t<tMax;t++)
  {
    //Compute dynamics and do all sort of stuff
    pointerToState = myDynamics.getState();
    myTAMS->writeState(pointerToState , t);
    //....
  }
```

E.2.2 *Getting the restart state*

After computing the branching time `tBranch`, one must simulate the dynamics starting from the nearest landmark available. This particular state is reffered to as *the restart state*. To get the restart state and time, libTAMS provides several methods :

- `int getRestartTime()` Computes and returns the restart time
- `T* getRestartState()` Returns a pointer to an array containing the restart state
- `int getRetstart(T* pointerToState)` Computes and returns the restart times, as well as copy the restart state to the location in memory pointed to by pointerToState.

To illustrate the use of these methods, let us consider to examples

E.2.2.1 *Example 1 : Using getRestart*

Consider again the case where the dynamics is implemented through an object `Dynamics myDynamics`. Suppose that `myDynamics` has its state as an attribute, and that the corresponding array is allocated in its constructor. A typical use of `getRestart` would be

```
Dynamics myDynamics;
//...

tBranch = myTAMS->copyTrajToBranchingPoint(); // Compute
    branching time
ptToState = myDynamics->getState(); // Dynamics::getState()
    returns a pointer to the array in memory
tRestart = myTAMS->getRestart(ptToState); // Then getRestart
    fills the array with the restart state and return the
    corresponding timestamp.
```

E.2.2.2 *Example 2 : Using getRestartTime in conjunction with getRestartState*

Consider again the case where the dynamics is implemented through an object `Dynamics myDynamics`. Suppose that `myDynamics` has a method `setInitialCondition(T* pointerToInitCondition)` that takes a pointer to an array of `T` as a parameter. A typical use of `getRestartTime` and `getRestartState` would then be

```
Dynamics myDynamics;
//...

tBranch = myTAMS->copyTrajToBranchingPoint(); // Compute branching time
tRestart = myTAMS->getRestartTime();

T* pointerToRestartState = myTAMS->getRestartState();
myDynamics.setInitialCondition(pointerToRestartState);
```

E.3 TAMS FOR AN INTEGRATED COST FUNCTION

The libTAMS library allows for the implementation of the TAMS algorithm, based on a cost function that is defined as a time-integral of an observable of the dynamics under study. If the instantaneous variable of interest is $x(t)$, then the integrated cost function is defined as

$$\xi(t) = \int_{t-T}^{t} x(t')dt' \quad T \leq t \leq T_a \tag{E.3}$$

Use of integrated cost functions is implemented in the `TAMSAVG` class. It is initialised as follows :

```
TAMSAVG<T> myTAMS(Nc, Nt, Nint);
```

Where `Nc` and `Nt` stand for the number of trajectories and timesteps, respectively. The third parameter is the number of timesteps over which the integration is performed. For high dimensional system, it is possible to enable the writing and reading of states from disk, by adding the parameter `ndof` as described in the previous section

```
TAMSAVG<T> myTAMS(Nc, Nt, Nint, ndof);
```

Apart from its definition, a `TAMSAVG` object is to be use in the same way as `TAMS`. For instance, the cost function is set passing the instantaneous observable and the corresponding timestep to `setCostFunction`.

```
double instantObs;
for(int t=tmin;t<tmax;t++)
  {
    //...
    myDynamics.computeOneTimeStep();
    instantObs = myDynamics.computeSomeDynamicalObservable();

    myTAMS->setCostFunction(instantObs, t);

    //...
  }
```

Times returned by `TAMSAVG` methods are defined with respect with respect to $t = 0$ and **not** $t = T$ which is the time origin of the integrated cost function.

E.4 TAMS WITH REJECTION

For some problems it can be useful to implement a modified TAMS algorithm, in which resampled trajectories result in a branching at an earlier time than the first time the parent's cost function reach Z. This is for example useful for deterministic dynamics, in which case two trajectories starting from very close initial conditions fully separate over a finite time.

Using libTAMS it is possible to implement a TAMS algorithm in which active trajectories are branched at the time when the selected parent reaches $Z-\delta Z$, instead of Z, thus branching at an earlier time t^*. The resulting resampled trajectory can then either be accepted or rejected :

- If it reaches any point above Z in between t^* and T_a, then it is *accepted*.
- Else, it is *rejected*.

Methods to implement such modified TAMS algorithm can be used using objects `rejecTAMS`. Such objects can be defined as follows :

```
rejecTAMS myTAMS(Nc, Nt, dZ);
```

where `Nt` and `Nc` are the number of running trajectories and `Nt` the number of timesteps. The parameter `dZ` is a `double` representing the threshold shift. Again, for high dimensional systems, it is always possible to enable reading/writing on disk by specifying the number of degrees of freedom of the dynamics :

```
rejecTAMS myTAMS(Nc, Nt, dZ, ndof);
```

Using libTAMS to implement a TAMS algorithm wit rejection is very similar to implementing a regular TAMS algorithm. The only to differences are :

- The parent trajectory is copied until it reaches $Z - \delta Z$
- A resampled trajectory can be rejected if it do not reach Z over its history.

Using `rejecTAMS`, the method `copyTrajToBranchingPoint` only copies the selected parent up to the point when its cost function gets **strictly** above $Z - \delta Z$. This defines the *shifted branching time*. After simulating the trajectory from `tBranch` to `Nt`, one must call the method `makeRejection()`, that decides whether the resampled trajectory is kept as part of the ensemble of running trajectories or not. If not, the selected active trajectory is not modified.

A typical code using `rejecTAMS` is very similar to one using a regular TAMS object

```
int Nt;
int Nc;
double dZ;

/* Initialize parameters
   ...
*/
/*Create rejecTAMS object*/
rejecTAMS<T> myTAMS(Nc, Nt, dZ);

/* Initialization :
   Simulate the Nc trajectory from 0 to Nt-1 and records cost
    function or state
   ...
*/

int K, tBranch;
bool isAccepted;
T x;
for (int i=0;i<Niter;i++)
  {
    K = myTAMS->makeSelection();
    for (int k=0; k<K;k++)
      {
  myTAMS->drawParent();
  myTAMS->setActiveTraj(k);

  tBranch = myTAMS->copyTrajToBranchingPoint();

