Simulating Incompressible Fluid Flows with Uncertainty Using Lattice Boltzmann Methods

Zur Erlangung des akademischen Grades eines

DOKTORS DER NATURWISSENSCHAFTEN

von der KIT-Fakultät für Mathematik
des Karlsruher Instituts für Technologie (KIT)

genehmigte

DISSERTATION

von

M.Sc. Mingliang Zhong
geb. in Shaanxi, China

Tag der mündlichen Prüfung:	12.11.2025
Hauptreferent:	PD Dr. Mathias J. Krause
Korreferent:	Prof. Dr. Martin Frank

Abstract

Uncertainty quantification (UQ) has become a critical component in computational fluid dynamics (CFD), particularly for assessing the reliability of simulation results in the presence of uncertain parameters such as inlet velocity, viscosity, or boundary conditions. The lattice Boltzmann method (LBM), a mesoscopic CFD technique known for its parallel scalability and flexibility with complex geometries, provides a promising platform for integrating UQ. However, systematic UQ capabilities remain underdeveloped in current LBM-based frameworks.

This dissertation develops a unified UQ framework for incompressible LBM simulations, combining both non-intrusive and intrusive approaches. The non-intrusive strategy is implemented by extending the open-source OpenLB library with a modular UQ module that supports Monte Carlo sampling, quasi-Monte Carlo methods, and stochastic collocation (SC) based on generalized polynomial chaos (gPC). The OpenLB-UQ module automates sampling, parallel execution, and statistical post-processing, enabling scalable UQ workflows. Validation on canonical benchmarks—such as the Taylor–Green vortex and flow past a cylinder—demonstrates accurate moment estimation and parallel performance gains.

In parallel, the first fully coupled stochastic Galerkin (SG) LBM is proposed, which reformulates the LBM equations using polynomial chaos expansions. The proposed intrusive SG LBM directly evolves the coefficients of the discrete-velocity distribution functions obtained via polynomial chaos expansion. Its algorithmic structure fully preserves the LBM streaming and collision processes, ensuring structural consistency with standard LBM frameworks. The SG LBM achieves spectral convergence and reduces computational cost by a factor of five to six compared to Monte Carlo methods in representative cases.

To demonstrate real-world applicability, we applied an uncertain data assimilation workflow based on OpenLB-UQ to an urban wind simulation around an isolated building in Reutlingen (Germany). Measurement uncertainty was directly injected into the inflow boundary data and propagated through a non-intrusive SC LBM pipeline, yielding spatio-temporal statistics of the velocity field across the domain. The workflow enabled the computation of mean and standard deviation fields, the identification of flow-sensitive zones such as wakes and shear layers, and the derivation of confidence intervals at monitoring probes. This provided interpretable uncertainty maps that respect the statistical nature of measurement-driven inflow and highlighted their relevance for urban planning and wind engineering applications.

Together, these contributions provide a scalable and comprehensive, open-source UQ toolkit for LBM-based CFD. Conclusively, this work advances efficient uncertainty-aware simulations of incompressible flows across scientific and engineering domains.

Aknowledgements

First, I would like to express my sincere gratitude to my collaborator, Dr. Stephan Simonis, for his continuous guidance, valuable advice, and many inspiring discussions throughout this work. His support has played a key role in shaping both my research and my professional development.

I am also thankful to my supervisors, PD Dr. Mathias J. Krause and Prof. Martin Frank, for their feedback and direction during my doctoral studies.

I would like to acknowledge my colleagues, co-workers, and co-authors for their collaboration, insightful exchanges, and shared efforts, all of which have greatly enriched this research.

The computational results presented in this dissertation were made possible through computing time on the high-performance computer HoreKa at the National High-Performance Computing Center at KIT (NHR@KIT). This center is jointly supported by the Federal Ministry of Education and Research and the Ministry of Science, Research and the Arts of Baden-Württemberg, with partial funding from the German Research Foundation (DFG).

I am deeply grateful to my friends and family for their encouragement and support during this time.

Above all, I wish to express my deepest love and gratitude to my wife Jingya. Her love, patience, and encouragement have carried me through the most difficult phases of this journey. She has shared the sacrifices, endured the long hours, and stood by me with unfailing support.

Contents

1 Introduction

Uncertainty Quantification (UQ) plays an essential role in computational modeling by rigorously characterizing the influence of input uncertainties on simulation outcomes [73, 76]. In Computational Fluid Dynamics (CFD), such uncertainties may arise from model parameters, boundary conditions, or geometric and environmental factors [58]. Without accounting for them, deterministic simulations may yield misleading predictions and compromise the robustness of engineering decisions. For example, UQ has been applied to ship motion prediction in irregular waves [14], heat transfer in turbine blade cooling [61], and pollutant dispersion in urban environments [60].

UQ methods are typically classified into non-intrusive and intrusive categories, see Figure 1.1. Non-intrusive techniques, such as Monte Carlo sampling (MCS), treat the solver as a black box and are favored for their generality and ease of implementation, although they suffer from slow convergence. To address this, more efficient variants have been developed, including quasi Monte Carlo (QMC) [57], multi-level Monte Carlo (MLMC) [22], and stochastic collocation (SC) methods based on generalized polynomial chaos (gPC) [2, 91]. Intrusive approaches, such as the stochastic Galerkin (SG) method, modify the governing equations to explicitly represent uncertainty, enabling spectral convergence when the model response is sufficiently smooth [90, 93].

There is now an expanding research field that integrates UQ with CFD across a wide range of flow regimes, numerical schemes, and UQ methodologies. Early intrusive and non-intrusive approaches applied gPC expansions to incompressible Navier–Stokes equations(NSE) [89]. MLMC methods have been implemented for problems ranging from elliptic diffusion [4] to unsteady open–cavity flows [5] and aerospace applications [20]. For turbulent Reynolds-averaged Navier–Stokes (RANS) simulations, Bayesian calibration techniques have been employed to quantify model-form uncertainty [59]. UQ for hyperbolic systems of conservation laws has also been systematically reviewed, encompassing SG and SC methods, MC and MLMC techniques, as well as alternative formulations such as measure-valued and statistical solutions [1]. Recent developments include intrusive high-order discontinuous Galerkin methods that incorporate SG projections to simulate compressible flows under uncertainty [6]. Collectively, these contributions demonstrate the increasing integration of sampling-based, projection-based, Bayesian UQ frameworks into modern CFD workflows.

The Lattice Boltzmann Method (LBM) is a mesoscopic CFD technique well-suited for incompressible and weakly compressible flows, offering simplicity, excellent parallel scalability, and adaptability to complex geometries [24, 75]. OpenLB [33, 36], a modern open-source LBM framework, provides a robust foundation for integrating uncertainty quantification. Non-intrusive UQ techniques have been

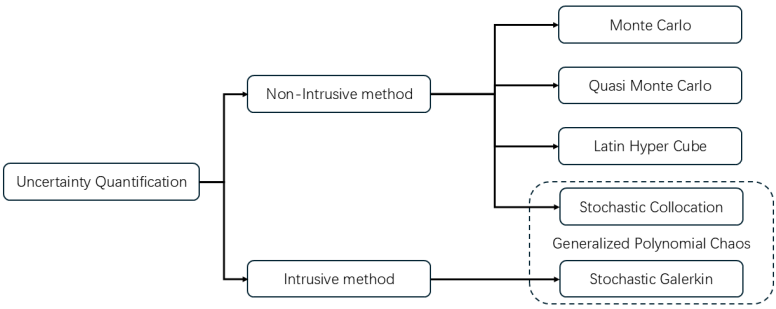


Figure 1.1: Classification of UQ methods into non-intrusive and intrusive categories. Non-intrusive approaches include MCS, QMC (with LHS), and SC. Intrusive approaches involve reformulation of the governing equations, such as the stochastic Galerkin method.

successfully applied in this context, such as the recent work by McCullough et al. [51], which used SC-gPC to analyze uncertainty in human-scale vascular flow simulations. Other studies have employed collocation and sparse grid methods for porous media and urban wind flow applications simulated with OpenLB [29, 84]. Further, the MC LBM has been used to compute statistical solutions of the incompressible Navier–Stokes and Euler equations for the first time in [62, 65] and enabled large-scale training data generation for generative diffusion models such as GenCFD [56]. In contrast, intrusive formulations remain relatively underdeveloped. Some prior works reformulate macroscopic equations using SG projection [94], while others apply SG to simplified LBM models [16], but a general-purpose intrusive UQ framework directly based on the standard LBM has yet to be established.

This dissertation addresses this gap by developing a comprehensive and scalable UQ framework for incompressible flows using the LBM. On the one hand, it introduces a non-intrusive UQ module integrated into OpenLB, enabling efficient MC LBM, QMC LBM, and SC LBM simulations with automated sampling and post-processing, while retaining compatibility with the existing parallel infrastructure. While demonstrated on fluid dynamics cases, this framework is general and readily applicable to other mesoscopic models currently supported by OpenLB, including thermal transport, radiative transfer, and reactive flows.

On the other hand, it presents the first full implementation of a SG LBM, which solves for the evolution of uncertainty modes within the mesoscopic LBM framework. Comparative benchmarks on canonical test cases, such as the Taylor–Green vortex (TGV) and lid-driven cavity flow (LDC), demonstrate that the SG LBM achieves spectral convergence and offers substantial speed-up of up to five to six

times faster than traditional MC simulations—especially when the model response is smooth and the number of uncertain parameters remains moderate.

By developing both a non-intrusive UQ extension to OpenLB and a separate SG LBM solver, this work advances the methodological and software foundations for uncertainty quantification in CFD. These complementary tools enable robust, uncertainty-aware simulations across a range of fluid dynamics applications, from biomedical engineering to environmental flow modeling.

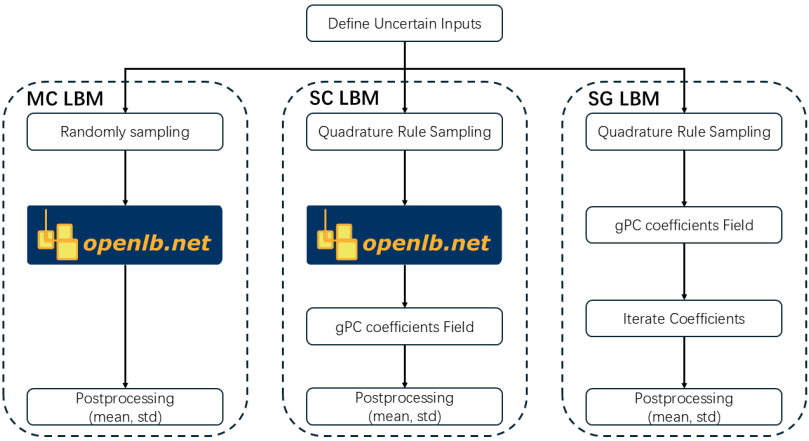


Figure 1.2: Overview of the UQ strategies used in this dissertation. The left two branches represent non-intrusive UQ methods: MC LBM using random sampling, and SC LBM using quadrature rule sampling. The right branch illustrates the intrusive SG LBM, which solves a coupled system for the gPC coefficients.

1.1 Research objectives and contributions

The overarching aim of this dissertation is to advance UQ for incompressible fluid flow simulations by embedding both non-intrusive and intrusive methodologies directly into the LBM framework.

Achieving this aim requires a dual focus:

1. extending the open-source library OpenLB with an efficient, scalable non-intrusive UQ module
2. formulating the first SG LBM that evolves uncertainty modes within the mesoscopic solver itself.

The research is therefore structured around five tightly inter-linked objectives, each paired with a concrete contribution.

Objective 1: Non-intrusive UQ framework for LBM.
The first objective is to design and implement a UQ extension that integrates MCS, QMC, SC into OpenLB, while preserving the modularity and high performance of the original solver.
Contribution: This effort results in the OpenLB-UQ module, which provides automated sampling, parallel execution, and statistical post-processing, all seamlessly integrated into the existing LBM code base.

Objective 2: Verification on canonical benchmarks.
To validate the correctness and efficiency of the developed framework, the second objective focuses on numerical experiments involving the Taylor–Green vortex and cylinder-flow benchmarks under stochastic inputs. The performance of the methods is assessed through convergence studies and parallel scaling analyses.
Contribution: The dissertation provides a benchmark suite and reproducible reference data that quantify the accuracy and efficiency of the non-intrusive UQ methods within the LBM context.

Objective 3: Comparative performance assessment.
A systematic comparison is conducted between non-intrusive and intrusive UQ approaches to evaluate their accuracy, computational cost, and parallel scalability.
Contribution: The study offers practical guidance on selecting appropriate UQ methods for different problem classes, balancing the trade-offs between accuracy, dimensionality, and computational budget.

Objective 4: Application to an urban-airflow scenario.
To demonstrate practical applicability, the framework is applied to an urban flow case with uncertain wind inflow conditions. The SC-gPC method is used to propagate uncertainty in measurements of wind speed and direction across a realistic domain in a data-assimilated LBM simulation.
Contribution: A surrogate-based workflow is developed that captures the stochastic behavior of key flow features while reducing computational cost by an order of magnitude compared to direct sampling.

Objective 5: Development of an intrusive SG LBM.
The final objective is to develop a fully intrusive SG formulation of the LBM. This includes reformulating the LB equation in terms of gPC, deriving the coupled equations for the gPC coefficients, and addressing numerical stability for stochastic collision operators.
Contribution: The result is the first SG LBM solver for incompressible flows, offering spectral convergence and demonstrating a 5–6× speed-up over MC LBM in representative test cases [98].

By fulfilling these objectives, this dissertation provides a comprehensive and extensible UQ framework for LBM-based incompressible flow simulations. It lowers the barrier to uncertainty-aware CFD in

both academic and engineering contexts and lays the foundation for future developments, such as adaptive or hybrid UQ strategies tailored for modern high-performance computing platforms.

1.2 Structure of the thesis

The dissertation is organized into three parts comprising seven chapters.

- **Chapter 1** introduces the context of UQ in CFD, defines the research objectives, and outlines the dissertation structure. It motivates the use of both non-intrusive and intrusive UQ methods with LBM.
- **Chapter 2** reviews the mathematical foundations of incompressible CFD and presents the LBM as a scheme constructed to recover the incompressible Navier–Stokes equations, emphasizing its consistency and its advantages for UQ, such as locality and parallel scalability.
- **Chapter 3** surveys UQ theory, covering sampling-based methods, gPC, and sparse-grid techniques. It explains how these approaches propagate input uncertainty and form the basis for the non-intrusive and intrusive methods developed later.
- **Chapter 4** presents the intrusive SG LBM, derived by applying Galerkin projection to the polynomial-expanded LB equations. It validates the method on canonical benchmarks, showing preserved accuracy and up to 5.7× speedup over MC LBM, highlighting SG LBM as an efficient intrusive UQ approach for CFD.
- **Chapter 5** introduces the OpenLB-UQ software module and its non-intrusive workflow.
- **Chapter 5** introduces the OpenLB-UQ software module, which implements a non-intrusive UQ workflow using MC LBM, QMC LBM, and SC LBM methods. It details the integration into OpenLB and demonstrates automated sampling, parallel execution, and post-processing capabilities.
- **Chapter 6** evaluates non-intrusive UQ methods for LBM through convergence and parallel-scaling studies on benchmark cases. The results confirm spectral accuracy of SC LBM and highlight the scalability and efficiency of the OpenLB-UQ module.
- **Chapter 7** applies the complete UQ framework to an urban-wind simulation with uncertain inflow conditions. Using SC LBM, it quantifies the impact of wind speed and direction variability and demonstrates the framework's applicability to realistic, data-driven CFD problems.

Overall, the streamlined structure of this work spans theoretical foundations, methodological developments, extensive validation on benchmarks, and a real-world application to urban airflow simulation under uncertainty.

2 Mathematical Background

This chapter recalls the mathematical foundations for simulating fluid dynamics in the presence of uncertainty. We begin with a review of the deterministic incompressible NSE, followed by its extension into a stochastic framework. The LBM is then introduced from a kinetic theory perspective, leading to its discrete form suitable for numerical simulation. Finally, we formulate a stochastic LBM that incorporates parametric uncertainties, laying the groundwork for subsequent uncertainty propagation techniques.

This section is organized as follows. Section 2.1 presents the deterministic NSE and their extension to account for uncertainty. Section 2.2 details the derivation and key components of the LBM, followed by its extension to incorporate uncertainty.

2.1 Incompressible Navier–Stokes equations with uncertainty

The incompressible NSE describes the motion of a viscous, incompressible fluid. In the deterministic form, they are given by

$$\partial_t \boldsymbol{u} + (\boldsymbol{u} \cdot \boldsymbol{\nabla})\boldsymbol{u} - \nu \boldsymbol{\nabla}^2 \boldsymbol{u} = -\frac{1}{\rho}\boldsymbol{\nabla} p + \boldsymbol{g}, \quad \text{in } \mathcal{X} \times \mathcal{I} \subseteq \mathbb{R}^{d_x} \times \mathbb{R}_{>0}, \tag{2.1}$$

$$\boldsymbol{\nabla} \cdot \boldsymbol{u} = 0, \quad \text{in } \mathcal{X} \times \mathcal{I}, \tag{2.2}$$

with appropriate boundary and initial conditions, which model the behavior of Newtonian fluids at low Mach numbers. Here, $\rho > 0$ denotes fluid density, $\boldsymbol{u} \colon \mathcal{X} \times \mathcal{I} \to \mathbb{R}^{d_x}$ represents the velocity vector, $p \colon \mathcal{X} \times \mathcal{I} \to \mathbb{R}$ is the pressure, $\nu > 0$ is the kinematic viscosity, and $\boldsymbol{g} \colon \mathcal{X} \times \mathcal{I} \to \mathbb{R}^{d_x}$ represents an external force. In our study, the external force is neglected ($\boldsymbol{g} = \boldsymbol{0}$).

In many practical fluid dynamics applications, precise knowledge of the model parameters, boundary conditions, or initial conditions is often unavailable. This lack of information introduces uncertainty into the system and motivates the incorporation of uncertainty into the governing equations.

To incorporate uncertainty, we extend the deterministic NSE (Eq. (2.1), Eq. (2.2)) to a stochastic framework by introducing random variables into selected input parameters. Let $\boldsymbol{Z} = (Z_1, ..., Z_d) \in \mathcal{Z} \subseteq \mathbb{R}^{d_Z}$ be a vector of independent random variables defined on a probability space $(\Xi, \mathcal{F}, \mathbb{P})$. The stochastic form of the NSE thus reads

$$\partial_t \boldsymbol{u}(\boldsymbol{Z}) + (\boldsymbol{u}(\boldsymbol{Z}) \cdot \boldsymbol{\nabla})\boldsymbol{u}(\boldsymbol{Z}) - \nu(\boldsymbol{Z})\boldsymbol{\nabla}^2\boldsymbol{u}(\boldsymbol{Z}) = -\frac{1}{\rho}\boldsymbol{\nabla}p(\boldsymbol{Z}), \quad \text{in } \mathcal{X} \times \mathcal{I} \times \mathcal{Z}, \tag{2.3}$$

$$\boldsymbol{\nabla} \cdot \boldsymbol{u}(\boldsymbol{Z}) = 0, \quad \text{in } \mathcal{X} \times \mathcal{I} \times \mathcal{Z}, \tag{2.4}$$

where we neglected the space-time arguments $(\boldsymbol{x}, t) \in \mathcal{X} \times \mathcal{I}$ of $\boldsymbol{u}$ for the sake of readability. In this work, we consider both one- and multi-dimensional uncertainties. Two representative cases are examined, based on uncertainty in the velocity field $\boldsymbol{u}(\boldsymbol{Z})$ induced by (i) stochastic boundary or initial conditions or (ii) uncertainty in the viscosity $\nu(\boldsymbol{Z})$. In both cases, the random variables are uniformly distributed. Notably, for a fixed vector $\boldsymbol{Z}$, the stochastic NSE (Eq. (2.3), Eq. (2.4)) becomes a version of the deterministic NSE (Eq. (2.1), Eq. (2.2)), which forms the basis of the non-intrusive methodology in our framework.

This stochastic formulation provides the mathematical foundation for the uncertainty quantification techniques introduced later. It enables us to analyze how uncertain inputs influence the output quantities of interest, such as velocity fields, pressure distributions, or integral quantities like drag and lift.

2.2 Lattice Boltzmann method

The LBM is a mesoscopic numerical scheme for fluid dynamics, bridging the gap between molecular and continuum scales. Its foundation lies in kinetic theory, particularly the Boltzmann transport equation, which governs the evolution of the single-particle distribution function. In this section, we recall basic principles of LBM and subsequently extend it with a random variable in the arguments to account for uncertainty in the numerical scheme.

2.2.1 From kinetic theory to the Boltzmann–BGK equation

The Boltzmann equation (BE) governs the evolution of the single-particle distribution function $f\colon \mathcal{X} \times \mathbb{R}^{d_\xi} \times \mathcal{I} \to \mathbb{R}_{\geq 0}$, which describes the probability density of finding a particle at position $\boldsymbol{x} \in \mathcal{X} \subseteq \mathbb{R}^{d_x}$, with velocity $\boldsymbol{\xi} \in \mathbb{R}^{d_\xi}$, at time $t \in \mathcal{I} \subseteq \mathbb{R}_{>0}$. The equation reads

$$\partial_t f + \boldsymbol{\xi} \cdot \boldsymbol{\nabla}_{\boldsymbol{x}} f = \Omega(f), \quad \text{in } \mathcal{X} \times \mathbb{R}^{d_\xi} \times \mathcal{I}, \tag{2.5}$$

where $\Omega(f)$ denotes the collision operator, which encapsulates the effects of particle collisions. In general, $\Omega(f)$ is a nonlinear integral operator, making the direct numerical solution of the BE computationally expensive and challenging.

To simplify the collision operator $\Omega(f)$, the Bhatnagar–Gross–Krook (BGK) approximation [7] is commonly used. It models particle collisions as a relaxation toward a local equilibrium distribution and is defined by

$$\Omega(f) = -\frac{1}{\tau}\left(f - f^{\mathrm{eq}}\right), \tag{2.6}$$

where $\tau > 0$ is the relaxation time, and $f^{\mathrm{eq}} = f^{\mathrm{eq}}(\boldsymbol{x}, \boldsymbol{\xi}, t)$ denotes the Maxwell–Boltzmann equilibrium distribution function. This approximation reduces the complexity of the collision term while retaining essential physical properties such as mass, momentum, and energy conservation.

Substituting the BGK approximation Eq. (2.6) into the BE Eq. (2.5) yields the Boltzmann–BGK equation

$$\partial_t f + \boldsymbol{\xi} \cdot \boldsymbol{\nabla}_{\boldsymbol{x}} f = -\frac{1}{\tau}\left(f - f^{\mathrm{eq}}\right) \quad \text{in } \mathcal{X} \times \mathbb{R}^{d_\xi} \times \mathcal{I}. \tag{2.7}$$

The macroscopic fluid quantities, namely the density ρ and velocity $\boldsymbol{u}$ are computed as moments of the distribution function with respect to the microscopic velocity $\boldsymbol{\xi}$:

$$\rho(\boldsymbol{x}, t) = \int_{\mathbb{R}^{d_\xi}} f(\boldsymbol{x}, \boldsymbol{\xi}, t)\, d\boldsymbol{\xi} = \int_{\mathbb{R}^{d_\xi}} f^{\mathrm{eq}}(\boldsymbol{x}, \boldsymbol{\xi}, t)\, d\boldsymbol{\xi}, \tag{2.8}$$

$$\rho(\boldsymbol{x}, t)\, \boldsymbol{u}(\boldsymbol{x}, t) = \int_{\mathbb{R}^{d_\xi}} \boldsymbol{\xi}\, f(\boldsymbol{x}, \boldsymbol{\xi}, t)\, d\boldsymbol{\xi} = \int_{\mathbb{R}^{d_\xi}} \boldsymbol{\xi}\, f^{\mathrm{eq}}(\boldsymbol{x}, \boldsymbol{\xi}, t)\, d\boldsymbol{\xi}. \tag{2.9}$$

In lattice units, the kinematic viscosity ν is related to the relaxation time τ by

$$\nu = c_s^2 \left(\tau - \frac{1}{2}\right) \triangle t,$$

where c_s is the lattice speed of sound and $\triangle t$ is the time step.

This kinetic formulation provides the theoretical foundation for the LBM. The next subsection introduces its discrete form by applying velocity and spatial discretization to derive the fully discrete LB equation (LBE).

2.2.2 Lattice Boltzmann equation

The LBM provides a mesoscopic approach for simulating fluid dynamics by tracking the evolution of particle distribution functions $f_i(\boldsymbol{x}, t)$ along discrete velocity directions $\boldsymbol{c}_i$. In the absence of uncertainty, the deterministic LBE is given by

$$f_i(\boldsymbol{x} + \boldsymbol{c}_i \triangle t, t + \triangle t) = f_i(\boldsymbol{x}, t) - \frac{\triangle t}{\tau} \left(f_i(\boldsymbol{x}, t) - f_i^{\mathrm{eq}}(\boldsymbol{x}, t) \right), \quad \text{in } \mathcal{X}_{\triangle x} \times \mathcal{I}_{\triangle t} \tag{2.10}$$

where $\tau > \frac{1}{2}$ is the relaxation time, and f_i^{eq} is the local equilibrium distribution function. The LBE Eq. (2.10) can be derived from discretizing the continuous BGK-Boltzmann equation in velocity, space, and time. For further details, see e.g. [44, 64].

Through the Chapman–Enskog (CE) expansion [11], it can be shown that the Boltzmann–BGK equation Eq. (2.7) asymptotically recovers the incompressible NSE in the low Mach number regime.

2.2.3 Discrete velocity models

In the LBM, the continuous velocity space $\mathbb{R}^{d_\xi}$ from kinetic theory is discretized into a finite set of discrete velocities $\{\boldsymbol{c}_i\}_{i=0}^{q-1} \subset \mathbb{R}^{d_q}$, where q is the number of discrete velocities. The discrete set is constructed to ensure isotropy of the velocity moments and to recover the incompressible NSE up to second order through a suitable moment expansion.