  x = myTAMS.getRestart();
  myDynamics.setInitialCondition(x);
```

```
31    for(int t=tBranch;t<Nt;t++)
        {
33        myDynamics.advanceOneTimestep();
          x = myDynamics.getState();
35        myTAMS.setState(x,t);
          Cf = myDynamics.getSomeObservable();
37        myTAMS.setCostFunction(Cf,t);
        }
39    isAccepted=myTAMS.makeRejection();
          }
41    }
```

BIBLIOGRAPHY

[1] J Ahrens, Berk Geveci, and Charles Law. "ParaView: An End-User Tool for Large Data Visualization." In: *Visualization Handbook* (Jan. 2005).

[2] Rosalind J. Allen, Chantal Valeriani, and Pieter Rein ten Wolde. "Forward flux sampling for rare event simulations." en. In: *Journal of Physics: Condensed Matter* 21.46 (2009), p. 463102. ISSN: 0953-8984. DOI: 10.1088/0953-8984/21/46/463102.

[3] Mohammad Amir, Vladimir I. Nikora, and Mark T. Stewart. "Pressure forces on sediment particles in turbulent open-channel flow: a laboratory study." en. In: *Journal of Fluid Mechanics* 757 (Oct. 2014), pp. 458–497. ISSN: 0022-1120, 1469-7645. DOI: 10.1017/jfm.2014.498. (Visited on 06/04/2018).

[4] *ANSYS Fluent*. (Visited on 06/14/2018).

[5] David Aristoff, Tony Lelièvre, Christopher G. Mayne, and Ivan Teo. "Adaptive Multilevel Splitting in Molecular Dynamics Simulations." en. In: *ESAIM: Proceedings and Surveys* 48 (Jan. 2015), pp. 215–225. ISSN: 2267-3059. DOI: 10.1051/proc/201448009. (Visited on 06/06/2018).

[6] S. Asmussen and P.W. Glynn. *Stochastic simulation: algorithms and analysis*. Vol. 57. Stochastic Modelling and Applied Probability. Springer, New York, 2007, pp. xiv+476. ISBN: 978-0-387-30679-7.

[7] P. Bagchi and S. Balachandar. "Effect of turbulence on the drag and lift of a particle." In: *Physics of Fluids* 15.11 (Oct. 2003), pp. 3496–3513. ISSN: 1070-6631. DOI: 10.1063/1.1616031.

[8] Jeremie Bec and Konstantin Khanin. "Burgers Turbulence." In: *Physics Reports* 447.1-2 (Aug. 2007). arXiv: 0704.1611, pp. 1–66. ISSN: 03701573. DOI: 10.1016/j.physrep.2007.04.002.

[9] M. Berhanu et al. "Magnetic field reversals in an experimental turbulent dynamo." In: *EPL* 77.5 (2007), p. 59001. DOI: 10.1209/0295-5075/77/59001.

[10] Kurt Binder and Dieter Heermann. *Monte Carlo Simulation in Statistical Physics: An Introduction*. en. 5th ed. Graduate Texts in Physics. Berlin Heidelberg: Springer-Verlag, 2010. ISBN: 978-3-642-03162-5. (Visited on 06/04/2018).

[11] Peter G. Bolhuis, David Chandler, Christoph Dellago, and Phillip L. Geissler. "TRANSITION PATH SAMPLING: Throwing Ropes Over Rough Mountain Passes, in the Dark." In: *Annual Review of Physical Chemistry* 53.1 (Oct. 2002), pp. 291–318. ISSN: 0066-426X. DOI: 10.1146/annurev.physchem.53.082301.113146.

[12] Freddy Bouchet, Tobias Grafke, Tomás Tangarife, and Eric Vanden-Eijnden. "Large Deviations in Fast–Slow Systems." en. In: *Journal of Statistical Physics* 162.4 (Feb. 2016), pp. 793–812. ISSN: 0022-4715, 1572-9613. DOI: 10.1007/s10955-016-1449-4. (Visited on 05/30/2018).

[13] Freddy Bouchet, Jason Laurie, and Oleg Zaboronski. "Langevin Dynamics, Large Deviations and Instantons for the Quasi-Geostrophic Model and Two-Dimensional Euler Equations." en. In: *Journal of Statistical Physics* 156.6 (Sept. 2014), pp. 1066–1092. ISSN: 0022-4715, 1572-9613. DOI: 10.1007/s10955-014-1052-5. (Visited on 05/30/2018).

[14] Freddy Bouchet and Julien Reygner. "Generalisation of the Eyring–Kramers transition rate formula to irreversible diffusion processes." In: *Annales Henri Poincaré* 17.12 (2016), pp. 3499–3532. DOI: 10.1007/s00023-016-0507-4.

[15] Freddy Bouchet and Eric Simonnet. "Random Changes of Flow Topology in Two-Dimensional and Geophysical Turbulence." In: *Phys. Rev. Lett.* 102.9 (2009), p. 094504. DOI: 10.1103/PhysRevLett.102.094504.

[16] Alan J Bray, Satya N Majumdar, and Grégory Schehr. "Persistence and first-passage properties in nonequilibrium systems." In: *Adv. Phys.* 62.3 (2013), pp. 225–361. DOI: 10.1103/PhysRevE.87.022118.

[17] Charles-Edouard Bréhier, Maxime Gazeau, Ludovic Goudenège, Tony Lelièvre, and Mathias Rousset. "Unbiasedness of some generalized adaptive multilevel splitting algorithms." In: *Ann. Appl. Probab.* 26.6 (2016), pp. 3559–3601. ISSN: 1050-5164. DOI: 10.1214/16-AAP1185.

[18] Charles-Edouard Bréhier, Ludovic Goudenège, and Loïc Tudela. "Central limit theorem for adaptive multilevel splitting estimators in an idealized setting." In: *Monte Carlo and quasi-Monte Carlo methods*. Vol. 163. Springer Proc. Math. Stat. Springer, [Cham], 2016, pp. 245–260. DOI: 10.1007/978-3-319-33507-0_10.

[19] Charles-Edouard Bréhier, Tony Lelièvre, and Mathias Rousset. "Analysis of adaptive multilevel splitting algorithms in an idealized case." In: *ESAIM Probab. Stat.* 19 (2015), pp. 361–394. ISSN: 1292-8100. DOI: 10.1051/ps/2014029.

[20] M. Breuer, J. Bernsdorf, T. Zeiser, and F. Durst. "Accurate computations of the laminar flow past a square cylinder based on two different methods: lattice-Boltzmann and finite-volume." In: *International Journal of Heat and Fluid Flow* 21.2 (Apr. 2000), pp. 186–196. ISSN: 0142-727X. DOI: 10.1016/S0142-727X(99)00081-8.

[21] Tobias Brewer, Stephen R. Clark, Russell Bradford, and Robert L. Jack. "Efficient characterisation of large deviations using population dynamics." en. In: *Journal of Statistical Mechanics: Theory and Experiment* 2018.5 (2018), p. 053204. ISSN: 1742-5468. DOI: 10.1088/1742-5468/aab3ef.

[22] James Bucklew. *Introduction to Rare Event Simulation*. en. Springer Series in Statistics. New York: Springer-Verlag, 2004. ISBN: 978-0-387-20078-1. (Visited on 06/11/2018).

[23] Olivier Cadot, A. Courbois, D. Ricot, Tony Ruiz, F. Harambat, V. Herbert, R. Vigneron, and J. Délery. "Characterizations of force and pressure fluctuations on real vehicles." en. In: *International Journal of Engineering Systems Modelling and Simulation* 8.2 (Feb. 2016), pp. 99–105. (Visited on 06/04/2018).