For two-dimensional flows, we adopt the widely used D2Q9 model, which consists of $q = 9$ discrete velocities

$$\boldsymbol{c}_i = \begin{cases} (0,0), & i = 0, \\ (\pm 1, 0),\ (0, \pm 1), & i = 1, 2, 3, 4, \\ (\pm 1, \pm 1), & i = 5, 6, 7, 8, \end{cases} \tag{2.11}$$

with corresponding weights w_i given by

$$w_i = \begin{cases} \frac{4}{9}, & i = 0, \\ \frac{1}{9}, & i = 1, \dots, 4, \\ \frac{1}{36}, & i = 5, \dots, 8. \end{cases} \tag{2.12}$$

For three-dimensional flows, the D3Q19 model is used, featuring $q = 19$ discrete velocities

$$\boldsymbol{c}_i = \begin{cases} (0,0,0), & i = 0, \\ (\pm 1,0,0),\ (0,\pm 1,0),\ (0,0,\pm 1), & i = 1,\dots,6, \\ (\pm 1,\pm 1,0),\ (\pm 1,0,\pm 1),\ (0,\pm 1,\pm 1), & i = 7,\dots,18, \end{cases} \tag{2.13}$$

with the corresponding weights

$$w_i = \begin{cases} \frac{1}{3}, & i = 0, \\ \frac{1}{18}, & i = 1,\dots,6, \\ \frac{1}{36}, & i = 7,\dots,18. \end{cases} \tag{2.14}$$

These discrete velocity sets are designed to satisfy isotropy of the second-order moments, a crucial requirement for recovering the correct viscous stress tensor in the macroscopic equations. Additionally, their Cartesian structure enables efficient streaming operations on uniform grids.

In this work, two-dimensional simulations are performed using the D2Q9 model and three-dimensional simulations are performed using the D3Q19 model, respectively with uniform lattice spacing and time step.

2.2.4 Equilibrium distribution function

The equilibrium distribution function f_i^{eq} plays a central role in LBM and ensures the correct recovery of macroscopic hydrodynamics. For incompressible flows in the low-Mach number limit, f_i^{eq} is typically given by a second-order expansion of the Maxwell–Boltzmann distribution:

$$f_i^{\mathrm{eq}} = w_i \rho \left(1 + \frac{\boldsymbol{c}_i \cdot \boldsymbol{u}}{c_s^2} + \frac{(\boldsymbol{c}_i \cdot \boldsymbol{u})^2}{2c_s^4} - \frac{\boldsymbol{u} \cdot \boldsymbol{u}}{2c_s^2} \right), \tag{2.15}$$

where

$$\rho(\boldsymbol{x}, t) = \sum_{i=0}^{q-1} f_i(\boldsymbol{x}, t), \tag{2.16}$$

$$\boldsymbol{u}(\boldsymbol{x}, t) = \frac{1}{\rho(\boldsymbol{x}, t)} \sum_{i=0}^{q-1} f_i(\boldsymbol{x}, t) \tag{2.17}$$

approximate the macroscopic fluid density, and the macroscopic fluid viscosity, respectively, c_s is the lattice speed of sound (with $c_s^2 = 1/3$ in lattice units for the here used velocity sets) Eq. (2.11) Eq. (2.12), $\boldsymbol{c}_i$ are the respective discrete velocities and w_i denote the associated quadrature weights

Eq. (2.13) Eq. (2.14). The equilibrium distribution Eq.(2.15) satisfies the following moment constraints:

$$\sum_i f_i^{\mathrm{eq}} = \rho, \tag{2.18}$$

$$\sum_i \boldsymbol{c}_i f_i^{\mathrm{eq}} = \rho \boldsymbol{u}, \tag{2.19}$$

ensuring consistency with the continuity and momentum equations, respectively.

2.2.5 Lattice Boltzmann equation with uncertainty

To incorporate uncertainty, the LBM is extended by modeling selected parameters, such as viscosity, boundary conditions, or initial fields, as random variables. Let $\boldsymbol{Z} = (Z_1, \ldots, Z_{d_Z}) \in \mathbb{R}^{d_Z}$ be a vector of independent random variables defined on a probability space $(\Xi, \mathcal{F}, \mathbb{P})$, capturing the system's parametric uncertainty.

As a consequence, the distribution functions become stochastic fields $f_i(\boldsymbol{x}, t, \boldsymbol{Z})$, evolving under uncertain inputs. The stochastic LBE takes the form

$$f_i(\boldsymbol{x}+\boldsymbol{c}_i\triangle t, t+\triangle t, \boldsymbol{Z}) = f_i(\boldsymbol{x}, t, \boldsymbol{Z}) - \frac{\triangle t}{\tau(\boldsymbol{Z})}\left(f_i(\boldsymbol{x}, t, \boldsymbol{Z}) - f_i^{\mathrm{eq}}(\boldsymbol{x}, t, \boldsymbol{Z})\right), \quad \text{in } \mathcal{X}_{\triangle x}\times\mathcal{I}_{\triangle t}\times\mathcal{Z} \tag{2.20}$$

where $\tau(\boldsymbol{Z})$ is a stochastic relaxation time. Other uncertain quantities, such as the initial condition $f_i(\boldsymbol{x}, 0, \boldsymbol{Z})$ or inflow velocity $\boldsymbol{u}_{\mathrm{in}}(\boldsymbol{Z})$ are modeled similarly.

The stochastic kinematic viscosity is given by

$$\nu(\boldsymbol{Z}) = c_s^2\left(\tau(\boldsymbol{Z}) - \frac{1}{2}\right)\triangle t, \tag{2.21}$$

generalizing the standard viscosity relation to account for uncertainty. To ensure numerical stability and positivity of the viscosity, it is required that $\tau(\boldsymbol{Z}) > \frac{1}{2}$ almost surely, i.e., for all realizations of $\boldsymbol{Z}$.

The equilibrium distribution also depends on $\boldsymbol{Z}$ and is defined analogously to the deterministic case:

$$f_i^{\mathrm{eq}}(\boldsymbol{Z}) = w_i\rho(\boldsymbol{Z})\left(1 + \frac{\boldsymbol{c}_i\cdot\boldsymbol{u}(\boldsymbol{Z})}{c_s^2} + \frac{(\boldsymbol{c}_i\cdot\boldsymbol{u}(\boldsymbol{Z}))^2}{2c_s^4} - \frac{\boldsymbol{u}(\boldsymbol{Z})\cdot\boldsymbol{u}(\boldsymbol{Z})}{2c_s^2}\right). \tag{2.22}$$

The macroscopic fluid quantities are computed by taking velocity moments of the stochastic distribution functions, i.e.

$$\rho(\boldsymbol{Z}) = \sum_{i=0}^{q-1} f_i(\boldsymbol{Z}), \tag{2.23}$$

$$\rho(\boldsymbol{Z})\, \boldsymbol{u}(\boldsymbol{Z}) = \sum_{i=0}^{q-1} \boldsymbol{c}_i f_i(\boldsymbol{Z}). \tag{2.24}$$

Depending on the source of uncertainty, different scenarios can be modeled:

- **Uncertain viscosity:** $\tau(\boldsymbol{Z})$ varies, affecting the relaxation dynamics.
- **Uncertain inflow:** boundary velocity $\boldsymbol{u}_{\mathrm{in}}(\boldsymbol{Z})$ is prescribed by a probability distribution.
- **Uncertain initial fields:** Initial conditions $f_i(\boldsymbol{x}, 0, \boldsymbol{Z})$, or equivalently $\rho_0(\boldsymbol{x}, \boldsymbol{Z})$, $\boldsymbol{u}_0(\boldsymbol{x}, \boldsymbol{Z})$, are random.

Despite the introduction of uncertainty, the LBM algorithm retains its structure and efficiency. Each time step consists of:

$$\text{collision:} \quad f_i^{\star}\left(\boldsymbol{x}, t, \boldsymbol{Z}\right) = f_i\left(\boldsymbol{x}, t, \boldsymbol{Z}\right) - \frac{1}{\tau(\boldsymbol{Z})}\left(f_i\left(\boldsymbol{x}, t, \boldsymbol{Z}\right) - f_i^{\mathrm{eq}}\left(\boldsymbol{x}, t, \boldsymbol{Z}\right)\right), \tag{2.25}$$

$$\text{streaming:} \quad f_i\left(\boldsymbol{x} + \boldsymbol{c}_i \triangle t, t + \triangle t, \boldsymbol{Z}\right) = f_i^{\star}\left(\boldsymbol{x}, t, \boldsymbol{Z}\right), \tag{2.26}$$

where the upper index $\cdot^{\star}$ denotes post-collision variables.

Crucially, the stochastic formulation does not alter the algorithm's local communication pattern. Since the random variables $\boldsymbol{Z}$ are sampled independently for each realization, all samples evolve independently and can be computed in parallel. Within each realization, the LBM maintains its high data locality and is well suited for parallel execution on distributed-memory systems.

The stochastic LBE thus governs the evolution of a random process and provides a basis for uncertainty quantification. In this work, non-intrusive techniques such as MCS and gPC are employed to propagate input uncertainties and analyze their impact on macroscopic quantities of interest.

3 Uncertainty Quantification

This chapter introduces forward UQ, beginning with a general overview that distinguishes between aleatoric and epistemic uncertainty and motivates the focus on forward propagation of aleatoric variability. Next, uncertain inputs are modeled as random variables or fields on a probability space under simplifying assumptions. Building on this, numerical methods for forward propagation are presented, including sampling-based approaches such as MCS and QMC, as well as spectral methods based on gPC. The latter are discussed in both non-intrusive form, via SC with tensor-product or sparse-grid quadrature, and intrusive form, via SG projection.

This chapter is organized as follows. Section 3.1 introduces the foundations of UQ. Section 3.2 presents the modeling of uncertain inputs. Section 3.3 describes forward propagation techniques, covering sampling-based methods, non-intrusive SC-gPC, and the intrusive SG approach.

3.1 Introduction to uncertainty quantification

UQ provides a mathematical and computational framework for modeling, propagating, and analyzing uncertainties in physical systems and simulations. In CFD, uncertainties arise from various sources, including measurement errors, geometric tolerances, material variability, limited resolution, and model approximations. These uncertainties can significantly impact quantities of interest (QoIs), such as velocity, pressure, or drag coefficients, especially in nonlinear or multiscale flows.

Deterministic simulations, which assume exact input parameters, often fail to capture the variability inherent in real-world applications. UQ methods, in contrast, characterize the distribution of possible outcomes, enabling assessments of simulation reliability, robustness, and risk.

Uncertainties are commonly classified into two categories:

- **Aleatoric uncertainty**: Represents intrinsic randomness, such as turbulent fluctuations or environmental disturbances. It is typically modeled using probability distributions over input parameters.
- **Epistemic uncertainty**: Arises from incomplete knowledge, including unknown model coefficients or insufficient data. It is often reducible through additional information and addressed using Bayesian inference or model calibration.

This work focuses on **aleatoric uncertainty**, assuming that epistemic sources such as model form errors or incomplete boundary data are either negligible or incorporated into the prescribed variability of the input parameters.

In some applications, such as urban flow simulations [17, 77], input data (e.g., wind speed and direction) may contain measurement error and is traditionally viewed as epistemic. However, we adopt a modeling strategy that treats such uncertainties as *effective aleatoric variability*, represented via random variables or fields.

UQ tasks are typically divided into:

- **Forward UQ**: Propagating input uncertainties through the simulation to quantify their effect on output statistics e.g., mean, variance, or probability density of QoIs.
- **Inverse UQ**: Inferring uncertain model inputs from observational data, often within a Bayesian framework.

This dissertation is concerned solely with **forward UQ**, focusing on how input uncertainty influences fluid dynamic responses computed via the LBM. Forward UQ begins by modeling uncertain inputs as stochastic processes or random fields over a complete probability space $(\Xi, \mathcal{F}, \mathbb{P})$.

Given a deterministic model $\mathcal{M} : \mathbb{R}^{d_Z} \to \mathbb{R}^m$ and a random input vector $\boldsymbol{Z} \in \mathbb{R}^{d_Z}$ with probability law $\mathbb{P}_{\boldsymbol{Z}}$, cumulative distribution function (CDF) $F_{\boldsymbol{Z}}$, and probability density function (PDF) $\pi_{\boldsymbol{Z}}$, the goal is to compute statistical quantities of the model output $\mathcal{M}(\boldsymbol{Z})$, such as

$$\mathbb{E}[\mathcal{M}(\boldsymbol{Z})], \quad \mathrm{Var}[\mathcal{M}(\boldsymbol{Z})].$$

The remainder of this chapter introduces mathematical representations of uncertainties (random variables and fields), followed by a detailed discussion of numerical techniques for forward propagation.

3.2 Modeling uncertainty

A key step in uncertainty quantification is to represent uncertain inputs using appropriate mathematical structures. In this work, we focus on *parametric uncertainty*, where inputs, such as inflow profiles, material properties, or source terms—are modeled as random variables on a probability space $(\Xi, \mathcal{F}, \mathbb{P})$ (see Chapter 2 for formal definitions).

We adopt the following modeling assumptions:

1. all random variables are defined on a common probability space and are assumed mutually independent unless explicitly stated otherwise, which simplifies the joint distribution and facilitates tensor-product or sparse-grid quadrature;

2. the marginal distribution of each Z_i is chosen from a standard parametric family (e.g., uniform, Gaussian, Beta) that is orthogonal to a known polynomial basis, enabling direct application of generalized polynomial chaos expansions.

These assumptions imply certain limitations. Independence neglects any dependence structure between uncertain parameters, potentially underestimating or misrepresenting joint variability when correlations are significant. Restricting marginals to standard distributions simplifies the numerical framework but may introduce modeling error if the true input distributions are complex, multimodal, or heavy-tailed. In practice, such discrepancies can be mitigated by transforming empirical distributions into standard forms (e.g., via the Rosenblatt or Nataf transformation), but this step is not pursued in the present work.

3.3 Forward uncertainty propagation methods

Once input uncertainty has been represented via random variables or fields, the next step is to propagate this uncertainty through the model and quantify its impact on output QoIs. Mathematically, this involves estimating statistical descriptors, such as mean, variance, or probability distributions—of the model output $\mathcal{M}(\boldsymbol{Z})$ that is defined implicitly by numerical simulations.

Since the model response $\mathcal{M}(\boldsymbol{Z})$ is generally not available in closed form, various numerical methods have been developed to approximate its statistics efficiently. These methods differ in accuracy, computational cost, and suitability for high-dimensional problems.

We group the approaches into two main categories:

- **Sampling-based methods**: These estimate statistics by repeatedly evaluating the deterministic model at different input realizations. They are non-intrusive, embarrassingly parallel, and straightforward to implement. Typical examples include MCS and QMC.

- **Spectral methods based on gPC**: These approximate the model output as a polynomial function of the inputs, with coefficients determined either non–intrusive (SC) and intrusive (SG) method.

The following subsections provide a concise overview of these methods, including practical considerations for applying them in LBM fluid simulations.

3.3.1 Monte Carlo sampling

MCS is a fundamental and broadly applicable method for forward uncertainty quantification. It approximates statistical properties of a model response by evaluating the deterministic solver at independently sampled realizations of the uncertain input.

Assuming the notation from Section 3.1, let $\mathcal{M}(\boldsymbol{Z}) \in L^2(\Xi; \mathbb{R}^m)$, i.e., it has finite second moments. Let $\{\boldsymbol{Z}^{(i)}\}_{i=1}^{N_q}$ be independent and identically distributed (i.i.d.) samples drawn from the joint distribution of $\boldsymbol{Z}$. The Monte Carlo estimators for the mean and covariance are given by:

$$\mathbb{E}[\mathcal{M}(\boldsymbol{Z})] \approx \hat{\mu}_{N_q} := \frac{1}{N_q} \sum_{i=1}^{N_q} \mathcal{M}(\boldsymbol{Z}^{(i)}), \tag{3.1}$$

$$\mathrm{Cov}[\mathcal{M}(\boldsymbol{Z})] \approx \hat{\Sigma}_{N_q} := \frac{1}{N_q - 1} \sum_{i=1}^{N_q} \left(\mathcal{M}(\boldsymbol{Z}^{(i)}) - \hat{\mu}_{N_q}\right) \left(\mathcal{M}(\boldsymbol{Z}^{(i)}) - \hat{\mu}_{N_q}\right)^\top. \tag{3.2}$$

These estimators are unbiased and converge almost surely to the true moments as $N_q \to \infty$, by the strong law of large numbers:

$$\hat{\mu}_{N_q} \xrightarrow{a.s.} \mathbb{E}[\mathcal{M}(\boldsymbol{Z})], \qquad \hat{\Sigma}_{N_q} \xrightarrow{a.s.} \mathrm{Cov}[\mathcal{M}(\boldsymbol{Z})].$$

Moreover, by the multivariate Central Limit Theorem, the Monte Carlo mean estimator satisfies

$$\sqrt{N_q}\big(\hat{\mu}_{N_q} - \mathbb{E}[\mathcal{M}(\boldsymbol{Z})]\big) \xrightarrow{d} \mathcal{N}(0, \Sigma),$$

where $\Sigma = \mathrm{Cov}[\mathcal{M}(\boldsymbol{Z})] \in \mathbb{R}^{m \times m}$. As a consequence, the root-mean-square error (RMSE) of the estimator converges as

$$\|\hat{\mu}_{N_q} - \mathbb{E}[\mathcal{M}(\boldsymbol{Z})]\|_{L^2(\Xi)} = \mathcal{O}({N_q}^{-1/2}).$$

This convergence rate is independent of the input dimension d_Z or the regularity of the response function $\mathcal{M}$, making MCS particularly attractive for high-dimensional or non-smooth problems. Furthermore, it is trivially parallelizable, as each model evaluation $\mathcal{M}(\boldsymbol{Z}^{(i)})$ is independent of the others, which is especially advantageous in high-performance computing (HPC) environments such as those used for LBM-based simulations.

The main drawback of MCS is its relatively slow convergence, which may require a large number of samples when each model evaluation is computationally expensive. For this reason, it is often employed as a reference baseline to assess the efficiency of more advanced approaches, such as spectral or surrogate-based methods.

3.3.2 Quasi Monte Carlo

QMC methods enhance classical MC by replacing random samples with *deterministic low-discrepancy sequences*, which more uniformly cover the integration domain and reduce integration error for smooth functions.

Using the notation from Section 3.1, the goal is to approximate

$$\mathbb{E}[\mathcal{M}(\boldsymbol{Z})] = \int_{\mathbb{R}^{d_Z}} \mathcal{M}(\boldsymbol{z})\, \pi_{\boldsymbol{Z}}(\boldsymbol{z})\, d\boldsymbol{z}.$$

QMC maps low-discrepancy points $\{\boldsymbol{u}^{(i)}\}_{i=1}^{N_q} \subset [0,1]^{d_Z}$ (e.g., Sobol', Halton) to the target distribution via a transformation $T : [0,1]^{d_Z} \to \mathbb{R}^{d_Z}$, such as inverse cumulative distribution functions. The QMC estimator is

$$\hat{\mu}_{N_q}^{\text{QMC}} := \frac{1}{N_q} \sum_{i=1}^{N_q} \mathcal{M}(T(\boldsymbol{u}^{(i)})). \tag{3.3}$$

While QMC estimators are generally biased, they converge faster than MCS under regularity assumptions. For $\mathcal{M} \in \mathcal{V}_{\text{HK}}([0,1]^{d_Z})$ (bounded Hardy–Krause variation), the Koksma–Hlawka inequality gives

$$\left| \hat{\mu}_{N_q}^{\text{QMC}} - \mathbb{E}[\mathcal{M}(\boldsymbol{Z})] \right| \leq D^*(\{\boldsymbol{u}^{(i)}\}) \cdot V_{\text{HK}}(\mathcal{M}),$$

where D^* is the star discrepancy and V_{HK} the variation of $\mathcal{M}$. For smooth integrands, QMC achieves convergence rates of

$$\mathcal{O}\left(\frac{(\log N_q)^{d_Z}}{N_q} \right),$$

which outperform the $\mathcal{O}({N_q}^{-1/2})$ rate of MCS.

QMC remains non-intrusive and easily parallelizable. However, its efficiency degrades in high dimensions or for irregular responses. Remedies include:

- **Randomized QMC** (e.g., scrambled Sobol') for variance estimation,
- **Dimension reduction** via sensitivity reordering or active subspaces,
- **Smoothness-enhancing transformations** of input space.

In this work, QMC is applied to benchmark LBM simulations with moderate-dimensional smooth uncertainties, offering a more efficient alternative to standard MCS.

3.3.3 Spectral methods: generalized polynomial chaos

gPC provides a spectral framework for representing square-integrable random variables or model outputs in terms of orthogonal polynomials with respect to the input distribution. It generalizes Wiener's original construction [85] for Gaussian inputs to a broader class of probability measures.

For any model response $\mathcal{M}(\boldsymbol{Z}) \in L^2(\Xi; \mathbb{R}^m)$ as defined in Section 3.1, the gPC expansion reads:

$$\mathcal{M}(\boldsymbol{Z}) = \sum_{\boldsymbol{\alpha} \in \mathbb{N}_0^{d_Z}} \hat{\mathcal{M}}_{\boldsymbol{\alpha}} \, \Phi_{\boldsymbol{\alpha}}(\boldsymbol{Z}), \tag{3.4}$$

where:

- $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_{d_Z})$ is a multi-index,
- $\Phi_{\boldsymbol{\alpha}}(\boldsymbol{Z}) = \prod_{i=1}^{d_Z} \Phi_{\alpha_i}^{(i)}(Z_i)$ is a multivariate orthonormal polynomial,
- $\hat{\mathcal{M}}_{\boldsymbol{\alpha}} = \mathbb{E}[\mathcal{M}(\boldsymbol{Z}) \Phi_{\boldsymbol{\alpha}}(\boldsymbol{Z})]$ are the projection coefficients.

The univariate polynomials $\Phi_{\alpha_i}^{(i)}$ are orthonormal with respect to the marginal distribution of Z_i, chosen from the Askey scheme (see Table 3.1).

Table 3.1: Common distributions and their associated orthogonal polynomials in gPC.

Distribution	Support	Orthogonal Polynomial
Gaussian ($\mathcal{N}(0,1)$)	$(-\infty, \infty)$	Hermite
Uniform ($\mathcal{U}[-1,1]$)	$[-1,1]$	Legendre
Beta (Beta(a,b))	$[0,1]$	Jacobi
Gamma ($\Gamma(k,\theta)$)	$[0,\infty)$	Laguerre
Exponential ($\lambda e^{-\lambda z}$)	$[0,\infty)$	Laguerre
Poisson (Pois(λ))	$\{0,1,2,\ldots\}$	Charlier
Binomial (Bin(n,p))	$\{0,1,\ldots,n\}$	Krawtchouk
Negative Binomial	$\{0,1,\ldots\}$	Meixner

The basis functions satisfy:

$$\mathbb{E}[\Phi_{\boldsymbol{\alpha}}(\boldsymbol{Z}) \Phi_{\boldsymbol{\beta}}(\boldsymbol{Z})] = \delta_{\boldsymbol{\alpha},\boldsymbol{\beta}},$$

ensuring orthonormality and enabling the coefficient projection formula above.

If $\mathcal{M} \in L^2(\Omega)$, the expansion converges in the L^2-norm, and convergence is spectral (i.e., exponential) if $\mathcal{M}$ is analytic in $\boldsymbol{Z}$. In practice, the infinite series is truncated. A common strategy is the *total-degree truncation*, in which only multi-indices with total degree

$$|\boldsymbol{\alpha}| := \sum_{i=1}^{d_Z} \alpha_i \leq P$$

are retained. Here, $P \in \mathbb{N}_0$ denotes the *total polynomial order* of the expansion. The number of resulting basis functions is

$$N = \binom{d_Z + P}{P}, \tag{3.5}$$

which grows combinatorially with the stochastic dimension d_Z and the total order P.

For general models, exact computation of $\hat{g}_{\boldsymbol{\alpha}}$ is infeasible. Instead, we use non-intrusive spectral projection via numerical quadrature:

$$\hat{\mathcal{M}}_{\boldsymbol{\alpha}} \approx \sum_{j=1}^{N_q} a^{(j)}\, \mathcal{M}(\boldsymbol{Z}^{(j)})\, \Phi_{\boldsymbol{\alpha}}(\boldsymbol{Z}^{(j)}), \tag{3.6}$$

where $\{\boldsymbol{Z}^{(j)}, w^{(j)}\}_{j=1}^{N_q}$ are quadrature nodes and weights.

This method requires only sample evaluations of $\mathcal{M}(\boldsymbol{Z})$ and thus is non-intrusive, making it well-suited for designing UQ frameworks that couple to deterministic simulation codes.

In low dimensions, tensor-product Gauss quadrature is efficient:

$$N_q = n^{d_Z},$$

where n is the number of nodes per dimension. However, this scaling is exponential in d_Z, a manifestation of the *curse of dimensionality*.

To overcome this, sparse grid quadrature is introduced in Section 3.3.3.1, offering substantial computational savings while retaining accuracy.

3.3.3.1 Sparse grid quadrature

Sparse grid quadrature is an efficient numerical integration technique for approximating high-dimensional expectations, such as those arising in the non-intrusive spectral projection of gPC coefficients. Its key advantage is the mitigation of the exponential growth in quadrature nodes that occurs in tensor-product rules.

To compute the projection coefficient

$$\hat{\mathcal{M}}_{\boldsymbol{\alpha}} = \mathbb{E}[\mathcal{M}(\boldsymbol{Z})\Phi_{\boldsymbol{\alpha}}(\boldsymbol{Z})], \tag{3.7}$$

we approximate the expectation using a weighted sum of model evaluations:

$$\hat{\mathcal{M}}_{\boldsymbol{\alpha}} \approx \sum_{j=1}^{N_q} a^{(j)}\, \mathcal{M}(\boldsymbol{Z}^{(j)})\, \Phi_{\boldsymbol{\alpha}}(\boldsymbol{Z}^{(j)}), \tag{3.8}$$

where $\{\boldsymbol{Z}^{(j)}, w^{(j)}\}_{j=1}^{N_q}$ are the quadrature nodes and weights.

In a full tensor grid, the number of nodes scales as $N_q = n^{d_Z}$. This growth quickly becomes prohibitive for even moderate d_Z. Sparse grids address this challenge using the Smolyak algorithm [74], which constructs a sparse tensor-product quadrature of level $l \geq d_Z$ from one-dimensional quadrature rules $Q_\ell^{(i)}$ of level ℓ in each dimension i.

We define the difference operator

$$\Delta_\ell^{(i)} := Q_\ell^{(i)} - Q_{\ell-1}^{(i)}, \qquad Q_0^{(i)} := 0, \tag{3.9}$$

and express the Smolyak quadrature operator as

$$\mathcal{A}_l^{(d_Z)} = \sum_{\substack{|\boldsymbol{\ell}|=l \\ \boldsymbol{\ell}\in\mathbb{N}^{d_Z}}}^{l+d_Z-1} \left(\bigotimes_{i=1}^{d_Z} \Delta_{\ell_i}^{(i)} \right), \qquad |\boldsymbol{\ell}| := \sum_{i=1}^{d_Z} \ell_i. \tag{3.10}$$

This construction prioritizes low-dimensional interactions while controlling the overall resolution level l. The node count scales as

$$N_q = \mathcal{O}(2^l \cdot l^{d_Z - 1}), \tag{3.11}$$

which grows sub-exponentially in d_Z and thus offers a significant improvement over full tensor grids.