[24] Alfonso Caiazzo. "Analysis of Lattice Boltzmann Initialization Routines." en. In: *Journal of Statistical Physics* 121.1-2 (Oct. 2005), pp. 37–48. ISSN: 0022-4715, 1572-9613. DOI: 10.1007/s10955-005-7010-5. (Visited on 06/11/2018).

[25] Daniel F. Calef and J. M. Deutch. "Diffusion-Controlled Reactions." In: *Ann. Rev. Phys. Chem.* 34 (1983), pp. 493–524. DOI: 10.1146/annurev.pc.34.100183.002425.

[26] Julien Cattiaux, Robert Vautard, Christophe Cassou, Pascal Yiou, Valérie Masson-Delmotte, and Francis Codron. "Winter 2010 in Europe: A cold extreme in a warming climate." In: *Geophys. Res. Lett.* 37 (2010), p. L20704.

[27] Florent Cérou and Arnaud Guyader. "Adaptive multilevel splitting for rare event analysis." In: *Stoch. Anal. Appl.* 25 (2007), pp. 417–443. DOI: 10.1080/07362990601139628.

[28] Frederic Cerou, Bernard Delyon, Arnaud Guyader, and Mathias Rousset. "A Central Limit Theorem for Fleming-Viot Particle Systems with Soft Killing." arXiv:1611.00515. 2016.

[29] Frédéric Cérou, Arnaud Guyader, Tony Lelièvre, and David Pommier. "A multiple replica approach to simulate reactive trajectories." In: *J. Chem. Phys.* 134.5 (2011), p. 054108. DOI: 10.1063/1.3518708.

[30] Sydney Chapman. *The Mathematical Theory of Non-uniform Gases: An Account of the Kinetic Theory of Viscosity, Thermal Conduction and Diffusion in Gases*. Anglais. 3rd ed. Cambridge ; New York: Cambridge University Press, 1991. ISBN: 978-0-521-40844-8.

[31] Hudong Chen, Shiyi Chen, and William H. Matthaeus. "Recovery of the Navier-Stokes equations using a lattice-gas Boltzmann method." In: *Physical Review A* 45.8 (Apr. 1992), R5339–R5342. DOI: 10.1103/PhysRevA.45.R5339.

[32] A. I. Chernykh and M. G. Stepanov. "Large negative velocity gradients in Burgers turbulence." In: *Physical Review E* 64.2 (July 2001), p. 026306. DOI: 10.1103/PhysRevE.64.026306.

[33] Alvaro Corral. "Time-decreasing hazard and increasing time until the next earhtquake." In: *Phys. Rev. E* 71 (2005), p. 017101. DOI: 10.1103/PhysRevE.71.017101.

[34] Daan Frenkel, Berend Smit. *Understanding Molecular Simulation*. en. Elsevier, 2002. ISBN: 978-0-12-267351-1. DOI: 10.1016/B978-0-12-267351-1.X5000-7.

[35] Frank Den Hollander. *Large Deviations*. Vol. 14. American Mathematical Society, 2008.

[36] M. Denny. "Extreme Drag Forces and the Survival of Wind- and Water-Swept Organisms." en. In: *Journal of Experimental Biology* 194.1 (Sept. 1994), pp. 97–115. ISSN: 0022-0949, 1477-9145. (Visited on 06/04/2018).

[37] Arnaud Doucet, Nando De Freitas, and Neil Gordon. *Sequential Monte Carlo methods in practice*. English. New York: Springer, 2001. ISBN: 0-387-95146-6 978-0-387-95146-1 1-4419-2887-1 978-1-4419-2887-0.

[38] Mark Dykman. "Periodically modulated quantum nonlinear oscillators." In: *Fluctuating Nonlinear Oscillators: From Nanomechanics to Quantum Superconducting Circuits*. Ed. by Mark Dykman. Oxford University Press, 2012. ISBN: 9780199691388. DOI: 10.1093/acprof:oso/9780199691388.001.0001.

[39] Kristian Dysthe, Harald E Krogstad, and Peter Muller. "Oceanic Rogue Waves." In: *Ann. Rev. Fluid Mech.* 40.1 (2008), pp. 287–310. DOI: 10.1146/annurev.fluid.40.111406.102203.

[40] W. E, W. Ren, and Eric Vanden-Eijnden. "Minimum action method for the study of rare events." In: *Comm. Pure Appl. Math.* 52 (2004), pp. 637–656. DOI: 10.1002/cpa.20005.

[41] Weinan E, Weiqing Ren, and Eric Vanden-Eijnden. "String method for the study of rare events." In: *Phys. Rev. B* 66.5 (2002), p. 052301. DOI: 10.1103/PhysRevB.66.052301.

[42] Richard S. Ellis. *Entropy, Large Deviations, and Statistical Mechanics*. Springer, New-York, 1985.

[43] Paul Embrechts, Claudia Klüppelberg, and Thomas Mikosch. *Modelling Extremal Events: for Insurance and Finance*. Vol. 33. Stochastic Modelling and Applied Probability. Springer, 2013. ISBN: 9783540609315.

[44] Titus S. van Erp and Peter G. Bolhuis. "Elaborating transition interface sampling methods." In: *Journal of Computational Physics* 205.1 (May 2005), pp. 157–181. ISSN: 0021-9991. DOI: 10.1016/j.jcp.2004.11.003.

[45] Fernando A. Escobedo, Ernesto E. Borrero, and Juan C. Araque. "Transition path sampling and forward flux sampling. Applications to biological systems." en. In: *Journal of Physics: Condensed Matter* 21.33 (2009), p. 333101. ISSN: 0953-8984. DOI: 10.1088/0953-8984/21/33/333101.

[46] Joel H. Ferziger and Milovan Peric. *Computational Methods for Fluid Dynamics*. en. 3rd ed. Berlin Heidelberg: Springer-Verlag, 2002. ISBN: 978-3-540-42074-3. (Visited on 06/04/2018).

[47] L. Fiabane, M. Gohlke, and Olivier Cadot. "Characterization of flow contributions to drag and lift of a circular cylinder using a volume expression of the fluid force." en. In: *European Journal of Mechanics - B/Fluids* 30.3 (2010), pp. 311–315. DOI: 10.1016/j.euromechflu.2010.12.001. (Visited on 06/13/2018).

[48] Jean-Yves Fortin and Maxime Clusel. "Applications of extreme value statistics in physics." In: *J. Phys. A* 48.18 (2015), pp. 1–35. DOI: 10.1088/1751-8113/48/18/183001.

[49] M. I. Freidlin and A. D. Wentzell. *Random Perturbations of Dynamical Systems*. 2nd edition. Springer, New-York, 1998.

[50] U. Frisch, B. Hasslacher, and Y. Pomeau. "Lattice-Gas Automata for the Navier-Stokes Equation." In: *Physical Review Letters* 56.14 (Apr. 1986), pp. 1505–1508. DOI: 10.1103/PhysRevLett.56.1505.

[51] Uriel Frisch. *Turbulence: The Legacy of A. N. Kolmogorov*. Anglais. Cambridge, Eng. ; New York: Cambridge University Press, Jan. 2010. ISBN: 978-0-521-45713-2.

[52] Edgar Gabriel et al. "Open MPI: Goals, Concept, and Design of a Next Generation MPI Implementation." In: *Proceedings, 11th European PVM/MPI Users' Group Meeting*. Budapest, Hungary, Sept. 2004, pp. 97–104.

[53] C W Gardiner. *Handbook of Stochastic Methods for physics, chemistry, and the natural sciences*. 4th edition. Springer, Berlin, 2009.

[54] J. P. Garrahan, R. L. Jack, V. Lecomte, E. Pitard, K. van Duijvendijk, and F. van Wijland. "First-order dynamical phase transition in models of glasses: an approach based on ensembles of histories." In: *Journal of Physics A: Mathematical and Theoretical* 42.7 (Feb. 2009). arXiv: 0810.5298, p. 075007. ISSN: 1751-8113, 1751-8121. DOI: 10.1088/1751-8113/42/7/075007.

[55] Michael Ghil et al. "Extreme events: dynamics, statistics and prediction." In: *Nonlin. Proc. Geophys.* 18.3 (2011), pp. 295–350.

[56] C Giardina, J Kurchan, V Lecomte, and J Tailleur. "Simulating Rare Events in Dynamical Processes." In: *J. Stat. Phys.* 145 (2011), pp. 787–811. DOI: 10.1007/s10955-011-0350-4.

[57] Cristian Giardina, Jorge Kurchan, Vivien Lecomte, and Julien Tailleur. "Simulating Rare Events in Dynamical Processes." en. In: *Journal of Statistical Physics* 145.4 (Nov. 2011), pp. 787–811. ISSN: 0022-4715, 1572-9613. DOI: 10.1007/s10955-011-0350-4. (Visited on 05/31/2018).

[58] Cristian Giardina, Jorge Kurchan, and Luca Peliti. "Direct Evaluation of Large-Deviation Functions." English. In: *Phys. Rev. Lett.* 96.12 (2006), p. 120603. DOI: 10.1103/PhysRevLett.96.120603.

[59] Cristian Giardinà, Jorge Kurchan, and Luca Peliti. "Direct Evaluation of Large-Deviation Functions." In: *Physical Review Letters* 96.12 (Mar. 2006), p. 120603. DOI: 10.1103/PhysRevLett.96.120603.

[60] Paul Glasserman, Philip Heidelberger, Perwez Shahabuddin, and Tim Zajic. "A Look At Multilevel Splitting." en. In: *Monte Carlo and Quasi-Monte Carlo Methods 1996*. Lecture Notes in Statistics. Springer, New York, NY, 1998, pp. 98–108. ISBN: 978-0-387-98335-6 978-1-4612-1690-2. DOI: 10.1007/978-1-4612-1690-2_5. (Visited on 06/13/2018).

[61] Paul Glasserman, Philip Heidelberger, Perwez Shahabuddin, and Tim Zajic. "Multilevel Splitting for Estimating Rare Event Probabilities." In: *Operations Research* 47.4 (Aug. 1999), pp. 585–600. ISSN: 0030-364X. DOI: 10.1287/opre.47.4.585.

[62] Claude Godrèche, Satya N Majumdar, and Grégory Schehr. "Record statistics of a strongly correlated time series: random walks and Lévy flights." In: *Journal of Physics A: Mathematical and Theoretical* 50.33 (2017), p. 333001. DOI: 10.1088/1751-8121/aa71c1.

[63] Nigel Goldenfeld, Nicholas Guttenberg, and Gustavo Gioia. "Extreme fluctuations and the finite lifetime of the turbulent state." In: *Physical review. E, Statistical, nonlinear, and soft matter physics* 81 (Mar. 2010), p. 035304. DOI: 10.1103/PhysRevE.81.035304.

[64] T. Grafke, R. Grauer, T. Schäfer, and E. Vanden-Eijnden. "Relevance of instantons in Burgers turbulence." en. In: *EPL (Europhysics Letters)* 109.3 (2015), p. 34003. ISSN: 0295-5075. DOI: 10.1209/0295-5075/109/34003.

[65] Tobias Grafke, Rainer Grauer, and Tobias Schäfer. "Instanton filtering for the stochastic Burgers equation." en. In: *Journal of Physics A: Mathematical and Theoretical* 46.6 (2013), p. 062002. ISSN: 1751-8121. DOI: 10.1088/1751-8113/46/6/062002.

[66] Tobias Grafke, Rainer Grauer, and Tobias Schäfer. "The instanton method and its numerical implementation in fluid mechanics." English. In: *J. Phys. A* 48.33 (2015), pp. 1–39. DOI: 10.1088/1751-8113/48/33/333001.

[67] M. Grandemange, O. Cadot, A. Courbois, V. Herbert, D. Ricot, T. Ruiz, and R. Vigneron. "A study of wake effects on the drag of Ahmed's squareback model at the industrial scale." In: *Journal of Wind Engineering and Industrial Aerodynamics* 145 (Oct. 2015), pp. 282–291. ISSN: 0167-6105. DOI: 10.1016/j.jweia.2015.03.004.

[68] Peter Grassberger and Walter Nadler. "'Go with the Winners' Simulations." en. In: *Computational Statistical Physics*. Springer, Berlin, Heidelberg, 2002, pp. 169–190. ISBN: 978-3-642-07571-1 978-3-662-04804-7. DOI: 10.1007/978-3-662-04804-7_11. (Visited on 06/11/2018).

[69] L. S. Grigorio, F. Bouchet, R. M. Pereira, and L. Chevillard. "Instantons in a Lagrangian model of turbulence." en. In: *Journal of Physics A: Mathematical and Theoretical* 50.5 (2017), p. 055501. ISSN: 1751-8121. DOI: 10.1088/1751-8121/aa51a3.

[70] E J Gumbel. "The return period of flood flows." English. In: *Annals of Mathematical Statistics* 12 (1941), pp. 163–190.

[71] Victor Gurarie and Alexander Migdal. "Instantons in the Burgers equation." In: *Physical Review E* 54.5 (Nov. 1996), pp. 4908–4914. DOI: 10.1103/PhysRevE.54.4908.

[72] *Gwen Jorgensen*. http://www.gwenjorgensen.com. [Online; accessed 28-August-2018].

[73] H. Kahn and T. E. Harris. "Estimation of Particle Transmission by Random Sampling." In: *National Bureau of Standards*. Applied Mathematics Series 12 (1951), pp. 27–30.

[74] Ásta Hannesdóttir, Mark Kelly, and Nikolay Dimitrov. "Extreme fluctuations of wind speed for a coastal/offshore climate: statistics and impact on wind turbine loads." English. In: *Wind Energy Science Discussions* (Feb. 2018), pp. 1–21. ISSN: 2366-7443. DOI: https://doi.org/10.5194/wes-2018-12. (Visited on 06/04/2018).

[75] J. Hardy, O. de Pazzis, and Y. Pomeau. "Molecular dynamics of a classical lattice gas: Transport properties and time correlation functions." In: *Physical Review A* 13.5 (May 1976), pp. 1949–1961. DOI: 10.1103/PhysRevA.13.1949.

[76] J. Hardy, Y. Pomeau, and O. de Pazzis. "Time Evolution of a Two-Dimensional Classical Lattice System." In: *Physical Review Letters* 31.5 (July 1973), pp. 276–279. DOI: 10.1103/PhysRevLett.31.276.

[77] Alexander T. Hawk. "Milestoning with coarse memory." In: *The Journal of Chemical Physics* 138.15 (Apr. 2013), p. 154105. ISSN: 0021-9606. DOI: 10.1063/1.4795838.

[78] Esteban Guevara Hidalgo, Takahiro Nemoto, and Vivien Lecomte. "Finite-Time and -Size Scalings in the Evaluation of Large Deviation Functions: Numerical Approach in Continuous Time." In: *Physical Review E* 95.6 (June 2017). arXiv: 1607.08804. ISSN: 2470-0045, 2470-0053. DOI: 10.1103/PhysRevE.95.062134.