The choice of one-dimensional rules $Q_\ell^{(i)}$ influences both the accuracy and efficiency of the method. Gauss-type rules (e.g., Gauss–Legendre or Gauss–Hermite) are well suited to match input marginals, whereas nested rules (e.g., Clenshaw–Curtis or Genz–Keister) allow for node reuse across levels and therefore reduce redundant model evaluations.

In this work, we employ the Smolyak sparse grid construction with Genz–Keister quadrature rules for Gaussian random variables and Clenshaw–Curtis rules for uniform random variables. This choice ensures maximal reuse of nodes across levels and is particularly beneficial for our high-resolution LBM simulations, where each model evaluation is computationally expensive.

For sufficiently smooth functions $\mathcal{M} \in C^k([0,1]^{d_Z})$, the interpolation error satisfies

$$\|\mathcal{M} - \mathcal{A}_l^{(d_Z)}[\mathcal{M}]\|_\infty = \mathcal{O}(N^{-k} \log^{(k+1)(d_Z-1)} N_q), \tag{3.12}$$

indicating near-spectral accuracy with polynomial cost in N_q.

In the gPC framework, sparse grid quadrature enables the simultaneous projection of all coefficients $\hat{\mathcal{M}}_{\boldsymbol{\alpha}}$ with a greatly reduced computational burden. The integration of this method into our non-intrusive UQ framework significantly improves efficiency in the moderate-dimensional parameter spaces considered in our numerical experiments.

3.3.3.2 Stochastic collocation generalized polynomial chaos

Using the notation from Section 3.3.3, the truncated gPC expansion of the model response is

$$\mathcal{M}(\boldsymbol{Z}) \approx \sum_{|\boldsymbol{\alpha}| \leq P} \hat{\mathcal{M}}_{\boldsymbol{\alpha}} \, \Phi_{\boldsymbol{\alpha}}(\boldsymbol{Z}),$$

where P is the total polynomial order. The projection coefficients are approximated via numerical quadrature:

$$\hat{\mathcal{M}}_{\boldsymbol{\alpha}} \approx \sum_{j=1}^{N_q} w^{(j)} \, \mathcal{M}(\boldsymbol{Z}^{(j)}) \, \Phi_{\boldsymbol{\alpha}}(\boldsymbol{Z}^{(j)}),$$

with N_q quadrature nodes $\{\boldsymbol{Z}^{(j)}, w^{(j)}\}$ obtained from tensor-product or sparse-grid rules (cf. Section 3.3.3.1).

Each model evaluation $\mathcal{M}(\boldsymbol{Z}^{(j)}) = \mathcal{S}(\boldsymbol{Z}^{(j)})$ is obtained by calling the deterministic solver $\mathcal{S}$ at input $\boldsymbol{Z}^{(j)}$. Since these evaluations are independent, the method is embarrassingly parallel and scales well on high-performance computing architectures.

Once all coefficients $\hat{\mathcal{M}}_{\boldsymbol{\alpha}}$ are computed, statistical moments follow directly, e.g.:

$$\mathbb{E}[\mathcal{M}] = \hat{\mathcal{M}}_0, \tag{3.13}$$

$$\mathrm{Var}[\mathcal{M}] = \sum_{\boldsymbol{\alpha} \neq 0} \hat{\mathcal{M}}_{\boldsymbol{\alpha}}^2. \tag{3.14}$$

The resulting gPC surrogate serves as an efficient approximation of the full model. For any new input $\boldsymbol{Z}$, the surrogate output is evaluated by:

$$\mathcal{M}^{\mathrm{gPC}}(\boldsymbol{Z}) = \sum_{|\boldsymbol{\alpha}| \leq p} \hat{\mathcal{M}}_{\boldsymbol{\alpha}} \, \Phi_{\boldsymbol{\alpha}}(\boldsymbol{Z}),$$

requiring only polynomial evaluations. This enables fast uncertainty propagation, sensitivity analysis, and real-time inference.

Compared to MCS, SC-gPC achieves faster convergence—typically exponential in the total polynomial order P when $\mathcal{M}$ is analytic in $\boldsymbol{Z}$ whereas MCS converges at $\mathcal{O}({N_q}^{-1/2})$ regardless of regularity. This spectral convergence makes SC-gPC highly efficient for smooth problems with moderate stochastic dimensionality.

However, the curse of dimensionality limits the scalability of SC-gPC: the number of required samples grows rapidly with both d_Z and P. In contrast, MCS remains dimension-independent and is preferred in high-dimensional or non-smooth settings.

In this work, SC-gPC is adopted for cases where the model response is smooth and the stochastic dimension is moderate ($d_Z \lesssim 10$), as it combines high accuracy with non-intrusiveness and efficient parallel scalability. Its use enables the construction of surrogate models that support rapid post-processing tasks such as sensitivity analysis and real-time evaluations.

3.3.3.3 Stochastic Galerkin method

The SG method provides an intrusive approach to solving stochastic PDEs by projecting the governing equations onto a gPC basis. Unlike non-intrusive methods, SG requires modifying the numerical solver to operate directly on the system of coupled gPC modes.

Using the gPC representation from Section 3.3.3, the truncated expansion of the model response with total polynomial order P is

$$\mathcal{M}(\boldsymbol{Z}) \approx \sum_{|\boldsymbol{\alpha}| \leq P} \hat{\mathcal{M}}_{\boldsymbol{\alpha}} \, \Phi_{\boldsymbol{\alpha}}(\boldsymbol{Z}),$$

where the coefficients $\hat{\mathcal{M}}_{\boldsymbol{\alpha}}$ are computed via numerical quadrature.

Substituting this expansion into the stochastic PDE $\mathcal{L}[\mathcal{M}] = 0$ and applying Galerkin projection yields a deterministic coupled system:

$$\left\langle \mathcal{L}\left(\sum_{\boldsymbol{\alpha}} \hat{\mathcal{M}}_{\boldsymbol{\alpha}} \Phi_{\boldsymbol{\alpha}}\right), \Phi_{\boldsymbol{\beta}} \right\rangle = 0, \quad \forall \boldsymbol{\beta}, \tag{3.15}$$

where $\langle \cdot, \cdot \rangle$ denotes the $L^2(\Omega)$ inner product. The result is a deterministic PDE system for the $N = \binom{d_Z + p}{p}$ coefficient fields $\{\hat{\mathcal{M}}_{\boldsymbol{\alpha}}\}$, with couplings induced by products of gPC basis functions.

In practice, the Galerkin projection leads to the evaluation of expected residuals:

$$\mathbb{E}\left[\mathcal{L}(\mathcal{M}_{\mathrm{gPC}}) \, \Phi_{\boldsymbol{\beta}}\right], \tag{3.16}$$

which reduce to tensor contractions over basis products. These integrals can be computed analytically for polynomial data or precomputed offline and stored for repeated use.

While the SG method yields complete statistical information (e.g., mean, variance, higher moments) from a single simulation, it introduces several challenges:

- **Intrusiveness**: The solver must be modified to operate directly on the coupled gPC system.
- **Computational complexity**: The number of coupled equations grows linearly with the spatial degrees of freedom but combinatorially with d_Z and P, often producing dense coupling structures.
- **Stability concerns**: For nonlinear systems, hyperbolicity and conservation may be lost; stabilization techniques such as modal filtering or artificial dissipation may be required.

Despite these challenges, SG methods are highly effective for problems with smooth stochastic dependence and low-to-moderate input dimensions, where they can deliver spectral accuracy in the stochastic space at a significantly reduced cost compared to repeated deterministic runs.

In this work, we extend the SG methodology to the LBM framework (Chapter 4), deriving the Galerkin-projected collision and streaming operators, and implementing modal filtering to maintain numerical stability. This intrusive SG-LBM formulation enables high fidelity, uncertainty-aware flow simulations with full statistical output in a single solver execution.

4 Stochastic Galerkin Lattice Boltzmann Method

This chapter introduces the intrusive UQ approach developed in this dissertation, namely the SG LBM. Starting from the theoretical foundations of gPC expansions, which provide a systematic representation of stochastic model responses, we integrate this expansion directly into the LBE. The Galerkin projection then yields a coupled system for the polynomial expansion coefficients, enabling efficient propagation of parametric uncertainty while preserving the collision streaming structure of the solver. In contrast to non-intrusive methods, this approach reformulates the governing equations themselves and achieves significant gains in accuracy and efficiency. The methodology and results presented in this chapter have been published in [98].

This chapter is organized as follows. Section 4.1 derives the gPC expansion of the LBE and discusses the treatment of the collision operator, equilibrium distributions, and consistency analysis via CE expansion. Section 4.2 extends classical boundary conditions to the SG setting. Section 4.3 outlines implementational details of the proposed SG LBM algorithm. Finally, Section 4.4 presents numerical examples, including TGV flow, LDC flow, and isentropic vortex convection (IVC) flow, before conclusions are drawn in Section 4.5.

4.1 Generalized polynomial expansion of the lattice Boltzmann equation

The gPC framework introduced in Chapter 3 and Section 3.3.3 provides a systematic approach to represent the stochastic model response using orthogonal polynomial bases. Building on these foundations, we now apply the gPC-based SG approach to the LBE.

Let the model response $\mathcal{M}$ depend on the random vector $\boldsymbol{Z}$, and consider a truncated series expansion in terms of orthogonal basis functions Φ_α:

$$\mathcal{M}(\boldsymbol{Z}) \approx \mathcal{M}^N(\boldsymbol{Z}) = \sum_{\alpha=0}^{N} \hat{\mathcal{M}}_\alpha \Phi_\alpha(\boldsymbol{Z}). \tag{4.1}$$

The computation of expansion coefficients $\hat{\mathcal{M}}_\alpha$ and the evaluation of statistical quantities such as the mean $\bar{\mathcal{M}}$ and the variance $\sigma^2(\mathcal{M})$ follow the standard projection approach. These procedures are described in detail in Section 3.3.3.

Considering the LBE as a stochastic model, we form the stochastic evolution equation

$$f_i\left(\boldsymbol{x}+\boldsymbol{c}_i\triangle t, t+\triangle t, \boldsymbol{Z}\right) - f_i\left(\boldsymbol{x}, t, \boldsymbol{Z}\right) = \Omega_i\left(\boldsymbol{f}\left(\boldsymbol{x}, t, \boldsymbol{Z}\right)\right). \tag{4.2}$$

For all i, we form the gPC expansion of the particle distribution f_i with degree N, so that

$$f_i\left(\boldsymbol{x}, t, \boldsymbol{Z}\right) \approx f_i^N\left(\boldsymbol{x}, t, \boldsymbol{Z}\right) = \sum_{\alpha=0}^{N} \hat{f}_{i\alpha}\left(\boldsymbol{x}, t\right)\Phi_\alpha\left(\boldsymbol{Z}\right). \tag{4.3}$$

Notably, a gPC approximation of the distribution function f_i could lead to the loss of the positivity of the reconstructed f_i^N, which might cause instability in the gPC system. In some specific configurations of the numerical tests presented in Section 4.4, non-positivity of populations indeed occurs. From another perspective, hyperbolicity is based on non-negative physical quantities. The LBM does not inherently preserve the hyperbolic nature of the underlying equations, so the non-negativity problem in our proposed system is negligible. To still enforce the positivity of physical quantities, existing filtering strategies could be used [42, 52, 87].

The stochastic LBM can be expressed in terms of the non-projected gPC expansion via

$$\text{collision:} \quad f_i^{N,\star}\left(\boldsymbol{x}, t, \boldsymbol{Z}\right) = f_i^N\left(\boldsymbol{x}, t, \boldsymbol{Z}\right) - \Omega_i^N\left(\boldsymbol{f}^N\left(\boldsymbol{x}, t, \boldsymbol{Z}\right)\right), \tag{4.4}$$

$$\text{streaming:} \quad f_i^N\left(\boldsymbol{x}+\boldsymbol{c}_i\triangle t, t+\triangle t, \boldsymbol{Z}\right) = f_i^{N,\star}\left(\boldsymbol{x}, t, \boldsymbol{Z}\right). \tag{4.5}$$

By performing the standard SG projection on Eq. (4.4) and Eq. (4.5), we obtain

$$\text{collision:} \quad \hat{f}_{i\alpha}^{\star}\left(\boldsymbol{x}, t\right) = \hat{f}_{i\alpha}\left(\boldsymbol{x}, t\right) + \hat{\Omega}_{i\alpha}\left(\boldsymbol{f}^N\left(\boldsymbol{x}, t, \boldsymbol{Z}\right)\right), \tag{4.6}$$

$$\text{streaming:} \quad \hat{f}_{i\alpha}\left(\boldsymbol{x}+\boldsymbol{c}_i\triangle t, t+\triangle t\right) = \hat{f}_{i\alpha}^{\star}\left(\boldsymbol{x}, t\right), \tag{4.7}$$

respectively. Note that in the collision step Eq. (4.6), the expansion coefficients $\hat{f}_{i\alpha}^n$ are updated by adding the contribution from the collision term $\hat{\boldsymbol{\Omega}}_\alpha(\boldsymbol{f}^{n,N})$. The streaming step Eq. (4.7) only propagates the post-collision expansion coefficients $\hat{\boldsymbol{f}}_\alpha^{n,\star}$. In summary, Eq. (4.6) and Eq. (4.7) describe the Galerkin projection of the LBM using the gPC expansion, allowing for the representation and manipulation of the stochastic variables within the simulation.

4.1.1 Collision term in generalized polynomial chaos

Regarding the collision term Eq. (2.6), the relaxation time τ and thus the corresponding relaxation frequency $\omega = 1/\tau$ is determined by the kinematic viscosity ν via

$$\nu = c_s^2 \left(\tau - \frac{1}{2} \right). \tag{4.8}$$

In the present setting, the following two distinct cases are to be considered:

1. Deterministic relaxation time: If we assume that the collision frequency is constant, using Galerkin projection the coefficient of the collision term can be easily calculated as

$$\hat{\Omega}_{i\alpha} \left(\hat{\boldsymbol{f}}_\alpha \left(\boldsymbol{x}, t \right) \right) = -\omega \left(\hat{f}_{i\alpha} \left(\boldsymbol{x}, t \right) - \hat{f}_{i\alpha}^{\mathrm{eq}} \left(\boldsymbol{x}, t \right) \right). \tag{4.9}$$

2. Stochastic relaxation time: If we consider the kinematic viscosity ν as stochastic, the collision frequency is stochastic, too, and thus should be decomposed with the same polynomial basis and order, i.e.

$$\omega^N \left(\boldsymbol{Z} \right) = \sum_{\alpha=0}^{N} \hat{\omega}_\alpha \Phi_\alpha \left(\boldsymbol{Z} \right), \tag{4.10}$$

where $\hat{\omega}_\alpha$ are the expansion coefficients. In this case, the collision term is expanded in a different form, namely

$$\Omega_i^N \left(\boldsymbol{f}^N \left(\boldsymbol{x}, t, \boldsymbol{Z} \right) \right) = \omega^N \left(\boldsymbol{Z} \right) \left(f_i^{\mathrm{eq},N} \left(\boldsymbol{x}, t, \boldsymbol{Z} \right) - f_i^N \left(\boldsymbol{x}, t, \boldsymbol{Z} \right) \right). \tag{4.11}$$

Using SG projection once again

$$\begin{aligned} \hat{\Omega}_{i\alpha} \left(\hat{\boldsymbol{f}}_\alpha \right) = \frac{1}{\gamma_\alpha} \Bigg(& \sum_{j=0}^{N} \sum_{k=0}^{N} \hat{\omega}_j \hat{f}_{i\alpha}^{\mathrm{eq}} \mathbb{E} \left[\Phi_j \Phi_k \Phi_\alpha \right] \\ & - \sum_{j=0}^{N} \sum_{k=0}^{N} \hat{\omega}_j \hat{f}_{i\alpha} \mathbb{E} \left[\Phi_j \Phi_k \Phi_\alpha \right] \Bigg). \end{aligned} \tag{4.12}$$

In Eq. (4.12), the collision term coefficients $\hat{\boldsymbol{\Omega}}_\alpha(\hat{\boldsymbol{f}}_\alpha)$ are calculated based on the expansion coefficients $\hat{\omega}_j$ and $\hat{f}_{i\alpha}$, as well as the quadratures of the products of the orthogonal basis functions Φ_j, Φ_k, and Φ_α. This formulation allows for the treatment of stochastic kinematic viscosity and provides a novel way to incorporate the stochastic nature of the collision term in the LBM. Note that the methodology is similar to the one proposed in [87] for the gas kinetic scheme.

4.1.2 Moments and equilibrium populations in generalized polynomial chaos

In the deterministic LBE Eq. (2.10), obtaining the equilibrium distribution function is straight-forward using Eq. (2.15). However, in the stochastic system, direct multiplication and division cannot be applied to the stochastic moments. Therefore, we propose a method to obtain stochastic moments based on the existing stochastic distribution function and lattice velocity. The moments are defined as

$$\rho^N(\boldsymbol{Z}) = \sum_{i=0}^{q}\sum_{\alpha=0}^{N} \hat{f}_{i\alpha}\Phi_\alpha(\boldsymbol{Z}), \tag{4.13}$$

$$(\rho\boldsymbol{u})^N(\boldsymbol{Z}) = \sum_{i=0}^{q}\sum_{\alpha=0}^{N} \hat{f}_{i\alpha}\boldsymbol{c}_i\Phi_\alpha(\boldsymbol{Z}). \tag{4.14}$$

By calculating the moments on quadrature points Q_j, for $j = 0, 1, \ldots, N_q$ we can directly obtain the velocity on those quadrature points via

$$\boldsymbol{u}^N(Q_j) = \frac{(\rho\boldsymbol{u})^N(Q_j)}{\rho^N(Q_j)}. \tag{4.15}$$

To calculate the equilibrium populations on the quadrature points Q_j, we use the velocity and density on the same quadrature points, so that

$$f_i^{\mathrm{eq},N}(\boldsymbol{x},t,Q_j) = w_i\rho^N(Q_j)\Bigg(1 + \frac{\boldsymbol{u}^N(Q_j)\cdot\boldsymbol{c}_i}{c_s^2} + \frac{\left(\boldsymbol{u}^N(Q_j)\cdot\boldsymbol{c}_i\right)^2}{2c_s^4} - \frac{\boldsymbol{u}^N(Q_j)\cdot\boldsymbol{u}^N(Q_j)}{2c_s^2}\Bigg). \tag{4.16}$$

Next, we decompose the equilibrium distribution into a gPC expansion

$$f_i^{\mathrm{eq},N}(\boldsymbol{x},t,\boldsymbol{Z}) = \sum_{\alpha=0}^{N} \hat{f}_{i\alpha}^{\mathrm{eq}}\Phi_\alpha(\boldsymbol{Z}) \tag{4.17}$$

The coefficients of the equilibrium distribution function are calculated using a quadrature rule

$$\hat{f}_{i\alpha}^{\mathrm{eq}} = \frac{\mathbb{E}\left[f_i^{\mathrm{eq},N}(\boldsymbol{x},t,\boldsymbol{Z})\,\Phi_\alpha(\boldsymbol{Z})\right]}{\mathbb{E}\left[\Phi_\alpha^2(\boldsymbol{Z})\right]}, \tag{4.18}$$

for all $i = 0, 1, \ldots q-1$. When combined, the above expressions Eq. (4.13), Eq. (4.14), Eq. (4.15), Eq. (4.16), Eq. (4.17), and Eq. (4.18) provide a clear description of the moments and the computation of the equilibrium distribution function within the gPC expansion.

4.1.3 Chapman–Enskog expansion analysis

We use the CE expansion to provide a formal consistency analysis of the proposed SG LBM in terms of orders of magnitude $\mathcal{O}(\cdot)$ obtained in a multi-scale expansion. To the knowledge of the authors, this is the first CE expansion analysis of an SG LBM. For simplicity, we assume deterministic relaxation according to Eq. (4.8) and nondimensionalize the relaxation time. Since LBE asymptotically approaches the incompressible NSE, we directly make the incompressibility assumption.

With that, we start with the gPC expanded SG LBE

$$\hat{f}_{i\alpha}\left(\boldsymbol{x}+\boldsymbol{c}_i\triangle t, t+\triangle t\right) = \hat{f}_{i\alpha}\left(\boldsymbol{x}, t\right) - \triangle t\omega\left(\hat{f}_{i\alpha}\left(\boldsymbol{x}, t\right) - \hat{f}^{\mathrm{eq}}_{i\alpha}\left(\boldsymbol{x}, t\right)\right). \tag{4.19}$$

The expansion coefficients of the density distribution function, the temporal and the spatial derivatives are expanded as

$$\hat{f}_{i\alpha} = \sum_{k=0}^{\infty} \varepsilon^k \hat{f}^{(k)}_{i\alpha}, \tag{4.20}$$

$$\partial_t = \varepsilon\partial_t^{(1)} + \varepsilon^2\partial_t^{(2)}, \tag{4.21}$$

$$\boldsymbol{\nabla} = \varepsilon\boldsymbol{\nabla}^{(1)}, \tag{4.22}$$

respectively, where $\partial_\cdot = \partial/(\partial\cdot)$ denotes the partial derivative with respect to $\cdot$, and $\varepsilon > 0$ is an expansion parameter which indicates the terms of order $\mathcal{O}(Kn)$ with the Knudsen number denoted as Kn. Using a Taylor expansion, the SG LBE Eq. (4.19) becomes

$$\triangle t\left(\partial_t + \boldsymbol{c}_i\cdot\boldsymbol{\nabla}\right)\hat{f}_{i\alpha} + \frac{\triangle t^2}{2}\left(\partial_t + \boldsymbol{c}_i\cdot\boldsymbol{\nabla}\right)^2\hat{f}_{i\alpha} + \triangle t\omega\left(\hat{f}_{i\alpha} - \hat{f}^{\mathrm{eq}}_{i\alpha}\right) = \mathcal{O}\left(\delta t^3\right). \tag{4.23}$$

Neglecting terms of $\mathcal{O}(\triangle t^3)$ and higher, and substituting the multi-scale expansions Eq. (4.20), Eq. (4.21), and Eq. (4.22) into Eq. (4.23), we can split the equation in terms of orders of magnitudes $\mathcal{O}(\varepsilon^k)$. For $k = 0, 1, 2$, we obtain

$$\mathcal{O}\left(\varepsilon^0\right): \quad \hat{f}^{(0)}_{i\alpha} - \hat{f}^{\mathrm{eq}}_{i\alpha} = 0, \tag{4.24}$$

$$\mathcal{O}\left(\varepsilon^1\right): \quad \left(\partial_t^{(1)} + \boldsymbol{c}_i\cdot\boldsymbol{\nabla}^{(1)}\right)\hat{f}^{(0)}_{i\alpha} = -\omega\hat{f}^{(1)}_{i\alpha}, \tag{4.25}$$

$$\mathcal{O}\left(\varepsilon^2\right): \quad \partial_t^{(2)}\hat{f}^{(0)}_{i\alpha} + \left(1 - \frac{1}{2}\triangle t\omega\right)\left(\partial_t^{(1)} + \boldsymbol{c}_i\cdot\boldsymbol{\nabla}^{(1)}\right)\hat{f}^{(1)}_{i\alpha} = -\omega\hat{f}^{(2)}_{i\alpha}, \tag{4.26}$$

respectively. Obviously, from Eq. (4.24), we have that $\hat{f}_{i\alpha}^{(0)} = \hat{f}_{i\alpha}^{\mathrm{eq}}$. To proceed with the CE analysis, we form a gPC expansion of the stochastic density and the stochastic momentum density via

$$\rho^N(\boldsymbol{x}, t, \boldsymbol{Z}) = \sum_{\alpha=0}^{N} \hat{\rho}_\alpha(\boldsymbol{x}, t)\, \Phi_\alpha(\boldsymbol{Z}), \tag{4.27}$$

$$(\rho \boldsymbol{u})^N(\boldsymbol{x}, t, \boldsymbol{Z}) = \sum_{\alpha=0}^{N} (\rho \hat{\boldsymbol{u}})_\alpha(\boldsymbol{x}, t)\, \Phi_\alpha(\boldsymbol{Z}), \tag{4.28}$$

respectively, where

$$\hat{\rho}_\alpha(\boldsymbol{x}, t) = \sum_{i=0}^{q-1} \hat{f}_{i\alpha}(\boldsymbol{x}, t), \tag{4.29}$$

$$(\rho \hat{\boldsymbol{u}})_\alpha(\boldsymbol{x}, t) = \sum_{i=0}^{q-1} \boldsymbol{c}_i \hat{f}_{i\alpha}(\boldsymbol{x}, t). \tag{4.30}$$

To restore the stochastic macroscopic equation, the equilibrium distribution should meet the requirements of the following equations:

$$\sum_i \hat{f}_{i\alpha}^{eq} = \hat{\rho}_\alpha, \tag{4.31}$$

$$\sum_i \boldsymbol{c}_i \hat{f}_{i\alpha}^{eq} = \rho \hat{\boldsymbol{u}}_\alpha. \tag{4.32}$$

Here, we make an assumption that the density is asymptotically constant in incompressible fluids, which yields

$$\hat{\boldsymbol{u}}_\alpha = \frac{(\rho \hat{\boldsymbol{u}})_\alpha}{\hat{\rho}_\alpha}. \tag{4.33}$$

Simultaneously, with equations Eq. (4.29), Eq. (4.30), Eq. (4.31), and Eq. (4.32), we can easily obtain

$$\sum_n f_{i\alpha}^{(n)} = 0, \quad \forall n \neq 0, \tag{4.34}$$

$$\sum_n \boldsymbol{c}_i f_{i\alpha}^{(n)} = 0, \quad \forall n \neq 0. \tag{4.35}$$

Thus, the zeroth and first order moment summation Eq. (4.25) are

$$\partial_t^{(1)} \hat{\rho} + \boldsymbol{\nabla}^{(1)} \cdot (\rho \hat{\boldsymbol{u}}) = 0, \tag{4.36}$$

$$\partial_t^{(1)} \rho \hat{\boldsymbol{u}}_b + \boldsymbol{\nabla}^{(1)} \cdot (\rho \boldsymbol{u}_{\hat{a}} \boldsymbol{u}_b) = -\boldsymbol{\nabla}^{(1)} \hat{p}. \tag{4.37}$$

The zeroth and first order moment summation Eq. (4.26) are

$$\partial_t^{(2)} \hat{\rho} = 0, \tag{4.38}$$

$$\partial_t^{(2)} \rho \hat{\boldsymbol{u}}_b + \left(1 - \frac{\omega}{2}\right) \boldsymbol{\nabla}^{(1)} \cdot \left(\sum_i \boldsymbol{c}_{ia} \boldsymbol{c}_{ib} \hat{f}_{i\alpha}^{(1)} \right) = 0. \tag{4.39}$$

Finally, with the help of Eq. (4.25), we obtain

$$\sum_i \boldsymbol{c}_{ia} \boldsymbol{c}_{ib} \hat{f}_{i\alpha}^{(1)} = -\frac{1}{\omega} \Bigg[\partial_t^{(1)} \left((\rho \hat{\boldsymbol{u}_a} \boldsymbol{u}_b)_\alpha + \hat{\rho}_\alpha c_s^2 \delta_{ab} \right) + \boldsymbol{\nabla}^{(1)} \cdot \left(\sum_i \boldsymbol{c}_{ia} \boldsymbol{c}_{ib} \boldsymbol{c}_{ic} \hat{f}_{i\alpha}^{eq} \right) \Bigg] \tag{4.40}$$

$$\partial_t^{(1)} \left((\rho \hat{\boldsymbol{u}_a} \boldsymbol{u}_b)_\alpha + \hat{\rho}_\alpha c_s^2 \delta_{ab} \right) = \partial_t^{(1)} \left(\rho \hat{\boldsymbol{u}_a} \boldsymbol{u}_b \right)_\alpha + c_s^2 \boldsymbol{\nabla}^{(1)} \cdot (\rho \hat{\boldsymbol{u}}_\alpha) \delta_{ab}. \tag{4.41}$$

Note that the deterministic term $\partial_t^{(1)} \left(\rho \boldsymbol{u}_a \boldsymbol{u}_b \right)$ can be expanded as

$$\partial_t^{(1)} \left(\rho \boldsymbol{u}_a \boldsymbol{u}_b \right) = -c_s^2 \boldsymbol{u}_a \boldsymbol{\nabla} \cdot \rho - c_s^2 \boldsymbol{u}_b \nabla \cdot \rho - \boldsymbol{\nabla} \cdot (\rho \boldsymbol{u}_a \boldsymbol{u}_b \boldsymbol{u}_c). \tag{4.42}$$

In Eq. (4.42), under the incompressibility assumption, the density is asymptotically constant, allowing for the cancellation of the first two terms on the right. Moreover, the velocity in incompressible flow is significantly lower than the sound speed. By using the sound speed to make the last term dimensionless, it becomes a third-order small quantity, which can also be neglected. This implies that the term $\partial_t^{(1)} \left(\rho \boldsymbol{u}_a \boldsymbol{u}_b \right)$ can be disregarded under the incompressibility assumption. Therefore, its gPC coefficient, $\partial t^{(1)} (\rho \hat{\boldsymbol{u}_a} \boldsymbol{u}_b)_\alpha$, can be assumed to be negligible. Considering the latticed equilibrium distribution function, with a simple algebra we have that

$$\boldsymbol{\nabla}^{(1)} \cdot \left(\sum_i \boldsymbol{c}_{ia} \boldsymbol{c}_{ib} \boldsymbol{c}_{ic} \hat{f}_{i\alpha}^{eq} \right) = c_s^2 \boldsymbol{\nabla}^{(1)} \cdot (\hat{\rho \boldsymbol{u}}) \delta_{ab} + \hat{\rho} c_s^2 (\boldsymbol{\nabla} \cdot \hat{\boldsymbol{u}_a} + \boldsymbol{\nabla} \cdot \hat{\boldsymbol{u}_b}). \tag{4.43}$$

After sorting terms, we observe that

$$\partial_t \hat{\rho}_\alpha + \boldsymbol{\nabla} \cdot (\hat{\rho \boldsymbol{u}})_\alpha = 0, \tag{4.44}$$

$$\partial_t \left(\hat{\rho \boldsymbol{u}} \right)_\alpha + \boldsymbol{\nabla} \cdot \left((\hat{\rho \boldsymbol{u}})_\alpha \otimes \hat{\boldsymbol{u}}_\alpha \right) = -\boldsymbol{\nabla} \hat{p}_\alpha + \mu \boldsymbol{\nabla} \cdot \left[\boldsymbol{\nabla} \hat{\boldsymbol{u}}_\alpha + (\boldsymbol{\nabla} \hat{\boldsymbol{u}}_\alpha)^{\mathrm{T}} \right], \tag{4.45}$$

where the pressure expansion coefficient is defined as $\hat{p}_\alpha = c_s^2 \hat{\rho}_\alpha$. Again, assuming asymptotically incompressible flow, Eq. (4.44) becomes

$$\boldsymbol{\nabla} \cdot \hat{\boldsymbol{u}}_\alpha = 0. \tag{4.46}$$

Then, using the classical simplification steps for modeling deterministic incompressible fluid flow based on the weakly compressible NSE, e.g. $\boldsymbol{\nabla} \cdot (\boldsymbol{\nabla} \hat{\boldsymbol{u}}_\alpha)^{\mathrm{T}} = \boldsymbol{\nabla} (\boldsymbol{\nabla} \cdot \hat{\boldsymbol{u}}_\alpha) = 0$, can be used in Eq. (4.44) and Eq. (4.45). Based on that, it is observed that Eq. (4.44) and Eq. (4.45) structurally correspond to the stochastic weakly compressible NSE in terms of expansion coefficient variables $\hat{\rho}_\alpha$ and $(\hat{\rho \boldsymbol{u}})_\alpha$. Conclusively, we thus have formally proven the consistency of order two of the proposed SG LBM to stochastic density and density momentum moments Eq. (4.13) and Eq. (4.14), respectively.

4.2 Lattice Boltzmann boundary conditions in generalized polynomial chaos

4.2.1 No-slip wall boundary condition

The classical implementation of the mesoscopic bounce-back boundary condition for obtaining a macroscopic no-slip wall for the fluid velocity used in the present configuration. The standard streaming is replaced by switching the distribution function in the opposite velocity direction (see e.g. [78] and references therein). For our SG LBM, the expansion coefficients are switched instead of the distribution function. For example, in a specific setting in two dimensions, the coefficient for the incoming distribution function $\hat{f}^{\mathrm{eq}}_{2\alpha}$ is set equal to the coefficient of the outgoing distribution function $\hat{f}^{\mathrm{eq}}_{4\alpha}$.

$$\hat{f}_{2\alpha}\left(\mathbf{x}_b, t + \triangle t\right) = \hat{f}^{\star}_{4\alpha}\left(\mathbf{x}_b, t\right), \tag{4.47}$$

$$\hat{f}_{5\alpha}\left(\mathbf{x}_b, t + \triangle t\right) = \hat{f}^{\star}_{7\alpha}\left(\mathbf{x}_b, t\right), \tag{4.48}$$

$$\hat{f}_{6\alpha}\left(\mathbf{x}_b, t + \triangle t\right) = \hat{f}^{\star}_{8\alpha}\left(\mathbf{x}_b, t\right). \tag{4.49}$$

4.2.2 Moving wall boundary condition

For realizing macroscopic moving wall boundary conditions, we adopt the non-equilibrium extrapolation method. In the deterministic non-equilibrium extrapolation method [95], the following steps are performed at time step t_n, wall node $\boldsymbol{x}_{\mathrm{wall}}$, and incoming node $\boldsymbol{x}_{\mathrm{in}}$:

$$\rho\left(\boldsymbol{x}_{\mathrm{wall}}, t_n\right) = \rho\left(\boldsymbol{x}_{\mathrm{in}}, t_n\right), \tag{4.50}$$

$$\boldsymbol{u}\left(\boldsymbol{x}_{\mathrm{wall}}, t_n\right) = \boldsymbol{u}_{\mathrm{wall}}, \tag{4.51}$$

$$f_i\left(\boldsymbol{x}_{\mathrm{wall}}, t_n\right) = f^{\mathrm{eq}}_i\left(\boldsymbol{x}_{\mathrm{wall}}, t_n\right) + \left(f_i\left(\boldsymbol{x}_{\mathrm{in}}, t_n\right) - f^{\mathrm{eq}}_i\left(\boldsymbol{x}_{\mathrm{in}}, t_n\right)\right), \tag{4.52}$$

for all $i = 0, 1, \ldots q-1$, where $\boldsymbol{u}_{\mathrm{wall}}$ denotes the moving wall velocity. To account for the stochastic information, we apply the gPC expansion to Eq. (4.50), Eq. (4.51), and Eq. (4.52), respectively. The resulting stochastic non-equilibrium extrapolation method hence reads

$$\hat{\rho}_\alpha\left(\boldsymbol{x}_{\mathrm{wall}}, t\right) = \hat{\rho}_\alpha\left(\boldsymbol{x}_{\mathrm{in}}, t\right), \tag{4.53}$$

$$\hat{\boldsymbol{u}}_\alpha\left(\boldsymbol{x}_{\mathrm{wall}}, t\right) = \hat{\boldsymbol{u}}_{\mathrm{wall},\alpha}, \tag{4.54}$$

$$\hat{f}_{i\alpha}\left(\boldsymbol{x}_{\mathrm{wall}}, t\right) = \hat{f}^{\mathrm{eq}}_{i\alpha}\left(\boldsymbol{x}_{\mathrm{wall}}, t\right) + \left(\hat{f}_{i\alpha}\left(\boldsymbol{x}_{\mathrm{in}}, t\right) - \hat{f}^{\mathrm{eq}}_{i\alpha}\left(\boldsymbol{x}_{\mathrm{in}}, t\right)\right). \tag{4.55}$$

4.3 Implementational details

The deterministic LBE updates the distribution function $\boldsymbol{f}$ itself from time step t to $t+\triangle t$ (cf. Eq. (4.4) and Eq. (4.5)). However, in the here proposed SG LBM, we update the expansion coefficients $\hat{\boldsymbol{f}}_\alpha$ of the approximate stochastic distribution function $\boldsymbol{f}^N(\boldsymbol{Z})$ instead. The updating procedure is summarized as follows:

1. Calculate the expansion coefficients $\hat{f}^{\mathrm{eq}}_{i\alpha}$ using Eq. (4.18).
2. Compute the collision term using either Eq. (4.9) or Eq. (4.12), depending on whether the collision frequency is constant or stochastic.
3. Perform the collision and streaming by applying Eq. (4.6) and Eq. (4.7), respectively. This step involves combining the collision term with the current expansion coefficients to obtain the updated coefficients.
4. Apply the appropriate boundary conditions to ensure the desired behavior at the boundaries of the computational domain.
5. Update the moments of the distribution function using Eq. (4.13), Eq. (4.14) and Eq. (4.15). Based on these equations we calculate the stochastic moments, density $\rho^N(\boldsymbol{Z})$ and velocity $\boldsymbol{u}^N(\boldsymbol{Z})$ using the expansion coefficients $\hat{f}_{i\alpha}$.

4.4 Numerical examples

To evaluate the proposed SG LBM, we consider a series of benchmark problems of increasing complexity: the TGV flow with uncertain viscosity, its four–dimensional uncertain initial velocity, the LDC flow with uncertain boundary conditions, and the IVC problem with uncertain inflow.

4.4.1 Taylor–Green vortex flow with uncertain viscosity

We begin our numerical experiments with the two-dimensional decaying TGV flow, a classic benchmark problem for incompressible flow simulation [54]. The TGV flow is fully periodic and admits an analytical solution to the NSE, making it ideal for validating numerical methods by comparing against reference solutions. Based on this periodic test case we assess the accuracy and consistency. The TGV flow, given by

$$\boldsymbol{u}\left(x,y,t\right) = \begin{pmatrix} u\left(x,y,t\right) \\ v\left(x,y,t\right) \end{pmatrix} = \begin{pmatrix} -u_0 \cos\left(k_x x\right) \sin\left(k_y y\right) e^{-\frac{t}{t_d}} \\ u_0 \sin\left(k_x x\right) \cos\left(k_y y\right) e^{\frac{t}{t_d}} \end{pmatrix}, \tag{4.56}$$

$$p\left(x,y,t\right) = -\frac{1}{4} u_0^2 \left[\cos\left(2k_x x\right) + \left(\frac{k_x}{k_y}\right)^2 \cos\left(2k_y y\right) \right] e^{-\frac{2t}{t_d}} + P_0, \tag{4.57}$$

In these equations, $u_0 = 0.01$ is the initial velocity amplitude, k_x and k_y are wavenumbers corresponding to the domain length in the x and y directions, and t_d is a characteristic decay time that depends on the fluid viscosity.

We define a square domain $\Omega = [\,0, n_x\,] \times [\,0, n_y\,]$ in lattice units, where n_x and n_y denote the number of grid, we set the physical domain size to $L = 2\pi$. Hence, the wavenumbers become

$$k_x \;=\; \frac{2\pi}{n_x}, \quad k_y \;=\; \frac{2\pi}{n_y}. \tag{4.58}$$

The initial density is determined via the ideal equation of state $\rho = p/c_s^2$, with c_s the speed of sound.

We also fix a (nominal) shear viscosity ν via the chosen Reynolds number $\mathrm{Re} = 15$, so that

$$\nu \;=\; \frac{u_0\, L}{\mathrm{Re}}, \quad t_d \;=\; \frac{1}{\nu_0\left(k_x^2 + k_y^2\right)}, \tag{4.59}$$

and the relaxation time is set as

$$\tau \;=\; \frac{\nu}{c_s^2} + 0.5, \tag{4.60}$$

where ν can itself be a constant or—when modeling uncertainty—an uncertain parameter drawn from a specified distribution.

To investigate the impact of input uncertainty on flow evolution, we introduce *stochasticity* into the TGV setup by modeling the *kinematic viscosity* ν as an uncertain parameter. Specifically, we define

$$\nu = \zeta\nu_0, \quad \text{with} \quad \zeta \sim \mathcal{U}[0.8,\, 1.2], \tag{4.61}$$

where ν_0 is the nominal viscosity corresponding to the prescribed Reynolds number $\mathrm{Re} = 15$. This formulation reflects a $\pm 20\%$ variability around the reference viscosity.

All simulations in this work enforce periodic boundary conditions in both directions.

In this test case, the normalized total kinetic energy is computed as

$$K(t) = \frac{2}{|\Omega| u_0^2} \int_\Omega \left(u^2(x,y,t) + v^2(x,y,t)\right)\, \mathrm{d}x\mathrm{d}y, \tag{4.62}$$

where the integral is approximated with averaging over the domain constituted by grid nodes $(x, y) \in [0, n_x] \times [0, n_y]$.

Here, we also apply gPC expansion to the normalized total kinetic energy, which is computed as

$$\hat{K}_\alpha(t) = \frac{2}{|\Omega| u_0^2} \int_\Omega \left((\hat{u^2})_\alpha(x,y,t) + (\hat{v^2})_\alpha(x,y,t)\right)\, \mathrm{d}x\mathrm{d}y, \tag{4.63}$$

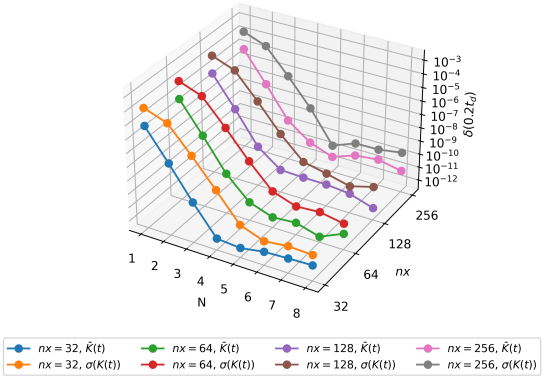
the gPC coefficient for normalized total kinetic energy $\hat{K}(t)$ is calculated based on the gPC coefficients for the squared velocities $(\hat{u^2})$ and $(\hat{v^2})$. These coefficients for the squared velocities are determined as follows:

$$(\hat{u^2})_\alpha = \sum_{j=0}^{N} \sum_{k=0}^{N} \hat{u}_j \hat{u}_\alpha E\left[\phi_j \phi_k \phi_\alpha\right] \tag{4.64}$$

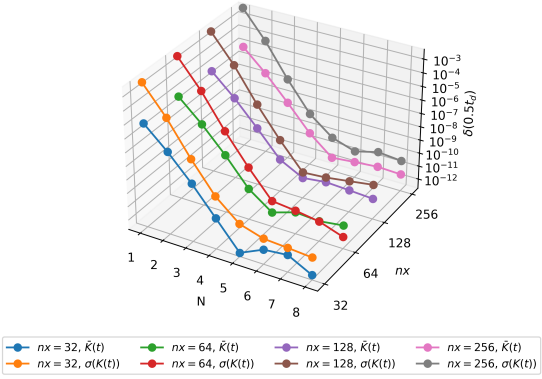
$$(\hat{v^2})_\alpha = \sum_{j=0}^{N} \sum_{k=0}^{N} \hat{v}_j \hat{v}_\alpha E\left[\phi_j \phi_k \phi_\alpha\right] \tag{4.65}$$

Across different resolutions, we compare the computed mean of normalized total kinetic energy with its analytical reference at two points in time $t = 0.2t_d$ and $t = 0.5t_d$, respectively.

Initially, we perform a stochastic consistency study for SG LBM on the two-dimensional TGV flow. This evaluation is based on the relative error, denoted as $\delta(t)$, of the mean and standard deviation of normalized total kinetic energy $K(t)$ concerning the results obtained at various points



(a) Relative error $(\delta(t))$ at $t = 0.2t_d$



(b) Relative error $(\delta(t))$ at $t = 0.5t_d$

Figure 4.1: Relative error $(\delta(t))$ of expectation value $(\bar{K}(t))$ and standard deviation $(\sigma(K(t)))$ of normalized total kinetic energy $K(t)$ for two-dimensional TGV flow computed with SG LBM with respect to highest polynomial order results at different points in time $t = 0.2t_d, 0.5t_d$. Several spatial resolutions $(n_x = 32, 64, 128, 256)$ and polynomial orders $(N = 1, 2, 3, \ldots, 8)$ are tested.

in time, namely, $t = 0.2t_d$ and $0.5t_d$. We investigate this across different spatial resolutions (here $n_x = 32, 64, 128, 256$) and polynomial orders $(N = 1, 2, 3, \ldots, 8)$. Hence, we compute

$$\delta(t) = \frac{\left|\bar{K}_N^{n_x}(t) - \bar{K}_9^{n_x}(t)\right|}{\left|\bar{K}_9^{n_x}(t)\right|}, \tag{4.66}$$

$$\delta(t) = \frac{\left|\sigma\left(K_N^{n_x}(t)\right) - \sigma\left(K_9^{n_x}(t)\right)\right|}{\left|\sigma\left(K_9^{n_x}(t)\right)\right|}, \tag{4.67}$$

respectively. The convergence results in terms of the obtained relative error over several resolutions and polynomial orders are presented in Figure 4.1 for two dedicated points in time. The results show that a polynomial order of $N = 5$ is sufficient for achieving the highest accuracy. Hence, all of the following tests use the polynomial order $N = 5$. From Figure 4.1, it can be observed that the relative errors converge exponentially to machine precision. This indicates that the here proposed SG LBM achieves spectral accuracy in the random space.

Moreover, we assess the spatial consistency of SG LBM by measuring the experimental order of convergence (EOC). This is done by comparing the results obtained at different resolutions with the analytical solution at the highest resolution. To validate the spatial consistency of our proposed method, we implemented it using a polynomial order of $N = 5$ and compute the following error

$$\delta = \frac{\left|\bar{K}_5^{n_x}(t) - K^{128}(t)\right|}{|K^{128}(t)|}, \tag{4.68}$$

Here, $\bar{K}_N^{n_x}(t)$ represents the total kinetic energy $K(t)$ using polynomial order N winth resolution nx.

The results of this EOC study are summarized in Table 4.1 and plotted in Figure 4.2. The linear regression fit validates a second order consistency in space.

Table 4.1: Spatial EOC results of SG LBM in terms of δ (see (4.68)).

Resolution	8	16	32	64	Order
δ	0.1512	0.04435	0.01104	0.002262	2.02

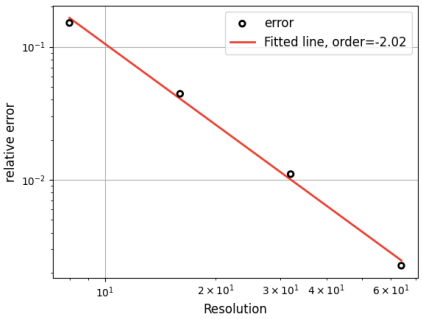
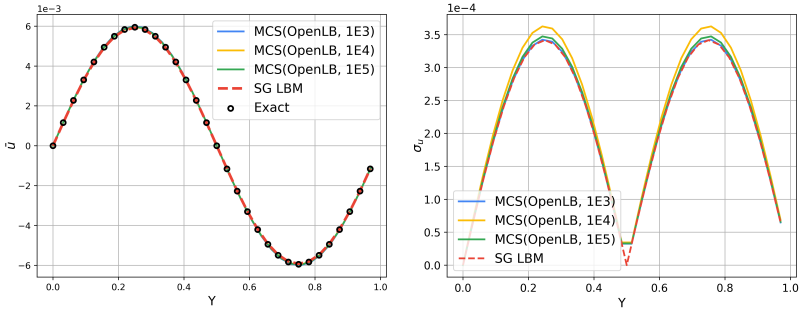


Figure 4.2: Spatial EOC results of SG LBM for TGV flow in terms of δ (see (4.68)).

Moreover, the EOC study shows that a grid size of 32×32 achieved a relative error of 1% compared to the highest resolution. Consequently, the computational domain was discretized into a grid size of $(n_x = n_y = 32)$.

For the purpose of comparison, we analyzed the expectation and standard deviation of the velocity component u in the x-direction at $t = 0.5t_d$ and compared it with the results obtained from the MCS for different sample sizes (1E2, 1E3, 1E4). These comparison results, illustrated in Figure 4.3, demonstrate the high accuracy achieved by our proposed method with small polynomial orders and a significantly improved convergence rate compared to MCS.



(a) Expectation **(b)** Standard deviation

Figure 4.3: Expectation values ($\bar{u}$) and standard deviations ($\sigma(u)$) of velocity in the x-direction along the central vertical line. Spatial resolution $nx = 32$ and polynomial order $N = 5$ are used.

We conducted a comparison of computational time costs among different numerical methods, and the results are presented in Table 4.2. This study was conducted using an 8-core, 16-thread Intel Core i7-10700KF processor with OpenMP support. For the comparison, we utilized SG LBM with polynomial orders of $N = 5$ and $N_q = 11$ to simulate the TGV flow with resolution $n_x = 32$. The number of quadrature points, $N_q = 2N + 1$, ensures that the gPC provides a robust approximation [28].

Additionally, we implemented MCS on the deterministic LBM part of our SG LBM code as well as on OpenLB. Both MCS employed 10,000 samples to ensure convergence in the TGV flow problem. The results demonstrate that MCS with OpenLB achieves a good speed-up rate (factor eleven) compared to the LBM code without any optimization. Moreover, our proposed SG LBM exhibits a significantly high speed-up rate (factor 334) compared to the MC LBM.

It is notable that the computational cost of SG LBM is influenced by the resolution, polynomial order, and quadrature points. While the resolution influence is similar to that of the standard LBM, the polynomial order and quadrature points are unique characteristics of SG LBM, and their specific impact remains unknown. Therefore, it is imperative to conduct an analysis of these factors. To ensure sufficient data for an efficiency study, we choose a spatial resolution of $n_x = 64$. Subsequently,

Table 4.2: Computational time costs of several numerical methods in seconds for TGV flow.

method	cpu time [s]
SG LBM (OpenMP)	1.02
MC LBM (OpenMP)	341.05
MC LBM (OpenLB, MPI)	240.395

we investigate the individual influence of polynomial order and quadrature points separately, and the results are presented in Figure 4.4 and Figure 4.5, respectively. The results demonstrate that both polynomial order and quadrature points exhibit a linear increase in computational cost.

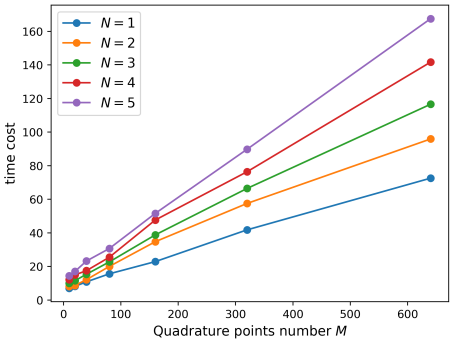


Figure 4.4: Influence of number of quadrature points N_q for different polynomial orders $N = 1, 2, 3, 4, 5$ on time cost of TGV flow simulation with SG LBM at spatial resolution $n_x = 64$.