[79] *High-fidelity lattice Boltzmann CFD simulations*. en-US. URL: http://www.prolb-cfd.com/ (visited on 06/14/2018).

[80] F. J. Higuera and J. Jiménez. "Boltzmann Approach to Lattice Gas Simulations." en. In: *EPL (Europhysics Letters)* 9.7 (1989), p. 663. ISSN: 0295-5075. DOI: 10.1209/0295-5075/9/7/009.

[81] F. J. Higuera, S. Succi, and R. Benzi. "Lattice Gas Dynamics with Enhanced Collisions." en. In: *EPL (Europhysics Letters)* 9.4 (1989), p. 345. ISSN: 0295-5075. DOI: 10.1209/0295-5075/9/4/008.

[82] Holger Homann, Jérémie Bec, and Rainer Grauer. "Effect of turbulent fluctuations on the drag and lift forces on a towed sphere and its boundary layer." en. In: *Journal of Fluid Mechanics* 721 (Apr. 2013), pp. 155–179. ISSN: 0022-1120, 1469-7645. DOI: 10.1017/jfm.2013.66. (Visited on 06/04/2018).

[83] Juntao Huang, Hao Wu, and Wen-An Yong. "On Initial Conditions for the Lattice Boltzmann Method." en. In: *Communications in Computational Physics* 18.2 (Aug. 2015), pp. 450–468. ISSN: 1815-2406, 1991-7120. DOI: 10.4208/cicp.040913.220115a. (Visited on 06/11/2018).

[84] Salvador Izquierdo, Paula Martínez-Lera, and Norberto Fueyo. "Analysis of open boundary effects in unsteady lattice Boltzmann simulations." In: *Computers & Mathematics with Applications*. Mesoscopic Methods in Engineering and Science 58.5 (Sept. 2009), pp. 914–921. ISSN: 0898-1221. DOI: 10.1016/j.camwa.2009.02.014.

[85] Anurag Jain, Mukund Srinivasan, and Gary C. Hart. "Performance based design extreme wind loads on a tall building." en. In: *The Structural Design of Tall Buildings* 10.1 (), pp. 9–26. ISSN: 1099-1794. DOI: 10.1002/tal.165.

[86] Hrvoje Jasak. "OpenFOAM: Open source CFD in research and industry." In: *International Journal of Naval Architecture and Ocean Engineering* 1.2 (Dec. 2009), pp. 89–94. ISSN: 2092-6782. DOI: 10.2478/IJNAOE-2013-0011.

[87] Christopher Jung, Dirk Schindler, Alexander Buchholz, and Jessica Laible. "Global Gust Climate Evaluation and Its Influence on Wind Turbines." en. In: *Energies* 10.10 (Sept. 2017), p. 1474. DOI: 10.3390/en10101474. (Visited on 06/04/2018).

[88] Peter C. Kalverla, Gert-Jan Steeneveld, Reinder J. Ronda, and Albert A. M. Holtslag. "An observational climatology of anomalous wind events at offshore meteomast IJmuiden (North Sea)." In: *Journal of Wind Engineering and Industrial Aerodynamics* 165 (June 2017), pp. 86–99. ISSN: 0167-6105. DOI: 10.1016/j.jweia.2017.03.008.

[89] Ioan Kosztin, Byron Faber, and Klaus Schulten. "Introduction to the Diffusion Monte Carlo Method." In: *American Journal of Physics* 64.5 (May 1996). arXiv: physics/9702023, pp. 633–644. ISSN: 0002-9505, 1943-2909. DOI: 10.1119/1.18168.

[90] H A Kramers. "Brownian motion in a field of force and the diffusion model of chemical reactions." In: *Physica* 7 (1940), pp. 284–304. DOI: 10.1016/S0031-8914(40)90098-2.

[91] I. Krönke and H. Sockel. "Measurement of extreme drag coefficients of building models." In: *Journal of Wind Engineering and Industrial Aerodynamics*. Special Issue 6th Colloquium on Industrial Aerodynamics Building Aerodynamics 23 (Jan. 1986), pp. 149–163. ISSN: 0167-6105. DOI: 10.1016/0167-6105(86)90039-5.

[92] Timm Krüger, Halim Kusumaatmaja, Alexandr Kuzmin, Orest Shardt, Goncalo Silva, and Erlend Magnus Viggen. *The Lattice Boltzmann Method: Principles and Practice*. en. Graduate Texts in Physics. Springer International Publishing, 2017. ISBN: 978-3-319-44647-9. (Visited on 06/10/2018).

[93] Juhani Kurkijärvi. "Intrinsic Fluctuations in a Superconducting Ring Closed with a Josephson Junction." English. In: *Phys. Rev. B* 6.3 (1972), p. 832. DOI: 10.1103/PhysRevB.6.832.

[94] Tanguy Laffargue. "Grandes déviations d'exposants de Lyapunov dans les systèmes étendus." fr. PhD thesis. Université Paris Diderot (Paris 7), Jan. 2015. (Visited on 06/12/2018).

[95] Agnes Lagnoux. "Rare event simulation." In: *Probability in the Engineering and Informational Sciences* 20.1 (2006), pp. 45–66. DOI: 10.1017/S0269964806060025.

[96] David P. Landau and Kurt Binder. *A guide to Monte Carlo simulations in statistical physics*. Fourth. Cambridge University Press, Cambridge, 2015, pp. xvii+519. ISBN: 978-1-107-07402-6.

[97] Lev Davidovich Landau and Evgeny Mikhailovich Lifshitz. "Fluid Mechanics, Volume 6 of course of theoretical physics." In: *Course of theoretical physics/by LD Landau and EM Lifshitz* 6 (1987).

[98] J S Langer. "Statistical theory of the decay of metastable states." In: *Annals of Physics* 54 (1969), pp. 258–275.

[99] Jonas Latt and Bastien Chopard. "Lattice Boltzmann Method with regularized non-equilibrium distribution functions." In: *arXiv:physics/0506157* (June 2005). arXiv: physics/0506157.

[100] Jonas Latt, Bastien Chopard, Orestis Malaspinas, Michel Deville, and Andreas Michler. "Straight velocity boundaries in the lattice Boltzmann method." In: *Physical Review E* 77.5 (May 2008), p. 056703. DOI: 10.1103/PhysRevE.77.056703.

[101] Jason Laurie and Freddy Bouchet. "Computation of rare transitions in the barotropic quasi-geostrophic equations." en. In: *New Journal of Physics* 17.1 (2015), p. 015009. ISSN: 1367-2630. DOI: 10.1088/1367-2630/17/1/015009.

[102] M. R. Leadbetter. *Extremes and related properties of random sequences and processes*. Springer series in statistics. New York: Springer-Verlag, 1983. ISBN: 0-387-90731-9.

[103] Vivien Lecomte and Julien Tailleur. "A numerical approach to large deviations in continuous time." en. In: *Journal of Statistical Mechanics: Theory and Experiment* 2007.03 (2007), P03004. ISSN: 1742-5468. DOI: 10.1088/1742-5468/2007/03/P03004.

[104] Peter Lenaers, Qiang Li, Geert Brethouwer, Philipp Schlatter, and Ramis Örlü. "Rare backflow and extreme wall-normal velocity fluctuations in near-wall turbulence." In: *Physics of Fluids* 24.3 (Mar. 2012), p. 035110. ISSN: 1070-6631. DOI: 10.1063/1.3696304.