In our analysis, we evaluate and compare the performance of the SG LBM against the MC LBM for simulating TGV flow at $t = 0.5t_d$ on an Intel Xeon Platinum 8368 CPU. This comparative study is conducted across various resolutions to understand the efficiency with respect to consistency of the here used methods in handling stochastic variables. In the implementation of the SG LBM, polynomial orders ranging from 1 to 8 are employed. For the MC LBM, 200 samples are used, with the number of samples denoted as n. The CPU time for 2, 10, and 100 samples is used for comparison. The results (efficiency with respect to consistency), as illustrated in the series of plots in Figure 4.6, reveal

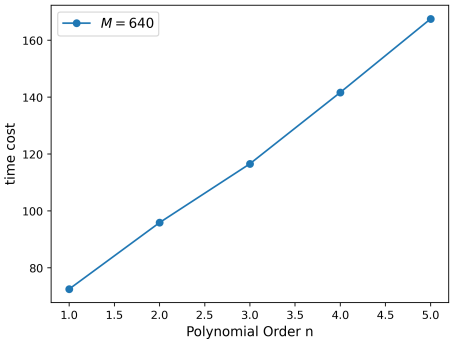


Figure 4.5: Influence of polynomial order (tested $n = 1, 2, 3, 4, 5$) with number of quadrature points $N_q = 640$ on time cost of TGV flow simulation with SG LBM at spatial resolution $n_x = 64$.

that the SG LBM exhibits strongly increased consistency orders and reduced computational demand, compared to the MC LBM. The relative error metric used in this comparison is defined as

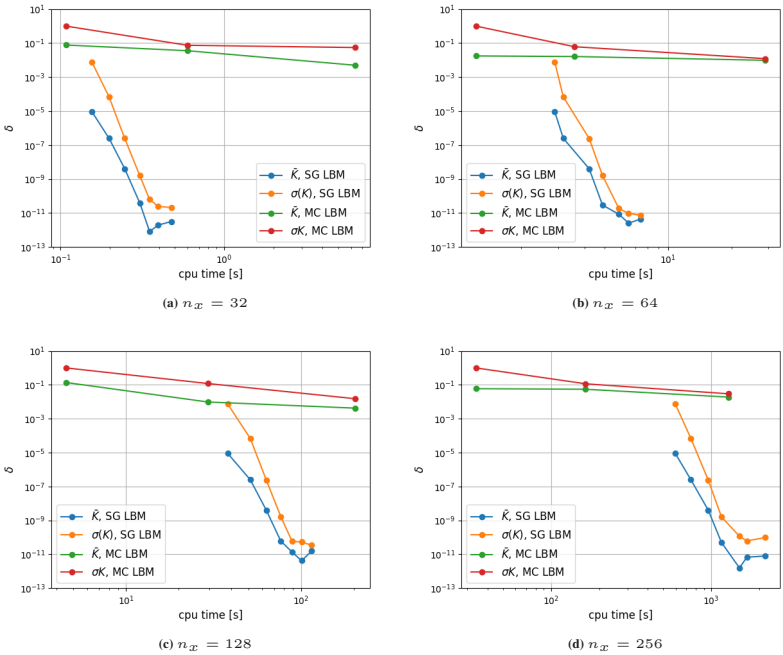
$$\delta_{\mathrm{SGLBM}} = \frac{\left|\bar{K}_N^{n_x} - \bar{K}_8^{n_x}\right|}{\left|K_8^{n_x}\right|}, \tag{4.69}$$

$$\delta_{\mathrm{MCLBM}} = \frac{\left|\bar{K}_n^{n_x} - \bar{K}_{200}^{n_x}\right|}{\left|\bar{K}_{200}^{n_x}\right|}. \tag{4.70}$$

To measure the efficiency with respect to accuracy of SG LBM compared to MC LBM, an experimental speedup analysis is conducted. For the MC LBM, the resolutions $n_x = 16, 32, 128, 256$ are used, with sample numbers of $N_q = 12, 25, 50, 100, 200$, respectively. For the SG LBM, polynomial orders $N = 5, 6, 7$ are used. Since exact statistical solutions for a given problem are generally unknown [3], we study accuracy with respect to the highest spatial and stochastic resolution MCS. The converged MC LBM results at resolution $n_x = 256$ are considered as the reference solution with a sample number of $N_q = 60000$. The relative error σ with respect to this reference solution of $\bar{K}$ has been computed as above. The speedup results according to

$$\mathrm{speedup}_{\mathrm{SG}} = \left(\frac{\delta_{\mathrm{MCLBM}}}{\delta_{\mathrm{SGLBM}}}\right)^{\frac{1}{2}} \tag{4.71}$$

are presented in Figure 4.7. On average, our novel SG LBM shows a conventional speedup factor of $\delta_{\mathrm{MCLBM}}/\delta_{\mathrm{SGLBM}} \approx 5.72$.



(a) $n_x = 32$ **(b)** $n_x = 64$

(c) $n_x = 128$ **(d)** $n_x = 256$

Figure 4.6: Comparative performance analysis of the SG LBM and MC LBM for the TGV flow at several spatial resolutions $n_x = 32, 64, 128, 256$. The error σ is measured in terms of (4.69) and (4.70). CPU time in seconds for MC LBM measured for $N_q = 2, 10, 100$ samples and for SG LBM for polynomial orders from 1 to 8, respectively.

4.4.2 Taylor–Green Vortex flow with four-dimensional uncertain initial velocity

The initial condition of the TGV flow with four-dimensional uncertainties is defined as [62, 65]

$$\boldsymbol{u}\left(x,y,0\right) = \begin{pmatrix} u\left(x,y,0\right) \\ v\left(x,y,0\right) \end{pmatrix} = \begin{pmatrix} -u_0^R \cos\left(k_x x\right) \sin\left(k_y y\right) \\ u_0^R \sin\left(k_x x\right) \cos\left(k_y y\right) \end{pmatrix}, \tag{4.72}$$

$$p\left(x,y,0\right) = -\frac{1}{4}(u_0^R)^2 \left[\cos\left(2k_x x\right) + \left(\frac{k_x}{k_y}\right)^2 \cos\left(2k_y y\right)\right] + P_0, \tag{4.73}$$

where all deterministic parameters are kept the same in the Section 4.4.1. The difference is that uncertainties are introduced into the velocity $u_0^R = u_0 + \epsilon_d(x)$, where the perturbation ϵ_d is given

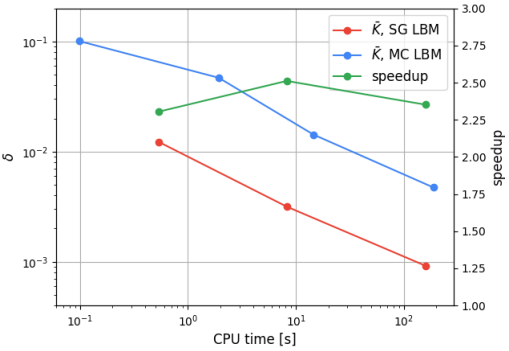


Figure 4.7: Speedup analysis of SG LBM (polynomial orders of $N = 5, 6, 7$) compared to MC LBM (resolutions are $n_x = 16, 32, 128, 256$ with corresponding sample numbers of $N_q = 12, 25, 50, 100, 200$) according to (4.71) measured in terms of to CPU time over accuracy with respect to MC LBM reference solution with resolution $n_x = 256$ and $N_q = 60000$ samples.

by the first-order harmonics with four-dimensional i.i.d. uniformly distributed random amplitude $\zeta_{d,i,j} \sim \mathcal{U}(-0.025, 0.025)$, i.e.

$$\epsilon_d(x, y) = \frac{1}{4} \sum_{(i,j)\in\{0,1\}^2} \zeta_{d,i,j} \alpha_i(4x)\alpha_j(4y), \tag{4.74}$$

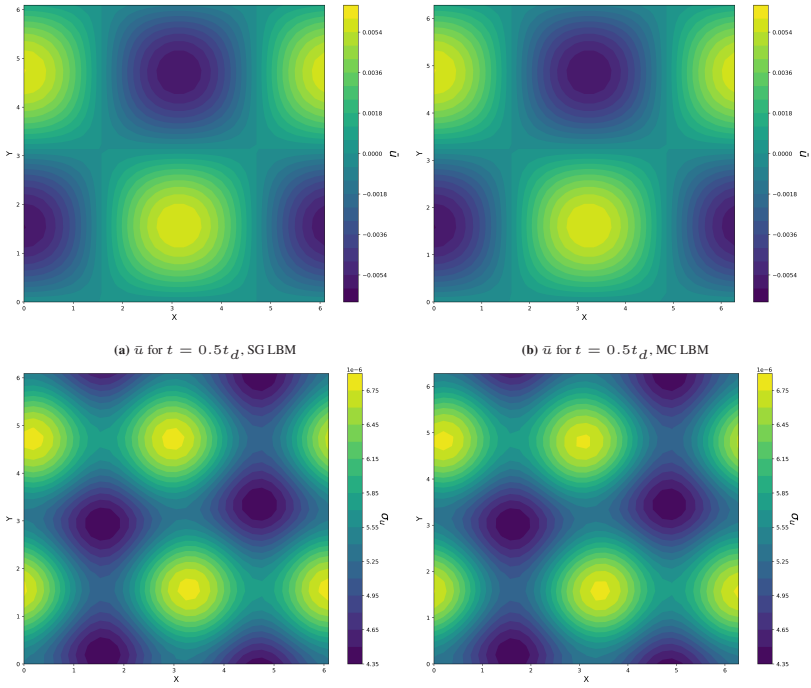
where

$$\alpha_i(x) = \begin{cases} \sin(x) & \text{if } i = 0, \\ \cos(x) & \text{if } i = 1. \end{cases} \tag{4.75}$$

The computational results are shown in Figure 4.8 and Figure 4.9, demonstrating that the velocity field obtained from the SG LBM is consistent with those from the MC LBM.

For the purpose of comparison, we analyze the expectation and standard deviation of the velocity component u in the x-direction at $t = 0.5t_d$ and compare it with the results obtained from the MC LBM for different sample sizes 1E3, 1E4, and 1E5. These comparison results, illustrated in Figure 4.10, demonstrate the higher accuracy achieved by our proposed method compared to MC LBM.

We also evaluate and compare the performance of the SG LBM and the MC LBM on an Intel Xeon Platinum 8368 CPU. In the implementation of the SG LBM, polynomial orders ranging from $N = 1$ to $N = 5$ are used. For the reference MC LBM, 10^6 samples are used. The CPU time



(a) $\bar{u}$ for $t = 0.5t_d$, SG LBM

(b) $\bar{u}$ for $t = 0.5t_d$, MC LBM

(c) σ_u for $t = 0.5t_d$, SG LBM

(d) σ_u for $t = 0.5t_d$, MC LBM

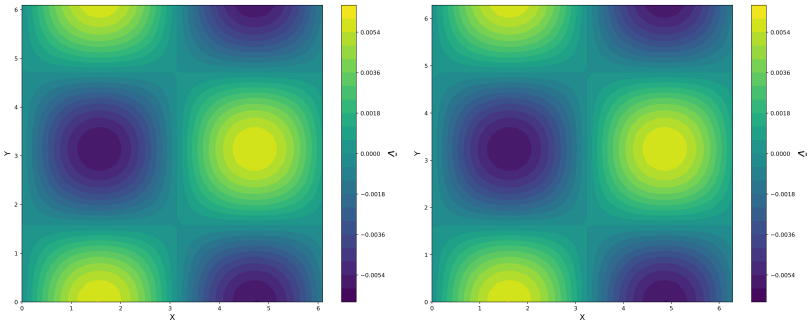
Figure 4.8: Expectation values ($\bar{u}$) and standard deviations ($\sigma(u)$) of velocity in the x-direction of TGV flow computed with SG LBM and MC LBM (sample numbers $N_q = 10000$). Spatial resolution $n_x = 33$, polynomial order $N = 3$, and number of quadrature points $N_q = 7$ are used.

for $N_q = 10, 10^2, 10^3$ samples in MC LBM is used for comparison. The relative error in this comparison is defined as follows:

$$\delta_{\mathrm{SGLBM}} = \frac{\left|\bar{K}_N^{32} - \bar{K}_8^{32}\right|}{\left|\bar{K}_5^{32}\right|}, \tag{4.76}$$

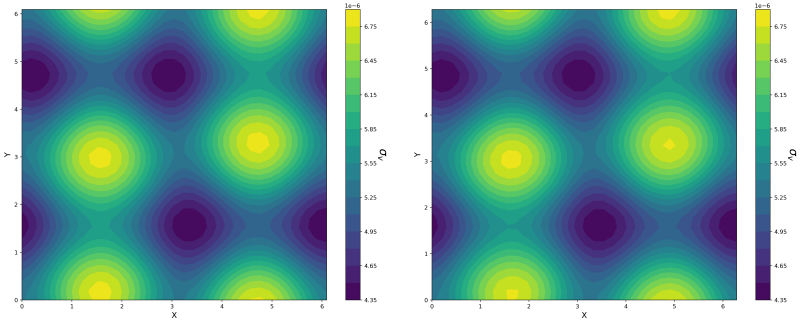
$$\delta_{\mathrm{MCLBM}} = \frac{\left|\bar{K}_n^{32} - \bar{K}_{100000}^{32}\right|}{\left|\bar{K}_{100000}^{32}\right|}. \tag{4.77}$$

The results are illustrated in Figure 4.11. It is found that both the expectation $\bar{K}$ and the standard deviation $\sigma(K)$ from the SG LBM converge very quickly, indicating that the TGV flow is not sensitive to this four-dimensional uncertainty in velocity. Although the computational cost of the SG LBM increases exponentially with the number of uncertainties, for the four-dimensional uncertainty case, it remains a superior choice for this test problem compared to the MC LBM.



(a) $\bar{v}$ for $t = 0.5t_d$, SG LBM

(b) $\bar{v}$ for $t = 0.5t_d$, MC LBM



(c) σ_v for $t = 0.5t_d$, SG LBM

(d) σ_v for $t = 0.5t_d$, MC LBM

Figure 4.9: Expectation values ($\bar{v}$) and standard deviations ($\sigma(v)$) of velocity in the y-direction of TGV flow computed with SG LBM and MC LBM (sample numbers $N_q = 10000$). Spatial resolution $nx = 33$, polynomial order $N = 3$, and number of quadrature points $N_q = 7$ are used.

4.4.3 Lid-driven cavity flow with uncertain lid-driven velocity

The classical incompressible LDC flow involves a square cavity filled with fluid, where the top wall is driven to move while the other three walls remain stationary. The velocity of the top wall is set to a constant value, denoted as $u_w = 1.0$, and the characteristic length is defined as $L = 1.0$. This case is characterized solely by the Reynolds number $Re = u_w L/\nu = 1000$, which governs the flow behavior. We have chosen a relaxation time of $\tau = 0.5384$ for our simulations.

To evaluate the effect of input variability, we introduce *parametric uncertainty* in the boundary condition. Specifically, the top-wall velocity is treated as a *random variable*:

$$u_w \sim \mathcal{U}[0.9,\ 1.1], \tag{4.78}$$

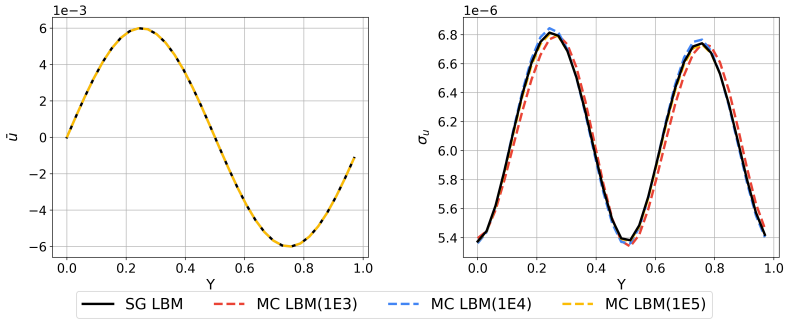
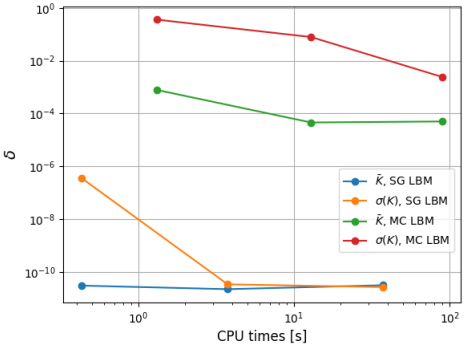


Figure 4.10: Expectation values $\bar{u}$ (left) and standard deviations $(\sigma(u))$ (right) of velocity in the x-direction along the central vertical line computed with MC LBM for sample sizes 1E3, 1E4, 1E5 and with SG LBM. The spatial resolution is $nx = 32$ and for SG LBM a polynomial order $N = 3$ and $N_q = 7$ quadrature points are used.



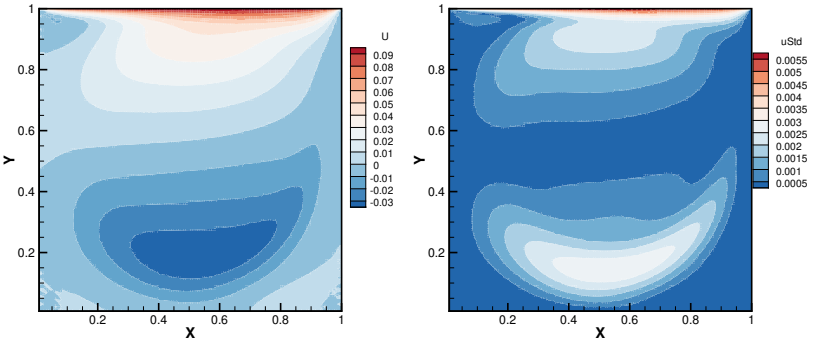
(a)

Figure 4.11: Comparative performance analysis of the SG LBM and MC LBM for the TGV flow with spatial resolution $n_x = 32$ and four-dimensional uncertainties. The error σ is measured in terms of (4.69) and (4.70). CPU time in seconds for MC LBM measured for $N_q = 10, 100, 1000$ samples and for SG LBM for polynomial orders from 1 to 4, respectively.

representing a $\pm 10\%$ perturbation around the nominal velocity. [86].

To capture the uncertainty in the system, we utilize a polynomial order of $N = 3$ and a set of quadrature points with $N_q = 7$.

The computational domain in our simulation is discretized into a grid $(x, y) \in [0, n_x] \times [0, n_y]$. The mean and standard deviation of velocity in the x-direction on a grid $n_x = n_y = 128$ are shown in Figure 4.12.

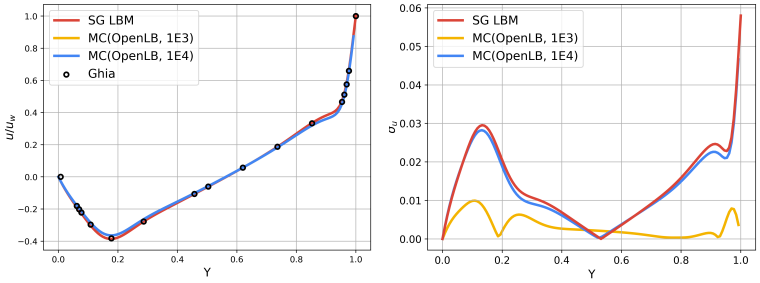


(a) Expectation values ($\bar{u}$)

(b) Standard deviation ($\sigma(u)$)

Figure 4.12: Expectation values ($\bar{u}$) and standard deviations ($\sigma(u)$) of velocity in the x-direction of LDC flow computed with SG LBM. Spatial resolution $n_x = 128$, polynomial order $N = 3$, and number of quadrature points $N_q = 7$ are used.

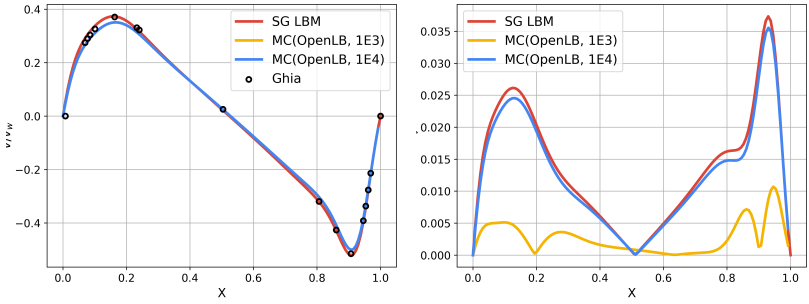
To enable a comprehensive comparison, we include benchmark solutions from Ghia *et al.* [21] and results obtained using the MC LBM. Figure 4.13 and Figure 4.14 illustrate the comparison of the expectation and standard deviation of the velocity in the x-direction along the central vertical line and the y-direction along the central horizontal line, respectively. The results demonstrate that the expectation values of SG LBM approximate the reference result from Ghia *et al.* [21]. In addition SG LBM exhibits a significantly faster convergence rate compared to the MC LBM (tested for sample sizes 1E3 and 1E4) as indicated by the standard deviations plotted in Figures 4.13 and 4.14.



(a) Expectation

(b) Standard deviation

Figure 4.13: Expectation values ($\bar{u}$) and standard deviations ($\sigma(u)$) of velocity in the x-direction along the central vertical line of LDC flow computed with SG LBM. Spatial resolution $n_x = 128$, polynomial order $N = 3$, and number of quadrature points $N_q = 7$ are used.



(a) Expectation **(b)** Standard deviation

Figure 4.14: Expectation values ($\bar{u}$) and standard deviations ($\sigma(u)$) of velocity in the y-direction along the central horizontal line of LDC flow computed with SG LBM. Spatial resolution $n_x = 128$, polynomial order $N = 3$, and number of quadrature points $N_q = 7$ are used.

We investigate the spatial consistency of SG LBM also for the LDC flow. The relative L^2-norm error is measured based on the velocity along the central line, i.e.

$$\delta_2 = \left\| \left(\frac{|\bar{\boldsymbol{u}}_3^{nx} - \bar{\boldsymbol{u}}_3^{512}|}{|\bar{\boldsymbol{u}}_3^{512}|} \right) \right\|_2, \tag{4.79}$$

and plotted in Figure 4.15. As expected, the results reveal a second-order convergence rate in space.

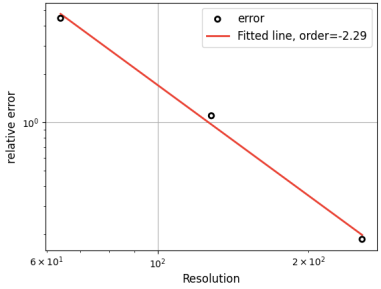
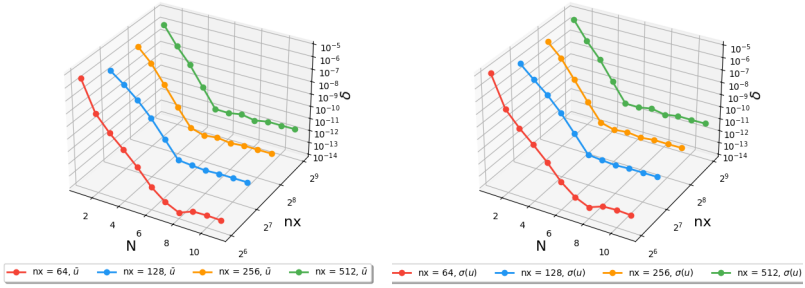


Figure 4.15: Spatial EOC results of SG LBM for LDC flow in terms of δ_2 (see (4.79)).

The stochastic consistency is also investigated for several resolutions based on δ (see Eq. (4.66) with respect to u-velocity). The results are shown in Figure 4.16, which shows that unlike the TGV flow, the LDC flow requires a higher polynomial order to converge. However, the spectral convergence rate

is also numerically approved for the LDC flow. In case of a polynomial order $N = 8$, the results reach machine precision.



(a) Expectation

(b) Standard deviation

Figure 4.16: Relative error (δ, see (4.66)) of expectation value ($\bar{u}$) and standard deviation ($\sigma(u)$) of u-velocity along the central vertical line of LDC flow computed with SG LBM. Several spatial resolutions ($n_x = 64, 128, 256, 512$) and polynomial orders ($N = 1, 2, 3, \ldots, 13$) are tested.

4.4.4 Isentropic vortex convection

As a final test case, we consider the IVC, which involves the advection of a smooth, inviscid vortex by a uniform background flow with free-stream Mach number $Ma = 0.042$, corresponding to a free-stream velocity $u_\infty = 0.05$.

The computational domain is defined as $[0,\ 10] \times [0,\ 10]$, discretized into a 100×100 uniform grid. This yields a characteristic length $L = 10$ and a spatial resolution $\triangle x = 0.1$. Periodic boundary conditions are imposed on all boundaries to ensure continuous convection [97].

To emulate nearly inviscid conditions, the viscosity is set to $\nu = 10^{-15}$, resulting in a very high Reynolds number $\mathrm{Re} = 5 \times 10^{14}$.

This test is designed to validate the ability of the SG LBM to accurately capture smooth vortical flow under parametric uncertainty. In particular, we introduce *uncertainty in the background flow velocity* by defining:

$$\rho_\infty = 1.0, \quad u_\infty = 0.05\,\zeta, \quad v_\infty = 0.0, \quad \zeta \sim \mathcal{U}[0.9,\ 1.1], \tag{4.80}$$

where ζ is a uniformly distributed random variable modeling $\pm 10\%$ variability in the free-stream speed. This represents potential fluctuations or control uncertainty in inflow conditions.

The isentropic vortex is embedded into this uncertain background flow through the following perturbations:

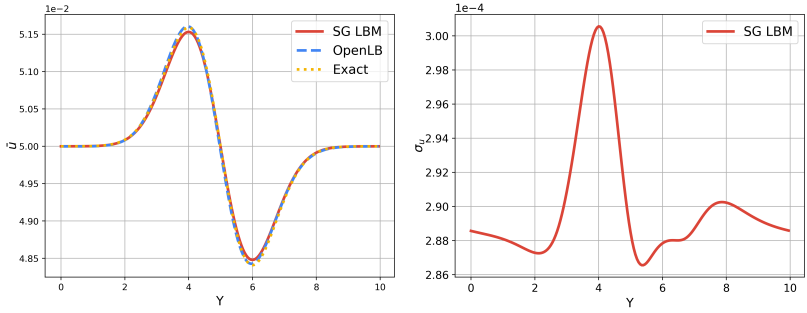
$$\rho(x,y) = \left[1 - \frac{(\gamma-1)b^2}{8\gamma\pi^2}\exp\left(1-r^2\right)\right]^{\frac{1}{\gamma-1}}, \tag{4.81}$$

$$u(x,y) = u_\infty - \frac{b}{2\pi}\exp\left(\frac{1}{2}(1-r^2)\right)(y-y_c), \tag{4.82}$$

$$v(x,y) = v_\infty + \frac{b}{2\pi}\exp\left(\frac{1}{2}(1-r^2)\right)(x-x_c), \tag{4.83}$$

where $b = 0.05$ denotes the vortex strength, and the distance from the vortex center is given by $r = \sqrt{(x-x_c)^2 + (y-y_c)^2}$ with $(x_c,\, y_c) = (5,\, 5)$.

To capture the uncertainty in the system, we utilize a polynomial order of $N = 4$ and a set of quadrature points with $N_q = 9$. The velocity profile is extracted from the flow field evolved for 20 flow-through-times Figure 4.17 shows the expectation values and standard deviation specifically at $t = 20\text{FFTs}$. The SG LBM results for approximating inviscid flows show good agreement to the exact prediction as well as to the MC LBM (OpenLB) results.



(a) Expectation

(b) Standard deviation

Figure 4.17: Expectation values ($\bar{u}$) and standard deviations ($\sigma(u)$) of velocity in the y-direction along the central horizontal line of IVC computed with SG LBM. Spatial resolution $n_x = 100$, polynomial order $N = 4$, and $N_q = 9$ quadrature points are tested. MC LBM (OpenLB) results and an exact prediction are plotted as a reference.

4.5 Conclusion

In conclusion, by decomposing the LBE into polynomials and employing the Galerkin projection, we have first implemented the Stochastic Galerkin method on the conventional lattice Boltzmann

equation. This work aims to address the need for a more efficient and accurate UQ method in CFD problems.

Our numerical results, obtained through simulations of TGV flow, LDC flow, and IVC flow, showcased the accuracy and computational efficiency of the SG LBM approach. The comparison with MC LBM as a benchmark demonstrated the superior convergence rate and smaller computational cost of SG LBM. Compared to MC LBM the novel SG LBM reaches a speedup factor of 5.72 on average in a randomized two-dimensional Taylor–Green vortex flow test case.

Overall, our study highlights the potential of the SG LBM as an intrusive UQ technique for UQ in CFD simulations. The combination of high efficiency, accuracy, and computational effectiveness makes SG LBM a valuable tool for addressing uncertainties arising from various sources in CFD.

5 OpenLB-UQ Framework

In this chapter, we present the OpenLB-UQ framework, a modular and extensible software extension for UQ in LBM simulations. Building upon the deterministic capabilities of OpenLB, the framework introduces a non-intrusive UQ layer that orchestrates sample generation, solver execution, and statistical postprocessing. It supports multiple sampling and collocation techniques, including MCS, QMC, and SC. This design enables efficient uncertainty propagation without requiring modifications to existing OpenLB applications. The framework, including its class hierarchy and runtime workflow, has been developed as part of this dissertation and is documented in [99]. Most of its core functionalities have already been released in OpenLB versions 1.8 [39] and 1.8.1 [40].

5.1 Software architecture

The software architecture of the proposed framework is designed to enable UQ on top of the existing deterministic LBM simulations provided by OpenLB. Instead of modifying the internal solver kernel, the UQ layer operates as a high-level orchestrator that coordinates the sampling process, invokes the existing LBM application for each realization, and subsequently aggregates statistical results.

As illustrated in Figure 5.1, the architecture consists of two loosely coupled layers:

- **UQ layer:** This layer is responsible for generating samples from uncertain inputs, managing the configuration of UQ methods, and performing statistical postprocessing. It includes components such as `Distribution`, `SamplingStrategy`, and `QuadratureRule`, all orchestrated by the manager class called `UncertaintyQuantification`. The UQ layer is agnostic to the underlying LBM discretization and solver details.
- **OpenLB layer:** This layer runs the deterministic simulation for each sample provided by the UQ layer. It utilizes OpenLB's existing modular structure, including `SuperLattice`, `BlockLattice`, and user-defined boundary/initial conditions. From the perspective of the LBM solver, each realization is simply a deterministic run with a specific set of input parameters.

Based on the above architecture, the overall execution follows a three-phase workflow:

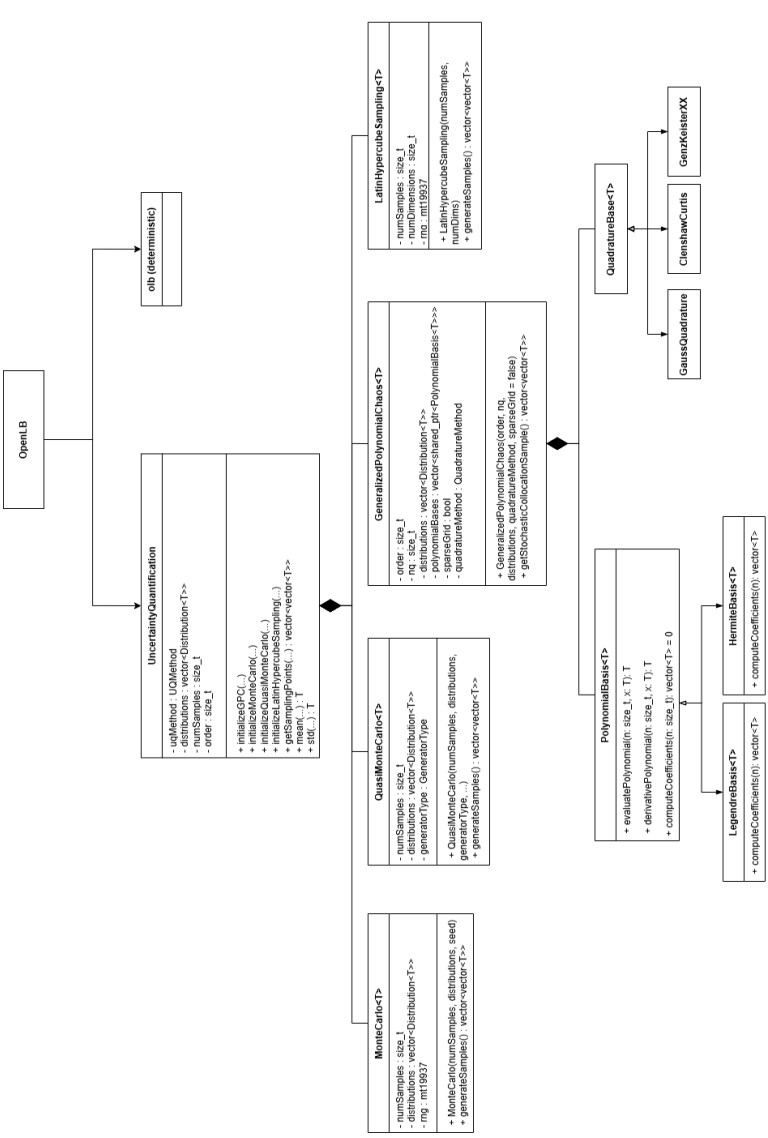


Figure 5.1: Class hierarchy for OpenLB-UQ framework.

1. **Sampling:** The UQ module generates a set of N_q samples $\{\boldsymbol{Z}^{(i)}\}_{i=1}^{N_q} \subset \mathbb{R}^{d_Z}$ from the prescribed input distribution using a chosen sampling or quadrature method (e.g., Monte Carlo, quasi-Monte Carlo, or gauss quadrature).

2. **Simulation:** Each sample $\boldsymbol{Z}^{(i)}$ is passed to the OpenLB application, which performs a deterministic simulation and returns a scalar quantity of interest $\mathcal{M}(\boldsymbol{Z}^{(i)})$, such as the drag coefficient or probe velocity.

3. **Postprocessing:** The UQ module collects the model outputs $\{\mathcal{M}(\boldsymbol{Z}^{(i)})\}_{i=1}^{N_q}$ and computes statistical quantities (e.g., mean and variance).

This design cleanly separates uncertainty propagation logic from the numerical solver implementation. As a result, all existing LBM applications in OpenLB (also beyond CFD) can be used without modification, and new UQ methods can be integrated into the framework by extending the modular interfaces of the UQ layer.

5.2 Core class design

The core of the OpenLB-UQ framework is organized as a collection of modular C++ classes, structured around abstract base classes and template-based specializations. This design enables flexibility in composing different UQ strategies and facilitates seamless integration of new methods.

Figure 5.1 presents the class hierarchy underlying the UQ layer. The framework is centered around the `UncertaintyQuantification` manager class, which coordinates all stages of the UQ process, including sampling, solver invocation, and postprocessing. Its functionality is composed of the following key components.

5.2.1 Distribution and polynomial basis

The `Distribution` class hierarchy defines input uncertainty models. It supports common univariate distributions such as uniform and Gaussian, as well as joint multivariate distributions constructed through product or correlated measures. Each distribution instance provides sampling functions and moments necessary for stochastic analysis.

To support spectral methods such as SC-gPC, each distribution is associated with an orthogonal polynomial basis through the `PolynomialBasis` interface. Concrete implementations like `LegendreBasis` and `HermiteBasis` correspond to uniform and Gaussian inputs, respectively. This automatic pairing guarantees consistency between sampling and projection in gPC expansions.

5.2.2 Sampling strategies

The `SamplingStrategy` interface defines a general contract for generating sample points in the stochastic space. It is implemented by various non-intrusive UQ methods, including:

- `MonteCarloSampling`
- `QuasiMonteCarloSampling`
- `LatinHypercubeSampling`
- `StochasticCollocation` (gPC)

Each strategy class produces a sample matrix $\{\boldsymbol{Z}^{(i)}\}_{i=1}^{N_q} \subseteq \mathbb{R}^{d_Z}$, which is passed to the deterministic solver.

For sampling-based methods, samples are drawn from the input distribution. For collocation-based methods, points are selected via tensor-product or sparse-grid quadrature rules.

5.2.3 Quadrature and collocation

The `QuadratureRule` component handles the numerical integration over the stochastic space. This base class, `QuadratureBase`, defines the interface for evaluating quadrature points and weights. Currently, Gauss–Quadrature is implemented and available in OpenLB release 1.8 and 1.8.1. Additional quadrature rules, including Genz–Keister and Clenshaw–Curtis, as well as support for sparse Smolyak grids [18, 19, 79], will be included in the next release. The selection of quadrature rules is managed through a type-safe enumerator `QuadratureMethod`, which ensures consistency in point generation and projection.

For SC, the class `GeneralizedPolynomialChaos` computes the gPC coefficients by projecting the QoIs onto the polynomial basis using a quadrature rule. Its primary functionality is to compute statistical moments, such as the mean and variance directly from the gPC coefficients, which are obtained from deterministic solver outputs evaluated at the quadrature points.

5.2.4 Solver orchestration and data management

The `UncertaintyQuantification` class acts as the top-level orchestrator. It holds instances of `SamplingStrategy`, `Distribution`, and `QuadratureRule`, and manages all simulation inputs and outputs. For each sample, it launches the solver, collects the quantity of interest, and updates statistical aggregates.

To interface with existing OpenLB applications, the UQ layer requires only a minimal wrapper that connects input samples to solver parameters and extracts output quantities. This lightweight coupling preserves the modularity of both the UQ and LBM components.

5.3 Runtime workflow

The execution of OpenLB-UQ follows a modular, three-phase runtime workflow: (i) sample generation, (ii) deterministic simulation, and (iii) statistical postprocessing. This structure enables a clear separation between the UQ logic and the numerical solver. Algorithm 1 provides a high-level overview of this process. At runtime, the `UncertaintyQuantification` manager orchestrates the following

Algorithm 1 Non-intrusive UQ algorithm in OpenLB-UQ.

Input: random input $\boldsymbol{Z} \sim h(\boldsymbol{Z})$, LBM solver $\mathcal{S}$, number of samples N_q
Output: mean $\mathbb{E}[\mathcal{M}]$ and standard deviation $\sigma(\mathcal{M})$ of the quantity of interest $\mathcal{M}$

Generate sample set $\{\boldsymbol{Z}^{(i)}\}_{i=1}^{N_q}$ from $h(\boldsymbol{Z})$
for $i = 1, \ldots, N_q$ **do**
    $\mathcal{M}^{(i)} \leftarrow \mathcal{S}(\boldsymbol{Z}^{(i)})$ ▷ Run LBM simulation with sample $\boldsymbol{Z}^{(i)}$
end for
if Monte Carlo sampling **then**
    $\mathbb{E}[\mathcal{M}] \leftarrow \frac{1}{N} \sum_{i=1}^{N} \mathcal{M}^{(i)}$ ▷ Eq. (3.1)
    $\sigma(\mathcal{M}) \leftarrow \sqrt{\frac{1}{N} \sum_{i=1}^{N} \left(\mathcal{M}^{(i)} - \mathbb{E}[\mathcal{M}]\right)^2}$ ▷ Eq. (3.2)
end if
if Stochastic collocation **then**
    Approximate: $\mathcal{M}(\boldsymbol{Z}) \approx \sum_{\alpha=0}^{N} \hat{\mathcal{M}}_\alpha \, \Phi_\alpha(\boldsymbol{Z})$ ▷ Eq. (3.4)
    Compute coefficients: $\hat{\mathcal{M}}_\alpha \approx \frac{1}{\gamma} \sum_{i=1}^{N_q} a_i \, \mathcal{M}^{(i)} \, \Phi_\alpha(\boldsymbol{Z}^{(i)})$ ▷ Eq. (3.6)
    $\mathbb{E}[\mathcal{M}] \leftarrow \hat{\mathcal{M}}_0$ ▷ Eq. (3.13)
    $\sigma(\mathcal{M}) \leftarrow \left(\sum_{\alpha=1}^{N} \hat{\mathcal{M}}_\alpha^2\right)^{1/2}$ ▷ Eq. (3.14)
end if

steps:

1. **Initialization:** The UQ configuration is defined directly within the C++ application code. This includes the choice of sampling strategy, distribution parameters, quadrature rule (if applicable), and the number of samples N_q. These parameters are passed to the `UncertaintyQuantification` manager.

2. **Sample generation:** A set of input samples $\{\boldsymbol{Z}^{(i)}\}_{i=1}^{N_q} \subset \mathbb{R}^{d_Z}$ is generated in the stochastic space using the selected `SamplingStrategy` instance. Depending on the method, samples

may come from random draws (e.g., Monte Carlo), low-discrepancy sequences (e.g., Sobol), or deterministic collocation points (e.g., Clenshaw–Curtis sparse grids).

3. **Deterministic simulation:** For each sample $\boldsymbol{Z}^{(i)}$, a corresponding deterministic simulation input is assembled and passed to the OpenLB solver. The solver executes a complete LBM run and computes a quantity of interest $\mathcal{M}(\boldsymbol{Z}^{(i)})$. The UQ layer treats each run as a black box and stores only the scalar output $\mathcal{M}^{(i)}$.

4. **Postprocessing:** The outputs $\{\mathcal{M}^{(i)}\}_{i=1}^{N_q}$ are aggregated after all simulations are completed. For sampling-based methods, statistical moments such as the mean and standard deviation are computed directly from the ensemble. For SC, the outputs are projected onto a polynomial chaos basis $\{\Phi_N\}$ using quadrature or regression to obtain the gPC coefficients $\{\hat{\mathcal{M}}_N\}$.

5. **Output:** Final results including statistical summaries and surrogate models (e.g., gPC expansions)—are written to disk. Optional postprocessing modules compute spatial statistics (e.g., pointwise mean and variance fields) and export them in `.vti` format for visualization in ParaView.

This workflow is inherently parallelizable. Currently, two levels of parallelism are supported: (i) *sample-level*, and (ii) *domain-level*. Sample-level parallelism distributes simulation runs across available cores or nodes, while domain-level parallelism is managed internally by OpenLB using MPI. This hybrid parallelization strategy enables the UQ framework to scale efficiently from desktop machines to high-performance clusters with minimal configuration changes.

The modular architecture and parallelization capabilities of the OpenLB-UQ framework enable efficient uncertainty propagation across a wide range of CFD applications. In the following chapter (Chapter 6), we validate the framework and assess its performance and scalability through numerical experiments and benchmark studies.

6 Non-Intrusive Uncertainty Quantification

In this chapter, we validate the OpenLB-UQ framework through a series of numerical experiments and performance evaluations. Benchmark cases, including flow past a circular cylinder and TGV flow are simulated under parametric uncertainties to assess the accuracy and convergence behavior of non-intrusive UQ methods such as MCS, QMC, and SC. We investigate the statistical consistency, spectral convergence, and sample efficiency of each method using reference solutions. Furthermore, we evaluate the computational performance of the framework on a modern supercomputing platform, comparing sample-level and domain-level parallelization strategies in terms of speedup and scalability. These results confirm the effectiveness, robustness, and parallel efficiency of our proposed OpenLB-UQ framework.

6.1 Numerical experiments

To demonstrate the effectiveness and applicability of the proposed OpenLB-UQ framework, we present results from benchmark simulations of a 2D flow past a circular cylinder as well as a 2D TGV flow, both with parametric uncertainties up to dimension four. Relative error metrics are used to determine the convergence behavior with respect to references of high polynomial order.

The errors are then compared for various numbers of samples to approve individual accuracy orders of MCS-, QMC- and SC LBM, respectively. Finally, we validate the convergence of our framework in terms of a Wasserstein metric when computing statistical solutions with four-dimensional uncertainty.

6.1.1 Flow past a circular cylinder with uncertain inlet velocity

We consider a standard benchmark of incompressible flow past a circular cylinder, extended to account for uncertainty in the inlet velocity. The inlet velocity is modeled as

$$u_{\text{in}}(\zeta) = 0.2 + 0.04\,\zeta, \qquad \zeta \sim \mathcal{U}(-1, 1),$$

which enters a parabolic inflow profile and induces variability in the Reynolds number and drag coefficient.

The domain is a two-dimensional channel of size $22D \times 4D$, with a cylinder of diameter D placed near the inlet and vertically centered. The inlet velocity profile is prescribed as

$$\boldsymbol{u}(0, y, t, \zeta) = \begin{pmatrix} 4\, u_{\text{in}}(\zeta)\, \dfrac{y(H-y)}{H^2} \\ 0 \end{pmatrix}, \quad y \in [0, H],\ t \geq 0, \quad H = 4D. \tag{6.1}$$

The Reynolds number is defined using the peak inlet velocity:

$$Re = \frac{u_{\text{in}} D}{\nu},$$

with a nominal value of $Re = 20$ corresponding to $u_{\text{in}} = 0.2$.

Boundary conditions are set as follows. At the inlet, an interpolated velocity boundary condition is applied [47]; at the outlet, an interpolated pressure condition ensures free outflow [47]. The cylinder surface is treated with a second-order accurate no-slip condition using the Bouzidi scheme [8], and the top and bottom walls are set to no-slip. The simulation is initialized with equilibrium populations, and the inflow is gradually ramped up to avoid transient oscillations.

The drag and lift forces are computed as surface integrals over the cylinder boundary S,

$$F_{\text{D}} = \int_S \left(-p\, n_1 + \rho\nu\, \frac{\partial v_t}{\partial \boldsymbol{n}}\, n_2 \right) \mathrm{d}S, \tag{6.2}$$

$$F_{\text{L}} = \int_S \left(-p\, n_2 - \rho\nu\, \frac{\partial v_t}{\partial \boldsymbol{n}}\, n_1 \right) \mathrm{d}S, \tag{6.3}$$

where $\boldsymbol{n} = (n_1, n_2)^{\text{T}}$ is the outward unit normal vector on S, and $v_t = \boldsymbol{u} \cdot \boldsymbol{s}$ is the tangential velocity, with the unit tangent vector defined as $\boldsymbol{s} = (s_1, s_2)^{\text{T}} = (n_2, -n_1)^{\text{T}}$. The derivative $\partial v_t / \partial n$ denotes the directional derivative along $\boldsymbol{n}$, i.e., $\boldsymbol{n} \cdot \nabla v_t$.

The drag and lift coefficients are defined as

$$C_{\text{D}} = \frac{2F_{\text{D}}}{\rho\, u_{\text{in}}^2\, D}, \tag{6.4}$$

$$C_{\text{L}} = \frac{2F_{\text{L}}}{\rho\, u_{\text{in}}^2\, D}, \tag{6.5}$$

Let $C_{\mathrm{D}}^{\mathrm{ref}}$ denote a reference value of the mean drag coefficient, obtained from a high-fidelity computation. Given an estimated mean $\bar{C}_{\mathrm{D}}$, the relative error is computed as

$$\delta := \frac{\left|C_{\mathrm{D}}^{\mathrm{ref}} - \bar{C}_{\mathrm{D}}\right|}{\left|C_{\mathrm{D}}^{\mathrm{ref}}\right|}. \tag{6.6}$$

This quantity is reported in decimal units throughout the results.

Before performing UQ, we need a deterministic solver that computes the drag coefficient for a given inlet velocity. In OpenLB, setting up a flow solver involves defining the computational domain, specifying boundary conditions, and executing the time-stepping loop until convergence. The function `simulateCylinder(...)` encapsulates this process. A typical OpenLB setup follows these steps [37]:

1. Initialize the unit conversion:
   The `UnitConverter` handles transformations between physical and lattice units. It is initialized based on the resolution and relaxation time.

```
UnitConverterFromResolutionAndRelaxationTime<T, DESCRIPTOR> const converter
    (...);
```

2. Prepare the computational geometry:
   The computational domain, including the flow boundaries and obstacle, is defined using the `SuperGeometry` class.

```
SuperGeometry<T,2> superGeometry(...);
```

3. Prepare the LBM grid:
   The flow is simulated on a discrete lattice. A $D2Q9$ lattice is commonly used.

```
SuperLattice2D<T, DESCRIPTOR> lattice(...);
prepareLattice(...);
```

4. Time-stepping loop:
   The solver iterates over time steps until the solution reaches a steady state or periodic behavior. Boundary conditions are also enforced at each time step.

```
for (std::size_t iT = 0; iT <= converter.getLatticeTime(maxPhysT); ++iT) {
    // Update boundary conditions
    setBoundaryValues(...);

    // Execute collide and stream step
```

```
    lattice.collideAndStream();

    // Compute and output results at regular intervals
    if (iT % converter.getLatticeTime(.1) == 0) {
        drag = getResults(...);
    }
}
```

After setting up the deterministic solver, we integrate it into the UQ framework. A function `simulateCylinder(...)` internally executes the steps above while allowing variations in the inlet velocity for different UQ samples.

In the following, we document how the uncertainty in the inlet velocity is incorporated into the cylinder flow simulation. The quantity of interest is the drag coefficient on the cylinder surface, which is sensitive to variations in the inflow velocity. By performing multiple simulations (referred to as samples), we can capture how uncertain inlet conditions affect the overall flow behavior, particularly the drag coefficient.

To account for uncertainty in the inlet velocity, we specify a uniform distribution over a user-defined range. In code, we define this range with:

```
auto dist = uniform(0.8 * u0, 1.2 * u0);
```

This object, `dist`, specifies that our inlet velocity is drawn from a uniform distribution with minimum `0.8 * u0` and maximum `1.2 * u0`.

Depending on the desired approach, the user can choose either MCS or a SC-gPC method for the UQ:

- **MCS:** This approach samples the distribution multiple times (denoted by `N`) to gather a statistical representation of the drag coefficient. A seed value can be used for reproducibility:

```
UncertaintyQuantification uq(UQMethod::MonteCarlo);
unsigned int seed = 123456;
uq.initializeMonteCarlo(N, dist, seed);
```

- **SC-gPC:** This method uses polynomial expansions to more efficiently approximate the dependence of the drag coefficient on the inlet velocity:

```
UncertaintyQuantification uq(UQMethod::GPC);
uq.initializeGPC(order, N, dist);
```

  The parameter `order` refers to the polynomial order in the generalized polynomial chaos expansion, and the parameter `N` refers to the quadrature points used in each uncertainty dimension.

After setting up the desired UQ method, the user extracts the actual sample points (inlet velocity values) to be used in the flow solver:

```
auto samples = uq.getSamplingPoints();
```

Each element `samples[n][0]` corresponds to one instance of the inlet velocity for the `n`th simulation.

With the sample velocities at hand, we can run the cylinder flow solver for each sample. In this test case, the solver computes the flow around the cylinder on a mesh of resolution `resolution`, and outputs the drag coefficient:

```
for (size_t n = 0; n < samples.size(); ++n) {
    // Run the cylinder simulation with the given inlet velocity
    dragCoefficients[n] = simulateCylinder(
        resolution,
        samples[n][0],
        exportResults
    );
}
```

Once all simulations are completed, the drag coefficients which are stored in `dragCoefficients` can be processed to compute the mean and standard deviation, thereby quantifying the statistical response of the flow to uncertain inflow conditions:

```
double meanDrag = uq.mean(dragCoefficients);
double stdDrag  = uq.std(dragCoefficients);
```

Initially, we perform a stochastic consistency study for SC-gPC on the 2D cylinder flow described above. This evaluation is based on the relative error, denoted as δ, of the mean and standard deviation of drag coefficient C_D. We investigate this across different spatial resolutions (here $n_x = 10, 20, 40, 80$) and polynomial orders ($N = 1, 2, 3, \ldots, 10$), quadrature points ($N_q = 2N+1$). Hence, we compute

$$\delta = \frac{\left|\bar{C_\mathrm{D}}^{n_x} - \bar{C_\mathrm{D}}_{10}^{nx}\right|}{\left|\bar{C_\mathrm{D}}_{10}^{n_x}(t)\right|}, \tag{6.7}$$

$$\delta = \frac{\left|\sigma\left(C_\mathrm{D}{}_N^{n_x}(t)\right) - \sigma\left(C_\mathrm{D}{}_{10}^{n_x}(t)\right)\right|}{\left|\sigma\left(C_\mathrm{D}{}_{10}^{n_x}(t)\right)\right|}, \tag{6.8}$$

respectively.

The convergence results in terms of the obtained relative error over several resolutions and polynomial orders are presented in Figure 6.1 for two dedicated points in time. The results show that a polynomial order of $N = 5$ is sufficient for achieving the highest accuracy.

From Figure 6.1, it can be observed that the relative errors converge exponentially to machine precision. This indicates that here the SC-gPC achieves spectral accuracy in the random space. The mean and

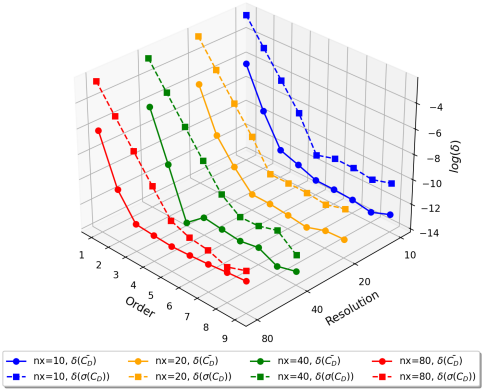


Figure 6.1: Relative error (δ) of expectation value ($\bar{C_{\mathrm{D}}}$) and standard deviation ($\sigma(C_{\mathrm{D}})$) of drag coefficient C_{D} for two-dimensional cylinder flow computed with SC-gPC with respect to highest polynomial order result. Several spatial resolutions ($n_x = 10, 20, 40, 80$) and polynomial orders ($N = 1, 2, 3, \ldots, 10$) are tested.

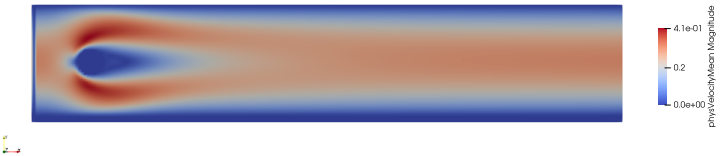
standard deviation of the velocity magnitude of the flow field, computed using a 5th-order SC-gPC method with 11 quadrature points ($N_q = 11$), are shown in Figure 6.2.

Comparing MCS at resolution $n_x = 20$, we consider different numbers of samples ($N_q = 10, 20, 40, 80, 100$). The results are compared with SC-gPC to analyze the convergence behavior. As shown in Figure 6.3, the relative error of the estimated mean and standard deviation of the C_{D} decreases as the number of MCS samples increases.

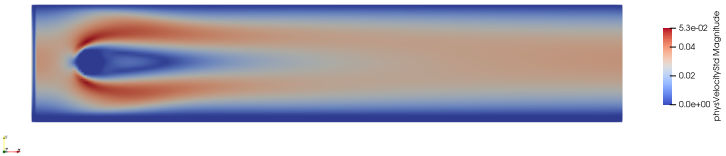
The SC-gPC results serve as a reference, demonstrating the efficiency of spectral approaches in capturing uncertainty with fewer samples.

Table 6.1: Comparison of mean drag coefficient $\bar{C_{\mathrm{D}}}$ and standard deviation $\sigma(C_{\mathrm{D}})$ across different UQ methods with resolution $n_x = 20$.

	MC	QMC	SC-gPC	deterministic
$\bar{C_{\mathrm{D}}}$	5.66731	5.67270	5.67261	5.63208
$\sigma(C_{\mathrm{D}})$	0.34896	0.34890	0.34888	–



(a) Mean velocity magnitude, $|\bar{\boldsymbol{u}}|$



(b) Standard deviation of velocity magnitude, $\sigma(|\boldsymbol{u}|)$

Figure 6.2: Mean ($|\bar{\boldsymbol{u}}|$) and standard deviation ($\sigma(|\boldsymbol{u}|)$) of the velocity magnitude in the two-dimensional cylinder flow, computed using SC-gPC with a 5th-order polynomial expansion and 11 quadrature points ($N_q = 11$).