[105] Thibault Lestang, Francesco Ragone, Charles-Edouard Bréhier, Corentin Herbert, and Freddy Bouchet. "Computing return times or return periods with rare event algorithms." en. In: *Journal of Statistical Mechanics: Theory and Experiment* 2018.4 (2018), p. 043213. ISSN: 1742-5468. DOI: 10.1088/1742-5468/aab856.

[106] Jun S. Liu. *Monte Carlo strategies in scientific computing*. Springer Series in Statistics. Springer, New York, 2008, pp. xvi+343. ISBN: 978-0-387-76369-9; 0-387-95230-6.

[107] Valerio Lucarini, Ana Cristina Moreira Freitas, Davide Faranda, Jorge Milhazes Freitas, Mark Holland, Tobias Kuna, Matthew Nicol, Mike Todd, and Sandro Vaienti. *Extremes and Recurrence in Dynamical Systems*. English. New-York: Wiley, 2016. ISBN: 1118632192.

[108] Guy R. McNamara and Gianluigi Zanetti. "Use of the Boltzmann Equation to Simulate Lattice-Gas Automata." In: *Physical Review Letters* 61.20 (Nov. 1988), pp. 2332–2335. DOI: 10.1103/PhysRevLett.61.2332.

[109] Gerald A Meehl and C Tebaldi. "More intense, more frequent, and longer lasting heat waves in the 21st century." English. In: *Science* 305 (2004), pp. 994–997.

[110] Renwei Mei, Li-Shi Luo, Pierre Lallemand, and Dominique d'Humières. "Consistent initial conditions for lattice Boltzmann simulations." In: *Computers & Fluids*. Proceedings of the First International Conference for Mesoscopic Methods in Engineering and Science 35.8 (Sept. 2006), pp. 855–862. ISSN: 0045-7930. DOI: 10.1016/j.compfluid.2005.08.008.

[111] D. Mesterházy and K. Jansen. "Anomalous scaling in the random-force-driven Burgers' equation: a Monte Carlo study." en. In: *New Journal of Physics* 13.10 (2011), p. 103028. ISSN: 1367-2630. DOI: 10.1088/1367-2630/13/10/103028.

[112] David Mesterházy, Luca Biferale, Karl Jansen, and Raffaele Tripiccione. "Lattice Monte Carlo methods for systems far from equilibrium." In: *arXiv:1311.4386 [cond-mat, physics:hep-lat, physics:nlin]* (Nov. 2013). arXiv: 1311.4386.

[113] Bertrand Meyer. *Object-Oriented Software Construction*. English. 2 edition. Upper Saddle River, N.J: Prentice Hall, Apr. 1997. ISBN: 978-0-13-629155-8.

[114] Patrick Milan, Matthias Wächter, and Joachim Peinke. "Turbulent Character of Wind Energy." In: *Physical Review Letters* 110.13 (Mar. 2013), p. 138701. DOI: 10.1103/PhysRevLett.110.138701.

[115] John C. Mitchell. *Concepts in Programming Languages*. English. 1 edition. Cambridge, UK ; New York: Cambridge University Press, Oct. 2002. ISBN: 978-0-521-78098-8.

[116] Parviz Moin. "Advances in large eddy simulation methodology for complex flows." In: *Int. J. Heat Fluid Flow* 23 (2002), pp. 710–720.

[117] Pierre Del Moral. *Feynman-Kac Formulae: Genealogical and Interacting Particle Systems with Applications*. en. Probability and Its Applications. New York: Springer-Verlag, 2004. ISBN: 978-0-387-20268-6. (Visited on 06/11/2018).

[118] Pierre Del Moral and Josselin Garnier. "Genealogical particle analysis of rare events." en. In: *The Annals of Applied Probability* 15.4 (Nov. 2005), pp. 2496–2534. ISSN: 1050-5164, 2168-8737. DOI: 10.1214/105051605000000566. (Visited on 06/11/2018).

[119] Takahiro Nemoto and Alexandros Alexakis. "Method to measure efficiently rare fluctuations of turbulence intensity for turbulent-laminar transitions in pipe flows." In: *Physical Review E* 97.2 (Feb. 2018). arXiv: 1707.04819. ISSN: 2470-0045, 2470-0053. DOI: 10.1103/PhysRevE.97.022207.

[120] Takahiro Nemoto, Freddy Bouchet, Robert L. Jack, and Vivien Lecomte. "Population-dynamics method with a multicanonical feedback control." In: *Physical Review E* 93.6 (June 2016), p. 062123. DOI: 10.1103/PhysRevE.93.062123.

[121] Takahiro Nemoto, Esteban Guevara Hidalgo, and Vivien Lecomte. "Finite-time and finite-size scalings in the evaluation of large-deviation functions: Analytical study using a birth-death process." In: *Physical Review E* 95.1 (Jan. 2017), p. 012102. DOI: 10.1103/PhysRevE.95.012102.

[122] M. E. J. Newman and G. T. Barkema. *Monte Carlo Methods in Statistical Physics*. Anglais. Oxford : New York: Clarendon Press, 1999. ISBN: 978-0-19-851797-9.

[123] C. Nicolis and S.C. Nicolis. "Return time statistics of extreme events in deterministic dynamical systems." In: *EPL* 80 (2007), p. 40003.

[124] F Noé, C Schütte, and E Vanden-Eijnden. "Constructing the equilibrium ensemble of folding pathways from short off-equilibrium simulations." In: *Proc. Natl. Acad. Sci. U.S.A.* 106 (2009), pp. 19011–19016.

[125] F.E.L. Otto, N. Massey, G.J. van Oldenborgh, R.G. Jones, and M.R. Allen. "Reconciling two approaches to attribution of the 2010 Russian heat wave." In: *Geophys. Res. Lett.* 39 (2012), p. L04702.

[126] Didier Paillard. "The timing of Pleistocene glaciations from a simple multiple-state climate model." In: *Nature* 391 (1998), pp. 378–381. DOI: 10.1038/34891.

[127] D J Peres and A Cancelliere. "Estimating return period of landslide triggering by Monte Carlo simulation." English. In: *Journal of Hydrology* 541 (2016), pp. 256–271. DOI: 10.1016/j.jhydrol.2016.03.036.

[128] *Pôle Scientifique de Modélisation Numérique*. http://www.ens-lyon.fr/PSMN/doku.php. [Online; accessed 11-June-2018].

[129] Y. H. Qian, D. D'Humières, and P. Lallemand. "Lattice BGK Models for Navier-Stokes Equation." en. In: *EPL (Europhysics Letters)* 17.6 (1992), p. 479. ISSN: 0295-5075. DOI: 10.1209/0295-5075/17/6/001.

[130] Francesco Ragone, Jeroen Wouters, and Freddy Bouchet. "Computation of extreme heat waves in climate models using a large deviation algorithm." en. In: *Proceedings of the National Academy of Sciences* 115.1 (Jan. 2018), pp. 24–29. ISSN: 0027-8424, 1091-6490. DOI: 10.1073/pnas.1712645115. (Visited on 05/30/2018).