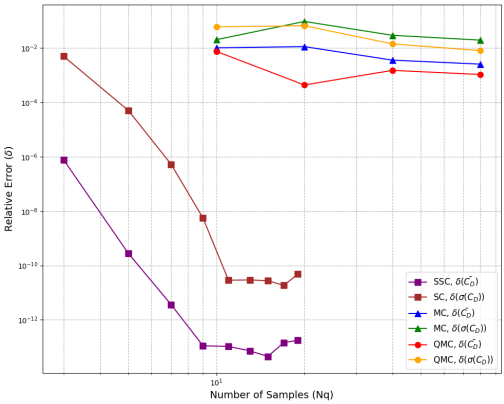


Figure 6.3: Relative error of the expectation value ($\delta(\bar{C_{\mathrm{D}}})$) and standard deviation ($\delta(\sigma(C_{\mathrm{D}}))$) of the drag coefficient for the two-dimensional cylinder flow at $n_x = 20$. The MCS and QMC results are computed with different sample sizes ($N_q = 10, 20, 40, 80, 100$) and compared against SC-gPC.

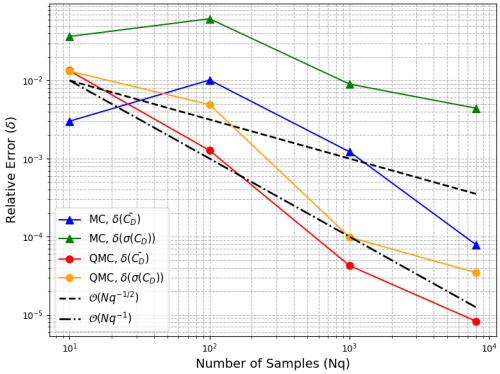


Figure 6.4: Relative error of the expectation value $\delta(\bar{C_{\mathrm{D}}})$ and standard deviation $\delta(\sigma(C_{\mathrm{D}}))$ of the drag coefficient for the two-dimensional cylinder flow at $n_x = 20$. The MCS and QMC results are computed with several sample sizes ($N_q = 10, 100, 1000, 8000$).

6.1.2 Taylor–Green vortex flow with uncertain viscosity

We proceed with the TGV flow in a two-dimensional periodic domain $d_x = 2$, a classical benchmark problem for incompressible flow simulations. The TGV flow is fully periodic and admits a known analytical solution to the NSE, making it ideal for validating numerical methods by direct comparison with reference solutions.

In this test case, we introduce uncertainty in the flow viscosity via a random Reynolds number. Specifically, we model the Reynolds number as a uniformly distributed random variable,

$$\mathrm{Re} \sim \mathcal{U}(0.8\,\mathrm{Re}_0,\ 1.2\,\mathrm{Re}_0), \quad \text{with } \mathrm{Re}_0 = 15.$$

The corresponding kinematic viscosity is computed per realization as

$$\nu = \frac{u_0 L}{\mathrm{Re}},$$

which induces uncertainty in the decay rate of the flow. This allows us to quantify the resulting variation in the velocity field and its kinetic energy. Statistical estimates (e.g., mean and standard deviation of velocity magnitude and kinetic energy) are computed using SC-gPC and MCS.

The exact velocity field $\boldsymbol{u}(x, y, t)$ and pressure field $p(x, y, t)$ are given by

$$\boldsymbol{u}(x, y, t) = \begin{pmatrix} -u_0 \cos\big(k_x x\big) \sin\big(k_y y\big) \mathrm{e}^{\frac{-t}{t_d}} \\ u_0 \sin\big(k_x x\big) \cos\big(k_y y\big) \mathrm{e}^{\frac{-t}{t_d}} \end{pmatrix}, \tag{6.9}$$

$$p(x, y, t) = -\frac{1}{4} u_0^2 \left[\cos\left(2k_x x\right) + \left(\frac{k_x}{k_y}\right)^2 \cos\left(2k_y y\right) \right] \mathrm{e}^{\frac{-2t}{t_d}} + P_0. \tag{6.10}$$

In Eq. (6.9) and Eq. (6.10), u_0 is the initial velocity amplitude, k_x and k_y are wavenumbers corresponding to the domain length in the x and y directions, and t_d is a characteristic decay time that depends on the fluid viscosity.

We define a square domain $\Omega = [\,0, n_x\,] \times [\,0, n_y\,]$ in lattice units, where n_x and n_y denote the number of grid nodes in each direction. For simplicity, we let $n_x = n_y$, and we set the physical domain size to $L = 2\pi$. Hence, the wavenumbers become

$$k_x = \frac{2\pi}{n_x}, \qquad k_y = \frac{2\pi}{n_y}. \tag{6.11}$$

The initial density is determined via the ideal equation of state $\rho = p/c_s^2$, where c_s is the speed of sound. The characteristic decay time is defined by

$$t_d = \frac{1}{\nu, (k_x^2 + k_y^2)} \tag{6.12}$$

and the corresponding lattice Boltzmann relaxation time is given by

$$\tau = \frac{\nu}{c_s^2} + \frac{1}{2}. \tag{6.13}$$

All simulations for this test case enforce periodic boundary conditions in both spatial directions to obtain a periodic numerical solution. The first global statistical quantity that we study is the (normalized) total kinetic energy, defined as

$$K(t) = \frac{2}{|\Omega|\, u_0^2} \int_\Omega \Big(u^2(x, y, t) + v^2(x, y, t) \Big) \,\mathrm{d}x \,\mathrm{d}y, \tag{6.14}$$

where Ω is the two-dimensional computational domain $[0, n_x] \times [0, n_y]$. The factor of $2/(|\Omega| u_0^2)$ ensures normalization by the domain area $|\Omega|$ and the square of the initial velocity amplitude u_0. In practice, we approximate the integral via a discrete sum over the lattice nodes, i.e.

$$K(t) \approx \frac{2}{n_x n_y u_0^2} \sum_{x=0}^{n_x - 1} \sum_{y=0}^{n_y - 1} \big[u^2(x, y, t) + v^2(x, y, t) \big]. \tag{6.15}$$

The implementation of the case in OpenLB-UQ follows the same guidelines as explained in detail for the flow around a cylinder case in Section 6.1.1.

To assess convergence toward a reference solution $K_{\mathrm{ref}}(t)$, we define the relative error in the mean kinetic energy as

$$\delta(t) = \frac{\left|\overline{K}(t) - K_{\mathrm{ref}}(t)\right|}{\left|K_{\mathrm{ref}}(t)\right|}, \tag{6.16}$$

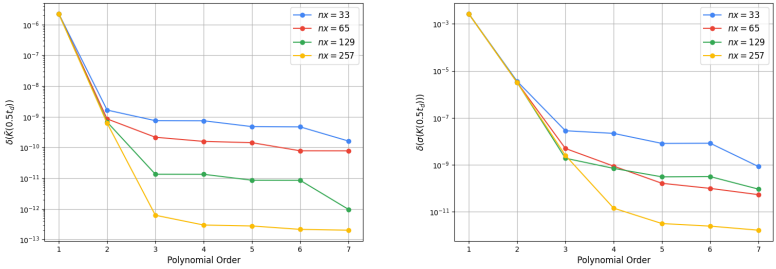
where $\overline{K}(t)$ is the computed mean and $K_{\mathrm{ref}}(t)$ is obtained from a finer spatial or stochastic discretization.

In SC-gPC, accuracy depends on both the polynomial order N and the quadrature points number N_q, since a deterministic solve is performed at each quadrature node. To study the effect of N alone, we fix $N_q = 1000$ and vary N from 1 to 8.

Figure 6.5 shows $\delta(t)$ for the mean $\overline{K}(t)$ and the standard deviation $\sigma(K(t))$ of the normalized total kinetic energy at $t = 0.5\,t_d$, measured against the highest polynomial order solution. The error decreases exponentially with N, confirming the spectral convergence of SC-gPC when N_q is sufficiently large.

Beyond a certain order (e.g., $N \approx 3$–4), the convergence curves flatten, indicating that the total error is dominated by the deterministic discretization in physical space.

In other words, at sufficiently high polynomial order, the SC approach becomes more accurate than the underlying spatial grid can represent, and hence the relative error plateaus at the level prescribed by the spatial discretization error.



(a) Expectation value errors $\delta(\overline{K}(0.5t_d))$

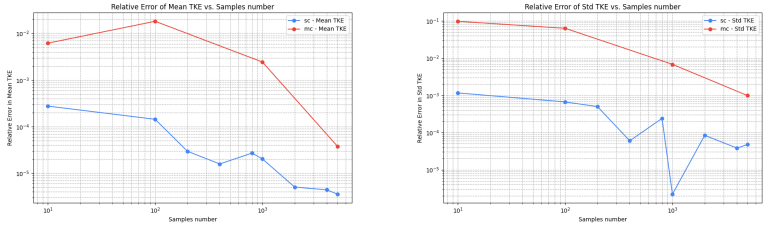
(b) Standard deviation errors $\delta(\sigma(K(0.5t_d)))$

Figure 6.5: Relative error ($\delta(t)$, Eq (6.16)) of expectation value $\overline{K}(t)$ and standard deviation $\sigma(K(t))$ of normalized total kinetic energy $K(t)$ for TGV flow with uncertain viscosity computed with SC LBM with respect to highest polynomial order ($k = 8$) results, respectively for individual resolutions at time $t = 0.5t_d$. Several spatial resolutions ($n_x = 33, 65, 129, 257$) and polynomial orders ($N = 1, 2, 3, \ldots, 8$), quadrature points ($N_q = 1000$) are tested.

Following the convergence analysis in Figure 6.5, we fix the polynomial order at $N = 5$ and investigate the effect of the quadrature resolution N_q in SC-gPC, comparing it against MCS.

Figure 6.6 compares SC-gPC ($N = 5$) and MCS for the two-dimensional TGV flow at $n_x = 33$, showing the relative error of the mean and standard deviation of $K(t)$ at $t = 0.5\,t_d$ as functions of the number of quadrature points (SC-gPC) or samples (MCS).

For MCS, the errors decay at the expected $O(N^{-0.5})$ rate for both mean and standard deviation. With SC-gPC at fixed polynomial order ($N = 5$), increasing the number of quadrature points still yields a much faster convergence than MCS.



(a) Expectation value errors $\delta(\bar{K}(0.5t_d))$

(b) Standard deviation errors $\delta(\sigma(K(0.5t_d)))$

Figure 6.6: Relative error ($\delta(t)$, Eq (6.16)) of expectation value $\bar{K}(t)$ and standard deviation $\sigma(K(t))$ of normalized total kinetic energy $K(t)$ for TGV flow with uncertain viscosity computed with SC LBM with respect to highest quadrature points number ($N_q = 10000$) results at time $t = 0.5t_d$. Several spatial resolutions ($n_x = 33$) and quadrature points ($N_q = 10, 100, 200, \ldots, 10000$), Monte Carlo samples ($N_q = 10, 100, 10000$) are tested.

6.1.3 Taylor–Green vortex flow with four-dimensional uncertain initial velocity perturbation

This second version of the TGV flow is taken from [65, 98]. The velocity field and the pressure of the TGV are defined as

$$\boldsymbol{u}(x,y,t) = \begin{pmatrix} u(x,y,t) \\ v(x,y,t) \end{pmatrix} = \begin{pmatrix} -u_0^R \cos(2x)\sin(2y)\, e^{-\frac{t}{t_d}} \\ u_0^R \sin(2x)\cos(2y)\, e^{\frac{t}{t_d}} \end{pmatrix}, \tag{6.17}$$

$$p(x,y,t) = -\frac{1}{4}u_0^{R\,2}\left[\cos(4x) + \left(\frac{2}{2}\right)^2 \cos(4y)\right] e^{-\frac{2t}{t_d}} + P_0, \tag{6.18}$$

respectively, where, $x, y \in [0, 2\pi]$. The uncertain velocity $u_0^R = u_0 + \epsilon_d(x)$ is based on the perturbation ϵ_d given by the first-order harmonics with four-dimensional uniformly i.i.d. random amplitude $\zeta_{d,i,j} \sim \mathcal{U}(-0.025, 0.025)$, i.e.

$$\epsilon_d(x, y) = \frac{1}{4} \sum_{(i,j)\in\{0,1\}^2} \zeta_{d,i,j} k_i(4x) k_j(4y), \tag{6.19}$$

where

$$k_i(x) = \begin{cases} \sin(x), & \text{if } i = 0, \\ \cos(x), & \text{if } i = 1. \end{cases} \tag{6.20}$$

The four-dimensional uncertainty is straightforward to set up in our framework by defining a joint distribution as follows:

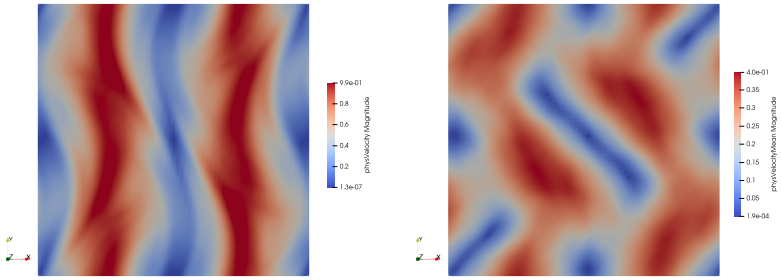
```
auto perturb1 = uniform(-0.025, 0.025);
auto perturb2 = uniform(-0.025, 0.025);
auto perturb3 = uniform(-0.025, 0.025);
auto perturb4 = uniform(-0.025, 0.025);
auto jointPerturb = joint({perturb1, perturb2, perturb3, perturb4});
```

The maximum simulation time is $T_{\max} = 100$. Other parameter settings for the test case are described below and summarized in Table 6.2. The number of samples N, the spatial resolution $n_x = n_y$, and the Reynolds number Re scale linearly, while the Mach number Ma scales inversely proportional. Based on the latter three parameters, the relaxation time τ, the time step $\triangle t$, and the grid spacing $\triangle x$ are determined.

Table 6.2: Simulation parameters for the TGV with four-dimensional uncertain initial condition.

N	$n_x = n_y$	Ma	Re	τ	$\triangle x$	$\triangle t$
8	8	0.2	320	0.501206	0.897598	0.103646
16	16	0.1	640	0.500646	0.418879	0.024184
32	32	0.05	1280	0.500334	0.202683	0.005850
64	64	0.025	2560	0.500170	0.099733	0.001439
128	128	0.0125	5120	0.500085	0.049473	0.000357
256	256	0.00625	10240	0.500043	0.024639	0.000088

Snapshots of the velocity magnitude computed from the TGV for deterministic and perturbed initial conditions (expected value in the latter case), respectively, are shown for $t = T_{\max}$ in Figure 6.7.



(a) Deterministic **(b)** Expected value

Figure 6.7: Deterministic and expected velocity magnitude of the TGV with four-dimensional uncertain initial velocity computed with OpenLB-UQ for a resolution of $n = 256$ at $t = T_{\max}$.

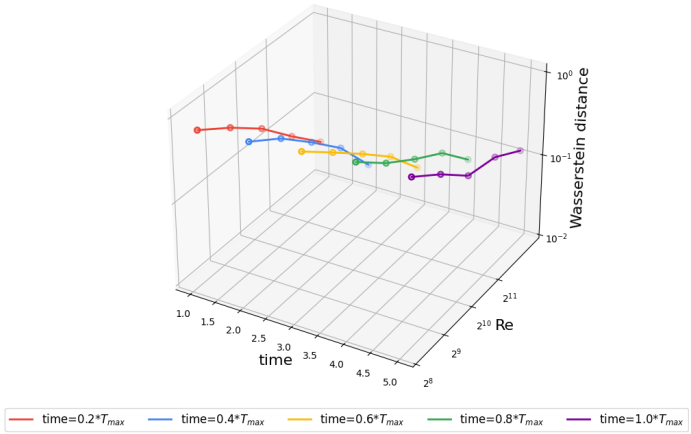


Figure 6.8: 1-Wasserstein distance at several time steps of the numerical statistical solutions with respect to the most resolved one ($n_x = 256$) for the TGV with four-dimensional uncertain initial velocity computed with OpenLB-UQ.

To evaluate the inviscid statistical convergence [45] of OpenLB-UQ, we approximate the 1-Wasserstein distance defined as

$$W_1(Q_1, Q_2) = \left(\inf_{\gamma \in \Gamma(Q_1, Q_2)} \int_{X^2} |x - y| \, d\gamma(x, y) \right), \tag{6.21}$$

where $\Gamma(Q_1, Q_2)$ denotes the set of all joint distributions γ on $M \times M$ whose marginals are Q_1 and Q_2 respectively, $d(x, y)$ is the distance between points x and y in the metric space M. Here, Q_1 and

Q_2 are numerical statistical solutions for resolutions $n_x < 256$ and the reference numerical statistical solution with resolution $n_x = 256$ (with matching parameters according to Table 6.2), respectively.

The time-dependent numerical statistical solutions are constructed by expectation values of Dirac measures from computed sample velocity fields at time t. Hence, with each spatial resolution n_x, we approximate the continuous statistical solutions to an initially perturbed NSE on $\mathcal{X} = \mathbb{R}^{d_x}$ defined by probability measures on $L^2_{\mathrm{div}}(\mathcal{X};\mathcal{U})$ [65].

We approximate the Wasserstein distance from the computed sample data using the function `scipy.stats.wasserstein_distance` in `SciPy` [82]. If the approximated Wasserstein distance of the numerical statistical solutions for $n_x = 8, 16, 32, 64, 128$ to the most resolved one for $n_x = 256$, i.e. $W_1(|u|_{n_x}(t), |u|_{n_x=256}(t))$ converges, we thus indicate Wasserstein convergence towards a unique statistical solution of the incompressible Euler equations [45]. The results indeed show that $W_1(|u|_n, |u|_{n_x=256})$ converges as the Reynolds number increases with n_x, cf. Figure 6.8. The estimated convergence rates are provided in Figure 6.9. Note that the decrease of the Wasserstein convergence order over time toward a value of 0.5 (see Figure 6.9f) is expected for MCS.

6.2 Performance evaluation

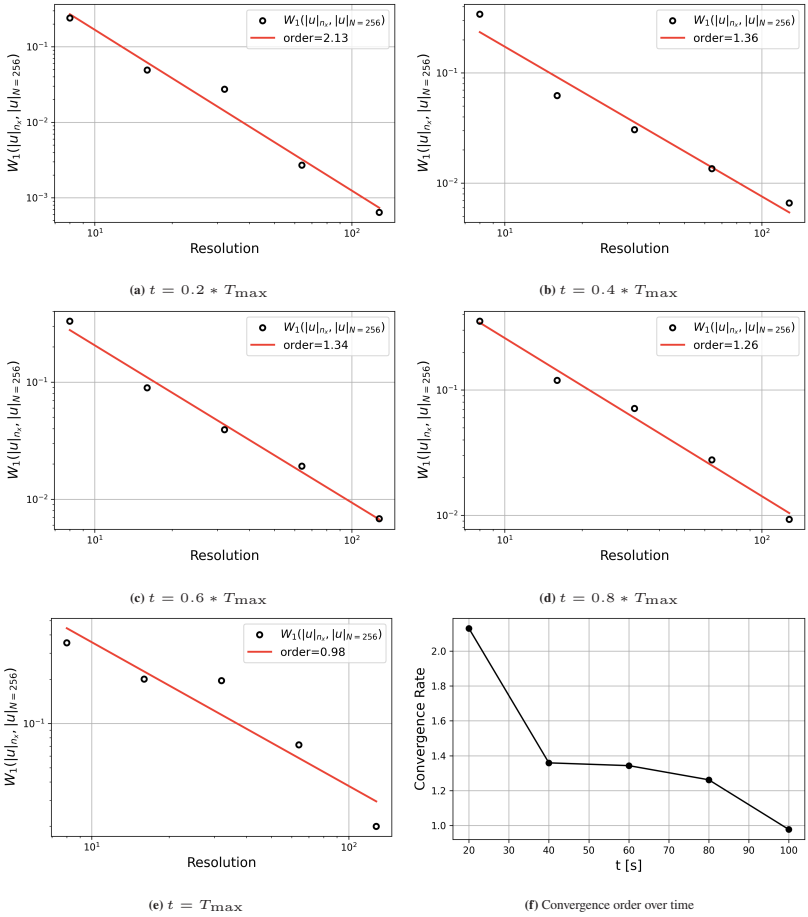
To evaluate the scalability of OpenLB-UQ, we conducted additional performance tests on the HoreKa supercomputer. Specifically, we utilized a CPU-only compute node with two Intel Xeon Platinum 8368 processors, offering a total of 76 physical cores (152 hardware threads) per node. Each processor operates at a base frequency of 2.40 GHz. All simulations were executed on a single node to focus exclusively on intra-node parallel efficiency.

To assess the practical benefits of different parallelization strategies, we compared the total simulation time required to process all 137 samples using two parallelization strategies:

1. **Sample-level decomposition**: Each sample is executed independently on different processes or cores, without MPI-based domain decomposition within a single sample.

2. **Domain-level decomposition**: Each sample runs in parallel using MPI across multiple cores with domain decomposition.

First, we evaluate the scalability of the domain-level (MPI-based) parallelization strategy using the `cylinder2d` test case. For each resolution ($n_x = 10$, 20, 40, and 80), we perform strong-scaling tests by varying the number of MPI processes from 2 to 64. As shown in Figure 6.10, the performance gain of domain-level parallelization is saturated by communication overhead at different process counts depending on the spatial resolution. To ensure efficient resource utilization without incurring excessive parallel overhead, we selected the saturation point for each resolution as the number of processes used in subsequent tests. The chosen MPI process counts for each resolution are summarized in Table 6.3.



(a) $t = 0.2 * T_{\max}$ **(b)** $t = 0.4 * T_{\max}$ **(c)** $t = 0.6 * T_{\max}$ **(d)** $t = 0.8 * T_{\max}$ **(e)** $t = T_{\max}$ **(f)** Convergence order over time

Figure 6.9: Convergence orders of 1-Wasserstein distance at several time steps (a,b,c,d,e) and over time (f) of the numerical statistical solutions with respect to the most resolved one ($n_x = 256$) for the TGV with four-dimensional uncertain initial velocity computed with OpenLB-UQ.

These configurations are consistently applied in all domain-level simulations and postprocessing runs presented in the following analyses.

Subsequently, we assessed the overall parallel performance of both sample-level and domain-level decomposition strategies on the `cylinder2d` case. For each resolution, we performed 100 Monte Carlo samples. Table 6.4 summarizes the average computational time per sample, total simulation

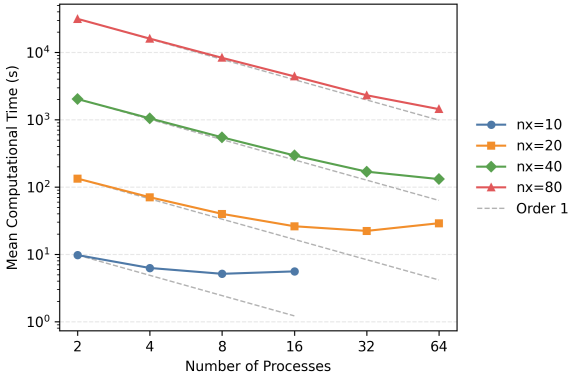


Figure 6.10: Strong-scaling performance of the domain-level parallelization (MPI) for the `cylinder2d` case. The plot shows the mean computational time per sample as a function of the number of processes for different resolutions n_x.

Table 6.3: Selected number of MPI processes at which domain-level parallelization saturates for each spatial resolution n_x in `cylinder2d`. These configurations are used in all subsequent domain-level simulations and postprocessing.

n_x	10	20	40	80
MPI processes	2	4	16	64

time, and the resulting speedup factors relative to the fully sequential execution. The speedup S achieved by each parallelization strategy is computed as

$$S = \frac{T_{\text{seq}}}{T_{\text{par}}}, \tag{6.22}$$

where T_{seq} denotes the total execution time of the fully sequential baseline and T_{par} represents the total execution time of the corresponding parallel execution.

In addition to the total simulation time, we also analyzed the parallel performance of the postprocessing phase, which aggregates and processes the results from all Monte Carlo samples. Table 6.5 reports the total postprocessing time required for 100 samples at each resolution. The results demonstrate that domain-level parallelization significantly reduces postprocessing time compared to the sample-level approach. Since the postprocessing under sample-level decomposition is executed sequentially—identical to the baseline—we omit it from the domain-level-only comparison.

Figure 6.11 presents the overall speedup factors achieved by the two parallelization strategies across different resolutions.

Table 6.4: Comparison of parallel performance for the `cylinder2d` case under sample-level and domain-level decomposition on a single node. The time per sample represents the average time across all samples.

n_x	Parallelization strategy	Time per sample [s]	Total time [s]	Speedup
10	Sequential	15.92	1645.49	–
	Sample-level	17.24	1004.09	1.63
	Domain-level	8.99	1025.20	1.60
20	Sequential	252.21	25473.95	–
	Sample-level	256.54	6689.76	3.80
	Domain-level	69.55	7151.24	3.56
40	Sequential	3814.69	382457.32	–
	Sample-level	4041.55	28971.26	13.20
	Domain-level	291.12	29433.72	12.99
80	Sequential	61535.14	6153514.42	–
	Sample-level	76213.13	152881.41	40.25
	Domain-level	1391.92	139358.12	44.15

Table 6.5: Comparison of parallel postprocessing time for 100 Monte Carlo samples of the `cylinder2d` case under sample-level and domain-level decomposition on a single node.

n_x	Parallelization strategy	Total postprocessing time [s]
10	Sample-level	51.37
	Domain-level	24.71
20	Sample-level	202.76
	Domain-level	50.66
40	Sample-level	802.26
	Domain-level	51.32
80	Sample-level	3213.26
	Domain-level	50.27

To visualize the relative contribution of sampling and postprocessing phases under different strategies and resolutions, Figure 6.12 shows the breakdown of the total computational time for 100 Monte Carlo samples.

Figure 6.13 shows weak scaling results for the `cylinder2d` case using sample-level parallelization without postprocessing. Each process is assigned 100 samples, resulting in batch sizes of 200, 400, 800, and 1600 samples for 2, 4, 8, and 16 processes, respectively. This batch configuration is kept consistent across all spatial resolutions. As expected for MCS, the total simulation time remains nearly constant across batch sizes due to the embarrassingly parallel nature of the computation. Since SC-gPC also employs non-intrusive sampling, this result equally confirms the scalability of SC-based UQ methods.