[131] Stefan Rahmstorf and Dim Coumou. "Increase of extreme events in a warming world." English. In: *Proc. Natl. Acad. Sci. U.S.A.* 108.44 (2011), pp. 17905–17909. DOI: 10.1073/pnas.1101766108.

[132] Ushnish Ray, Garnet Kin-Lic Chan, and David T. Limmer. "Importance sampling large deviations in nonequilibrium steady states. I." In: *The Journal of Chemical Physics* 148.12 (Mar. 2018). arXiv: 1708.00459, p. 124120. ISSN: 0021-9606, 1089-7690. DOI: 10.1063/1.5003151.

[133] Sidney Redner. *A Guide to First-Passage Processes*. Cambridge University Press, 2001. DOI: 10.1017/CBO9780511606014.

[134] H Risken. *The Fokker-Planck Equation*. 2nd edition. Springer, 1989.

[135] J.-P. Rivet and J. P. Boon. *Lattice Gas Hydrodynamics*. English. Cambridge; New York: Cambridge University Press, Sept. 2005. ISBN: 978-0-521-01971-2.

[136] Jean-Marie Robine, Siu Lan K Cheung, Sophie Le Roy, Herman Van Oyen, Clare Griffiths, Jean-Pierre Michel, and François Richard Herrmann. "Death toll exceeded 70,000 in Europe during the summer of 2003." In: *Comptes Rendus Biologies* 331.2 (2008), pp. 171–178.

[137] Christian M. Rohwer, Florian Angeletti, and Hugo Touchette. "Convergence of large deviation estimators." In: *Physical Review E* 92.5 (Nov. 2015). arXiv: 1409.8531. ISSN: 1539-3755, 1550-2376. DOI: 10.1103/PhysRevE.92.052104.

[138] Joran Rolland. "Extremely rare collapse and build-up of turbulence in stochastic models of transitional wall flows." In: *Physical Review E* 97.2 (Feb. 2018), p. 023109. DOI: 10.1103/PhysRevE.97.023109.

[139] Joran Rolland, Freddy Bouchet, and Eric Simonnet. "Computing Transition Rates for the 1-D Stochastic Ginzburg–Landau–Allen–Cahn Equation for Finite-Amplitude Noise with a Rare Event Algorithm." en. In: *Journal of Statistical Physics* 162.2 (Jan. 2016), pp. 277–311. ISSN: 0022-4715, 1572-9613. DOI: 10.1007/s10955-015-1417-4. (Visited on 05/30/2018).

[140] Joran Rolland and Eric Simonnet. "Statistical behavior of adaptive multilevel splitting algorithms in simple models." In: *Journal of Computational Physics* 283 (Feb. 2015). arXiv: 1412.3362, pp. 541–558. ISSN: 00219991. DOI: 10.1016/j.jcp.2014.12.009.

[141] Joran Rolland and Eric Simonnet. "Statistical behaviour of adaptive multilevel splitting algorithms in simple models." English. In: *J. Comput. Phys.* 283 (2015), pp. 541–558. DOI: 10.1016/j.jcp.2014.12.009.

[142] A. Roshko. "Perspectives on bluff body aerodynamics." In: *Journal of Wind Engineering and Industrial Aerodynamics* 49.1 (Dec. 1993), pp. 79–100. ISSN: 0167-6105. DOI: 10.1016/0167-6105(93)90007-B.

[143] Alessandro De Rosis. "Non-orthogonal central moments relaxing to a discrete equilibrium: A D2Q9 lattice Boltzmann model." en. In: *EPL (Europhysics Letters)* 116.4 (2016), p. 44003. ISSN: 0295-5075. DOI: 10.1209/0295-5075/116/44003.

[144] Alessandro De Rosis. "Central-moments–based lattice Boltzmann schemes with force-enriched equilibria." en. In: *EPL (Europhysics Letters)* 117.3 (2017), p. 34003. ISSN: 0295-5075. DOI: 10.1209/0295-5075/117/34003.

[145] G. Rubino and B. Tuffin. "Introduction to rare event simulation." In: *Rare event simulation using Monte Carlo methods*. Wiley, Chichester, 2009, pp. 1–13. DOI: 10.1002/9780470745403.ch1.

[146] P. Sagaut. *Large Eddy Simulation for Incompressible Flows: An Introduction*. en. 3rd ed. Scientific Computation. Berlin Heidelberg: Springer-Verlag, 2006. ISBN: 978-3-540-26344-9. (Visited on 06/13/2018).

[147] Ewe-Wei Saw, Gregory P. Bewley, Eberhard Bodenschatz, Samriddhi Sankar Ray, and Jérémie Bec. "Extreme fluctuations of the relative velocities between droplets in turbulent airflow." In: *Physics of Fluids* 26.11 (Nov. 2014). arXiv: 1407.1766, p. 111702. ISSN: 1070-6631, 1089-7666. DOI: 10.1063/1.4900848.

[148] Martin B. Schlaffer. "Non-reflecting boundary conditions for the lattice Boltzmann method." PhD Thesis. Technische Universität München, 2013.

[149] Theodore G. Shepherd. "A Common Framework for Approaches to Extreme Event Attribution." In: *Current Climate Change Reports* 2.1 (2016), pp. 28–38. ISSN: 2198-6061. DOI: 10.1007/s40641-016-0033-y.

[150] Sujeet Kumar Shukla, Prashant Shukla, and Pradyumna Ghosh. "The effect of modeling of velocity fluctuations on prediction of collection efficiency of cyclone separators." In: *Applied Mathematical Modelling* 37.8 (Apr. 2013), pp. 5774–5789. ISSN: 0307-904X. DOI: 10.1016/j.apm.2012.11.019.

[151] Eric Simonnet. "Combinatorial analysis of the adaptive last particle method." In: *Stat. Comput.* 26 (2016), pp. 211–230. DOI: 10.1007/s11222-014-9489-6.

[152] P. A. Skordos. "Initial and boundary conditions for the lattice Boltzmann method." In: *Physical Review E* 48.6 (Dec. 1993), pp. 4823–4842. DOI: 10.1103/PhysRevE.48.4823.

[153] P R Spalart. "Strategies for turbulence modelling and simulations." In: *Int. J. Heat Fluid Flow* 21.3 (2000), pp. 252–263. DOI: 10.1016/S0142-727X(00)00007-2.

[154] S. Succi, R. Benzi, and F. Massaioli. "A review of the lattice boltzmann method." In: *International Journal of Modern Physics C* 04.02 (Apr. 1993), pp. 409–415. ISSN: 0129-1831. DOI: 10.1142/S0129183193000446.

[155] Sauro Succi. *The Lattice Boltzmann Equation: For Fluid Dynamics and Beyond*. Numerical Mathematics and Scientific Computation. Oxford, New York: Oxford University Press, June 2001. ISBN: 978-0-19-850398-9.

[156] OGB Sveinsson, J D Salas, and C D Boes. "Regional frequency analysis of extreme precipitation in Northeastern Colorado and Fort Collins flood of 1997." English. In: *Journal of Hydrologic Engineering* 7.1 (2002), pp. 49–63.

[157] Mathieu Szmigiel. "Etude du flux de soubassement sur la dynamique du sillage d'un corps non profilé à culot droit : Application du contrôle actif pour la réduction de traînée de véhicule industriel." 2017LYSEC016. PhD thesis. 2017. URL: http://www.theses.fr/2017LYSEC016/document.

[158] Julien Tailleur and Jorge Kurchan. "Probing rare physical trajectories with Lyapunov weighted dynamics." en. In: *Nature Physics* 3.3 (Mar. 2007), pp. 203–207. ISSN: 1745-2481. DOI: 10.1038/nphys515. (Visited on 05/31/2018).

[159] Y Tamura, H Kikuchi, and K Hibi. "Extreme wind pressure distributions on low-rise building models." In: *Journal of Wind Engineering and Industrial Aerodynamics*. Bluff Body Aerodynamics and Applications 89.14 (Dec. 2001), pp. 1635–1646. ISSN: 0167-6105. DOI: 10.1016/S0167-6105(01)00153-2.

[160] Tomás Tangarife. "Théorie cinétique et grandes déviations en dynamique des fluides géophysiques." 2015ENSL1037. PhD thesis. 2015. URL: http://www.theses.fr/2015ENSL1037.

[161] Ivan Teo, Christopher G. Mayne, Klaus Schulten, and Tony Lelièvre. "Adaptive Multilevel Splitting Method for Molecular Dynamics Calculation of Benzamidine-Trypsin Dissociation Time." In: *Journal of Chemical Theory and Computation* 12.6 (June 2016), pp. 2983–2989. DOI: 10.1021/acs.jctc.6b00277.

[162] B. Thiria, O. Cadot, and J.-F. Beaudoin. "Drag fluctuations of a disk in a turbulent jet: Effect of turbulent scales averaging." en. In: *EPL (Europhysics Letters)* 87.4 (2009), p. 44007. ISSN: 0295-5075. DOI: 10.1209/0295-5075/87/44007.

[163] Raùl Toral and Pere Colet. *Stochastic Numerical Methods: An Introduction for Students and Scientists*. en. Google-Books-ID: N2vsAwAAQBAJ. John Wiley & Sons, June 2014. ISBN: 978-3-527-68312-3.

[164] Hugo Touchette. "The large deviation approach to statistical mechanics." In: *Phys. Rep.* 478 (2009), pp. 1–69. DOI: 10.1016/j.physrep.2009.05.002.

[165] Hugo Touchette. "The large deviation approach to statistical mechanics." In: *Physics Reports* 478.1-3 (July 2009). arXiv: 0804.0327, pp. 1–69. ISSN: 03701573. DOI: 10.1016/j.physrep.2009.05.002.

[166] Hugo Touchette. "A basic introduction to large deviations: Theory, applications, simulations." In: *arXiv:1106.4146 [cond-mat, physics:math-ph]* (June 2011). arXiv: 1106.4146.

[167] Umesh V. Vazirani. "Go-With-The-Winners Heuristic." en. In: *Algorithms and Data Structures*. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, Aug. 1999, pp. 217–218. ISBN: 978-3-540-66279-2 978-3-540-48447-9. DOI: 10.1007/3-540-48447-7_22. (Visited on 06/11/2018).

[168] E. Vergnault, O. Malaspinas, and P. Sagaut. "A lattice Boltzmann method for nonlinear disturbances around an arbitrary base flow." In: *Journal of Computational Physics* 231.24 (Oct. 2012), pp. 8070–8082. ISSN: 0021-9991. DOI: 10.1016/j.jcp.2012.07.021.

[169] Etienne Vergnault, Orestis Malaspinas, and Pierre Sagaut. "Noise source identification with the lattice Boltzmann method." In: *The Journal of the Acoustical Society of America* 133.3 (Mar. 2013), pp. 1293–1305. ISSN: 0001-4966. DOI: 10.1121/1.4776181.

[170] B. J. Vickery. "Fluctuating lift and drag on a long cylinder of square cross-section in a smooth and in a turbulent stream." en. In: *Journal of Fluid Mechanics* 25.3 (July 1966), pp. 481–494. ISSN: 1469-7645, 0022-1120. DOI: 10.1017/S002211206600020X. (Visited on 06/13/2018).

[171] M. Villen-Altamirano and J. Villen-Altamirano. "RESTART: a straightforward method for fast simulation of rare events." In: *Proceedings of Winter Simulation Conference*. Dec. 1994, pp. 282–289. DOI: 10.1109/WSC.1994.717150.

[172] Manuel Villén-Altamirano and José Villén-Altamirano. "RESTART: A method for accelerating rare event simulations." In: 3 (Jan. 1991).

[173] Manuel Villén-Altamirano and José Villén-Altamirano. "The Rare Event Simulation Method RESTART: Efficiency Analysis and Guidelines for Its Application." en. In: *Network Performance Engineering*. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, 2011, pp. 509–547. ISBN: 978-3-642-02741-3 978-3-642-02742-0. DOI: 10.1007/978-3-642-02742-0_22. (Visited on 06/13/2018).

[174] Angelo Vulpiani, Fabio Cecconi, Massimo Cencini, Andrea Puglisi, and Davide Vergni. *Large Deviations in Physics*. English. The Legacy of the Law of Large Numbers. Springer, 2014. ISBN: 3642542514.

[175] Wikibooks. *Memory Management/Garbage Collection — Wikibooks, The Free Textbook Project*. [Online; accessed 19-August-2018]. 2015. URL: https://en.wikibooks.org/w/index.php?title=Memory_Management/Garbage_Collection&oldid=3023861.

[176] Dieter A. Wolf-Gladrow. *Lattice-Gas Cellular Automata and Lattice Boltzmann Models: An Introduction*. en. Lecture Notes in Mathematics. Berlin Heidelberg: Springer-Verlag, 2000. ISBN: 978-3-540-66973-9. (Visited on 06/13/2018).

[177] J. Wouters and F. Bouchet. "Rare event computation in deterministic chaotic systems using genealogical particle analysis." en. In: *Journal of Physics A: Mathematical and Theoretical* 49.37 (2016), p. 374002. ISSN: 1751-8121. DOI: 10.1088/1751-8113/49/37/374002.

[178] P. K. Yeung, X. M. Zhai, and Katepalli R. Sreenivasan. "Extreme events in computational turbulence." In: *Proceedings of the National Academy of Sciences of the United States of America* 112.41 (Oct. 2015), pp. 12633–12638. ISSN: 0027-8424. DOI: 10.1073/pnas.1517368112.

[179] P K Yeung, X M Zhai, and Katepalli R Sreenivasan. "Extreme events in computational turbulence." English. In: *Proc. Natl. Acad. Sci. U.S.A.* 112.41 (2015), pp. 12633–12638. DOI: 10.1073/pnas.1517368112.

COLOPHON

This document was typeset using the typographical look-and-feel `classicthesis` developed by André Miede and Ivo Pletikosić. The style was inspired by Robert Bringhurst's seminal book on typography "*The Elements of Typographic Style*". `classicthesis` is available for both LaTeX and LyX:

https://bitbucket.org/amiede/classicthesis/

Happy users of `classicthesis` usually send a real postcard to the author, a collection of postcards received so far is featured here:

http://postcards.miede.de/

Thank you very much for your feedback and contribution.

Final Version as of December 17, 2018 (classicthesis v4.6).