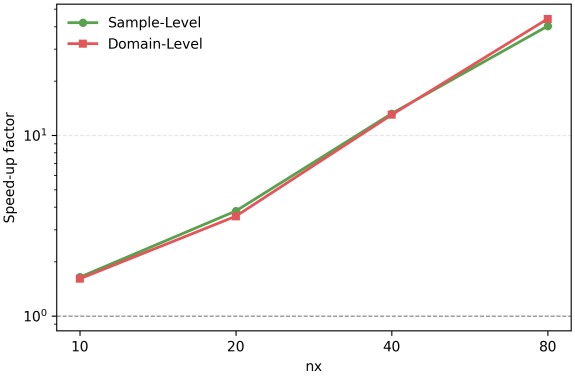


Figure 6.11: Speedup factors achieved by sample-level and domain-level parallelization strategies for the `cylinder2d` case at different resolutions n_x. The reference baseline for speedup calculation is the sequential execution time.

In summary, the choice between sample-level and domain-level parallelization depends strongly on the problem size and grid resolution. For coarse grids, the sample-level decomposition exhibits a clear advantage due to its embarrassingly parallel nature and minimal communication overhead. However, as the grid resolution increases, the scalability of domain-level (MPI-based) parallelization tends to improve, offering higher speedup for large-scale problems. Therefore, selecting the appropriate parallelization strategy requires balancing between available computational resources, problem size, and desired time-to-solution.

To evaluate the computational efficiency of the SC-gPC method, we compare the relative error of the mean and standard deviation of the drag coefficient C_D with respect to the total CPU time at multiple grid resolutions. The results are shown in Figure 6.14.

For the TGV case with four-dimensional uncertainty (`tgv2d`), a third-order, three-level Smolyak sparse grid quadrature was employed to compute the SC-gPC approximation. This configuration required a total of 137 quadrature samples, which provides a reasonable approximation of the target statistics.

To optimize parallel performance, we first evaluated the mean computational time per sample under domain-level MPI decomposition for different spatial resolutions $n_x \in \{64, 128, 256, 512\}$. The goal was to identify the optimal number of MPI processes for each grid size. The results, shown in Figure 6.15, indicate that process number $\{2, 8, 32, 64\}$ will get the best parallel performance.

As above, due to the saturation points, the chosen MPI process counts for each resolution are summarized in Table 6.6. These configurations are consistently applied in all domain-level simulations and postprocessing runs presented below.

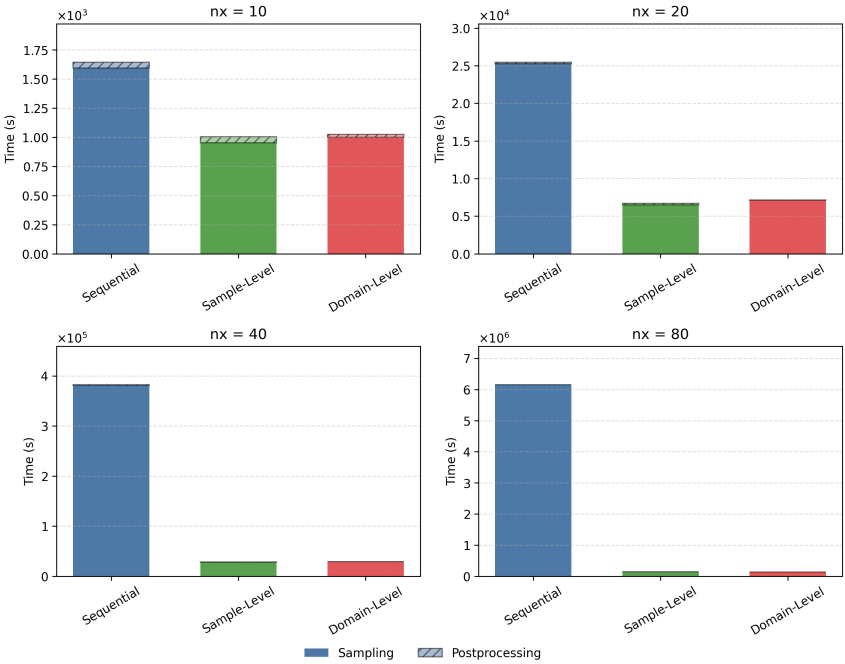


Figure 6.12: Breakdown of total computational time for 100 Monte Carlo samples of `cylinder2d` at different resolutions n_x, comparing sequential execution, sample-level parallelization, and domain-level parallelization. Each bar shows the sampling time and the postprocessing time.

Table 6.6: Selected number of MPI processes at which domain-level parallelization saturates for each spatial resolution n_x in `tgv2d`. These configurations are used in all subsequent domain-level simulations and postprocessing.

n_x	64	128	256	512
MPI Processes	2	4	16	64

Table 6.7 summarizes the average time per sample, total simulation time, and speedup relative to the fully sequential baseline for each strategy and resolution. As shown, sample-level decomposition exhibits superior performance for fine grids, achieving up to a speedup of 30.78 at $n_x = 512$. However, the domain-level strategy demonstrates increasing scalability as the grid size grows. For $n_x = 512$, it achieves a speedup of 30.97, closing the gap to sample-level decomposition.

Table 6.8 reports the total postprocessing time required for 100 samples at each resolution. The results demonstrate a similar behavior as in the `cylinder2d` case.

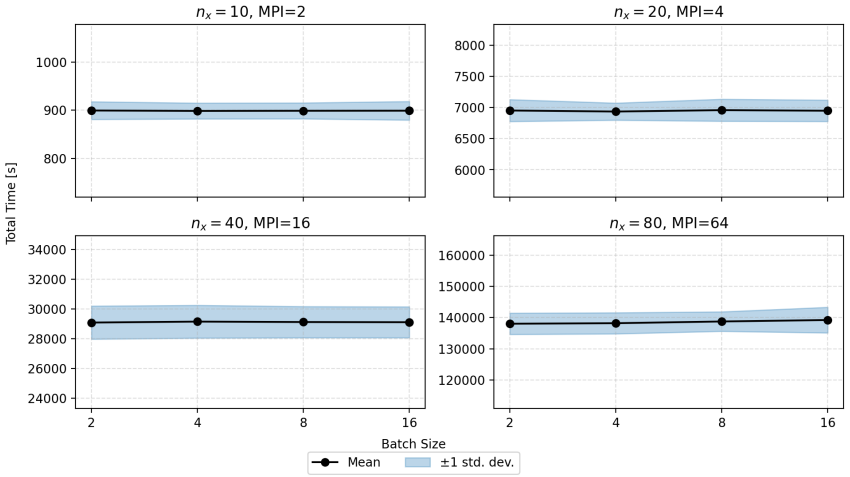


Figure 6.13: Weak scaling performance of sample-level parallelization for the `cylinder2d` case at several resolutions ($n_x = 10, 20, 40, 80$) without postprocessing. The black line shows the mean total time per batch size, with the blue shaded area indicating the standard deviation across repeated runs. Each MPI process is assigned 100 samples, resulting in batch sizes of 200, 400, 800, and 1600 for 2, 4, 8, and 16 processes, respectively. This batch configuration is applied uniformly across all resolutions.

Table 6.7: Comparison of parallel performance for the tgv2d case under sample-level and domain-level decomposition on a single node. The time per sample represents the average time across all samples.

n_x	Parallelization strategy	Time per sample [s]	Total time [s]	Speedup
64	Sequential	13.79	2031.12	–
	Sample-level	13.31	972.00	2.08
	Domain-level	7.17	1087.00	1.86
128	Sequential	209.23	28933.79	–
	Sample-level	210.23	7363.00	3.92
	Domain-level	56.00	7802.00	3.70
256	Sequential	3208.57	440211.52	–
	Sample-level	3240.71	29153.00	15.10
	Domain-level	233.05	32232.00	13.65
512	Sequential	50440.64	5046437.85	–
	Sample-level	60496.05	163902.88	30.78
	Domain-level	1180.67	162909.53	30.97

Figure 6.16 presents the overall speedup factors achieved by the two parallelization strategies across different resolutions. The sample-level strategy outperforms domain-level parallelization at all resolutions except for $n_x = 512$.

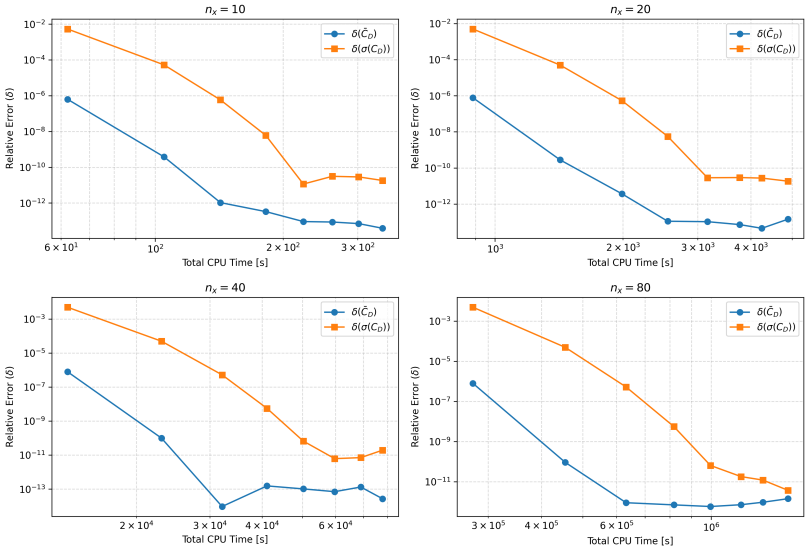


Figure 6.14: Performance of the SC-gPC method for estimating the mean and standard deviation of the drag coefficient C_D at spatial resolutions $n_x = 10, 20, 40, 80$. Each data point corresponds to a different polynomial order $N \in \{1, \ldots, 8\}$, and the reference solution is taken from the result at the highest order $N = 9$. The relative error is plotted against the total CPU time (sampling plus postprocessing).

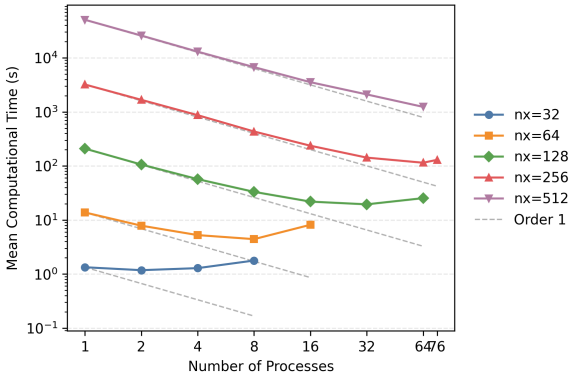


Figure 6.15: Strong-scaling performance of the domain-level parallelization (MPI) for the `tgv2d` case. The plot shows the mean computational time per sample as a function of the number of processes for different resolutions n_x.

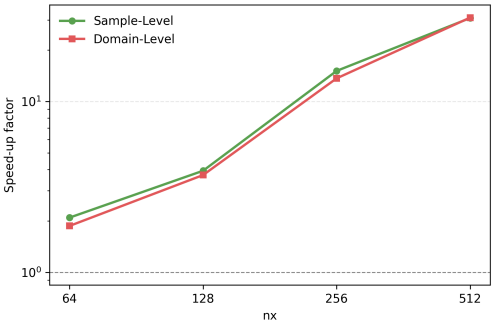


Figure 6.16: Speedup factors achieved by sample-level and domain-level parallelization strategies for the `tgv2d` case at different resolutions n_x. The reference baseline for speedup calculation is the sequential execution time.

Table 6.5 reports the total postprocessing time required for 100 samples at each resolution. The results demonstrate that domain-level parallelization significantly reduces postprocessing time compared to the sample-level approach. Since the postprocessing under sample-level decomposition is executed sequentially—identical to the baseline—we omit it from the domain-level-only comparison.

To visualize the relative contribution of sampling and postprocessing phases under different strategies and resolutions, Figure 6.17 shows the breakdown of the total computational time for 100 Monte Carlo samples.

Table 6.8: Comparison of parallel postprocessing time for third-order, three-level sparse grid samples of the `tgv2d` case under sample-level and domain-level decomposition on a single node.

n_x	Parallelization strategy	Total postprocessing time [s]
64	Sample-level	39.139
	Domain-level	20.605
128	Sample-level	150.6
	Domain-level	38.812
256	Sample-level	605.141
	Domain-level	39.119
512	Sample-level	2373.85
	Domain-level	42.530

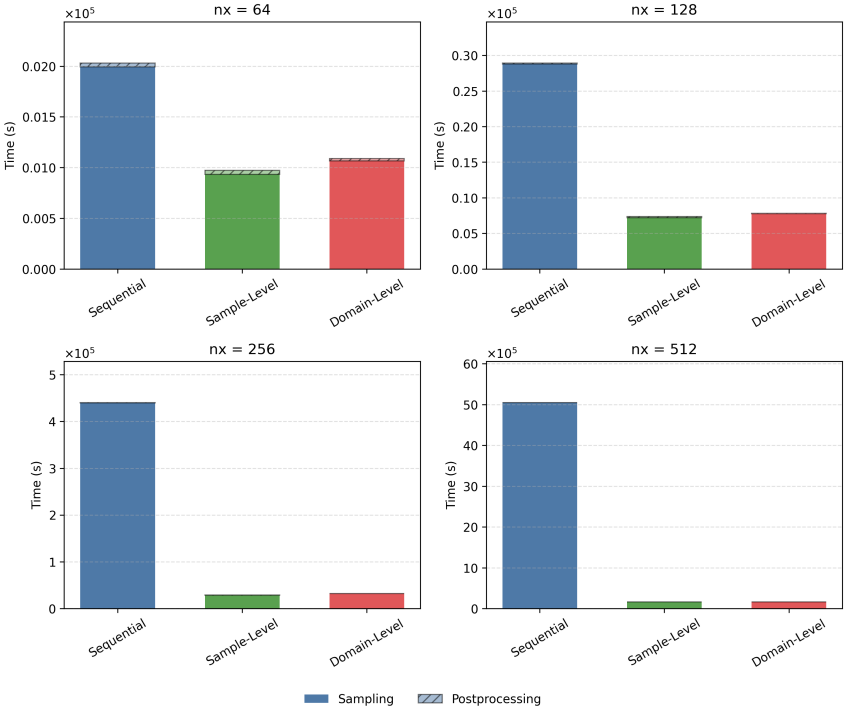


Figure 6.17: Breakdown of total computational time for 100 Monte Carlo samples of `tgv2d` at different resolutions n_x, comparing sequential execution, sample-level parallelization, and domain-level parallelization. Each bar shows the sampling time and the postprocessing time.

7 Uncertain Data Assimilation for Urban Wind Flow Simulations

In this chapter, we apply the OpenLB-UQ framework to the urban wind simulation with uncertain inflow conditions derived from measurements. Hourly wind speed and direction data from the Reutlingen weather station are introduced with uncertainty and imposed as boundary conditions in a large-eddy LBM simulation. Uncertainty is propagated using SC LBM, yielding ensembles from which spatio-temporal statistics such as mean fields, standard deviations, and confidence intervals at probe locations are computed. This study demonstrates how measurement uncertainty can be assimilated into CFD boundary data and translated into probabilistic flow predictions, providing interpretable diagnostics for urban design and environmental assessment, as further detailed in part of the preprint [100].

7.1 Deterministic lattice Boltzmann method (non-intrusive core)

We approximate the deterministic version of Eq. (2.3) and Eq. (2.4) using a recursive regularized LBM based on a third-order expanded equilibrium function and combined with a Smagorinsky–Lilly subgrid scale (SGS) model. An advanced version of this collision model, including hybridization and homogenization, is further described in [77]. The following quantities are thus assumed as filtered variables and not further denoted as such.

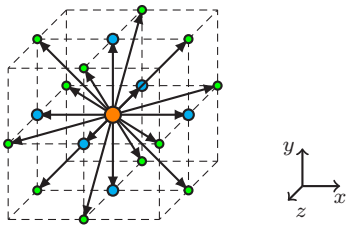


Figure 7.1: A schematic illustration of the discrete velocity set $D3Q19$. Coloring refers to energy shells: orange, cyan, green denote zeroth, first, second order, respectively. Figure from [62].

The evolution equation for the (filtered) particle distribution function $\boldsymbol{f} = \{f_i\}_{i=0,1,\ldots,q-1} \in \mathbb{R}^q$ is given by

$$f_i(\boldsymbol{x} + \boldsymbol{c}_i \triangle t, t + \triangle t) = f_i^{\mathrm{eq}}(\boldsymbol{x}, t) + \left(1 - \frac{1}{\tau_{\mathrm{eff}}(\boldsymbol{x}, t)}\right) f_i^{(1)}(\boldsymbol{x}, t), \quad \text{in } \mathcal{X}_{\triangle x} \times \mathcal{I}_{\triangle t}, \quad (7.1)$$

for $i = 0, 1, \ldots, q-1$, where $\boldsymbol{c}_i$ are the discrete lattice velocities of the $D3Q19$ model and $(\boldsymbol{x}, t)$ are grid nodes in the Cartesian discrete space-time cylinder $\mathcal{X}_{\triangle x} \times \mathcal{I}_{\triangle t}$. Moreover, τ_{eff} is the space-time adaptive, effective relaxation time of the SGS model, given by

$$\tau_{\mathrm{eff}}(\boldsymbol{x}, t) = \frac{\nu_{\mathrm{eff}}(\boldsymbol{x}, t)}{c_{\mathrm{s}}^2} \frac{\triangle t}{\triangle x^2} + \frac{1}{2}, \quad (7.2)$$

with $\nu_{\mathrm{eff}} = \nu + \nu_{\mathrm{turb}}$ representing the combined molecular (ν) and turbulent viscosity

$$\nu_{\mathrm{turb}}(\boldsymbol{x}, t) = (C_{\mathrm{S}} \triangle_{\boldsymbol{x}})^2 \left|\mathbf{S}(\boldsymbol{x}, t)\right|, \quad (7.3)$$

where $C_{\mathrm{S}} = 0.25$ is the Smagorinsky constant, $\triangle_{\boldsymbol{x}}$ is the filter width, and $\mathbf{S}$ is the (filtered) strain rate tensor

$$S_{\alpha\beta} = \frac{1}{2}\left(\frac{\partial u_\alpha}{\partial x_\beta} + \frac{\partial u_\beta}{\partial x_\alpha}\right). \quad (7.4)$$

Here, c_{s} is the lattice speed of sound, and we use $c_{\mathrm{s}}^2 = 1/3$ in lattice units. The regularization in Eq. (7.1) is based on the non-equilibrium function $f_i^{(1)} = f_i - f_i^{(0)}$ that is expanded as

$$f_i^{(1)}(\boldsymbol{x}, t) = \omega_i \sum_{n=0}^{N=3} \frac{1}{c_{\mathrm{s}}^{2n} n!} \mathbf{H}_i^{(n)} : \mathbf{a}_1^{(n)}(\boldsymbol{x}, t), \quad (7.5)$$

where ω_i are the lattice weights and we denote by $\mathbf{H}_i^{(n)}$ the nth order Hermite polynomial with the ith discrete velocity $\boldsymbol{c}_i$ as an argument. The Hermite coefficient for the non-equilibrium is defined as

$$\mathbf{a}_1^{(n)}(\boldsymbol{x}, t) = \sum_{i=0}^{q-1} \mathbf{H}_i^{(n)} f_i^{(1)}(\boldsymbol{x}, t). \quad (7.6)$$

Note that according to [32], we use Hermite polynomials that have correct orthogonality for $D3Q19$ only. The equilibrium distribution $f_i^{\mathrm{eq}}(\boldsymbol{f})$ is computed by an expansion using Hermite polynomials and Hermite equilibrium coefficients (Eq. (7.6) for f_i^{eq}) up to order three, i.e.

$$f_i^{\mathrm{eq}}(\boldsymbol{x}, t) = \omega_i \rho(\boldsymbol{x}, t) \left(1 + \frac{\boldsymbol{c}_i \cdot (\boldsymbol{u}(\boldsymbol{x}, t))}{c_{\mathrm{s}}^2} + \frac{\mathbf{H}_i^{(2)} : \mathbf{a}_{\mathrm{eq}}^{(2)}(\boldsymbol{x}, t)}{2c_{\mathrm{s}}^4 \rho(\boldsymbol{x}, t)} + \frac{\mathbf{H}_i^{(3)} : \mathbf{a}_{\mathrm{eq}}^{(3)}(\boldsymbol{x}, t)}{2c_{\mathrm{s}}^6 \rho(\boldsymbol{x}, t)}\right), \quad (7.7)$$

where the zeroth-order moment is denoted by $\rho = \sum_{i=0}^{q-1} f_i$ and the first-order local velocity moment is $\boldsymbol{u} = (1/\rho) \sum_{i=0}^{q-1} \boldsymbol{c}_i f_i$.

In addition, we prescribe deterministic initial conditions and stochastic boundary conditions to Eq. (2.3) and Eq. (2.4) to form a stochastic initial boundary value problem that models uncertain wind flow in an urban geometry. In this work, we focus on uncertainty in the inlet velocity due to measurement errors. Specifically, we model the uncertainty in the measured wind speed and direction with a respective relative perturbation ζ_1 and ζ_2, resulting in a logarithmic wind profile

$$\hat{\boldsymbol{y}}(\boldsymbol{x}, t, \boldsymbol{Z}), \quad \text{where } \boldsymbol{Z} \coloneqq \boldsymbol{\zeta} = (\zeta_1, \zeta_2)^{\mathrm{T}} \sim \mathcal{N}(1, 0.1^2) \times \mathcal{N}(0, 6^2). \tag{7.8}$$

The computational domain $\mathcal{X}_{\triangle x}$ is configured as a disc where the building geometry is centered on the ground. The boundary conditions are handled as follows:

- **Horizontal boundaries (mantle of cylinder):** The surrounding wind velocity $\hat{\boldsymbol{y}}(\boldsymbol{x}, t, \boldsymbol{Z})$ is prescribed via a (circular) local velocity boundary condition [33], where both the magnitude and the wind flow direction are treated as uncertain quantities (see Section 7.3).
- **Building walls and ground (bottom of cylinder):** A no-slip condition is enforced with the classical bounce-back method.
- **Sky (top of cylinder):** A free-slip condition is used by enforcing a zero normal velocity and a zero normal gradient of the tangential velocity.

The populations are initialized to equilibrium values, and the horizontal boundary conditions are increased to the first target values over time to stabilize the simulation. For a smooth change in the magnitude and direction of horizontal velocity boundaries, an interpolation time interval is used, and the boundary velocity is gradually transformed to the subsequent magnitude and direction.

that is further described below in Section 7.3 and realized as a convective boundary condition at the horizontal domain boundaries.

7.2 Uncertainty quantification workflow

Building on this deterministic setup, the following workflow is employed to propagate measurement uncertainty through the urban wind simulation:

1. **Select input measurement data.** Hourly wind speed $U_H(t)$ and direction $\Theta(t)$ recorded at the Reutlingen station (Germany, $48.4914°\,\mathrm{N},\ 9.2043°\,\mathrm{E}$), from 2024-11-07 20:00 to 2024-11-09 19:00 (CET).
2. **Define the input model.** Specify uncertain parameters, e.g., inlet velocity magnitude $\zeta_1 \sim \mathcal{N}(1, 0.1^2)$ and inflow direction $\zeta_2 \sim \mathcal{N}(0, 6^2)$. Select an appropriate polynomial basis (e.g., Hermite) for the expansion.

3. **Generate the quadrature rule.** Construct collocation nodes $\{\boldsymbol{Z}^{(j)}\}$ and weights $\{a_j\}$ using a dense tensor-product Gauss–Hermite quadrature.

4. **Apply boundary conditions.** For each node j, compute the inlet velocity uncertainty $\hat{\boldsymbol{y}}(\boldsymbol{x}, t, \boldsymbol{Z}^{(j)})$ via Eq. (7.8).

5. **Run deterministic simulations.** For each $\boldsymbol{Z}^{(j)}$, solve the LBM problem including hourly uncertain measurement data-assimilation during the simulation. Obtain the sample velocity $\boldsymbol{u}(\boldsymbol{x}, t, \boldsymbol{Z}^{(j)})$.

6. **Extract QoIs.** Examples include:

   - *Domain-wide fields:* single-sample, time-dependent and time-averaged velocity magnitudes at a given time (e.g., Figure 7.3 and Figure 7.4).
   - *Local probes:* vertical velocity profiles at different probe locations (e.g., Figure 7.7).

7. **Compute statistics.** Evaluate mean (e.g., Figure 7.5 and Figure 7.6), variance (e.g., Figure 7.8), and confidence intervals (e.g., Figure 7.7) using Eq. (3.13) and Eq. (3.14) or the gPC expansion coefficients.

7.3 Simulation case setup and parameter configuration

To represent input uncertainty in the inflow boundary condition, both the reference wind speed and direction are modeled with artificial measurement perturbation, respectively

$$\begin{aligned} \widehat{U}_H(t, \zeta_1) &= U_H(t)\zeta_1, & \zeta_1 &\sim \mathcal{N}\big(1,\, 0.1^2\big)\,, \\ \widehat{\Theta}(t, \zeta_2) &= \Theta(t) + \zeta_2, & \zeta_2 &\sim \mathcal{N}\big(0,\, 6^2\big)\,, \end{aligned}$$

where $U_H(t)$ and $\Theta(t)$ are the recorded wind speed and direction (the latter is given in radians).

In summary, we prescribe a logarithmic wind profile at horizontal boundaries

$$\hat{\boldsymbol{y}}(\boldsymbol{x}, t, \boldsymbol{Z}) = \hat{\boldsymbol{y}}(\boldsymbol{x}, t, \boldsymbol{\zeta}) = \frac{\widehat{U}_H(t, \zeta_1)}{\ln\left(\dfrac{H + z_0}{z_0}\right)} \ln\left(\frac{h + z_0}{z_0}\right) \begin{pmatrix} \cos\big(\widehat{\Theta}(t, \zeta_2)\big) \\ \sin\big(\widehat{\Theta}(t, \zeta_2)\big) \\ 0 \end{pmatrix}, \tag{7.9}$$

where the roughness length of the surface is $z_0 = 0.1$[m], H[m] is the reference height and $h = |z|$ is the vertical coordinate in meters above ground, where $\boldsymbol{x} = (x, y, z)^{\mathrm{T}} \in \mathbb{R}^3$.

Figure 7.2a shows the hourly reference wind speed $U_H(t)$ together with wind direction, recorded at the Reutlingen station (Germany, $48.4914°\,\mathrm{N}$, $9.2043°\,\mathrm{E}$) from 2024-11-07 20:00 to 2024-11-09 19:00 (local time, CET). Figure 7.2b displays the same wind speed data along with the shaded 95% confidence interval (CI) band. Our simulations use the 48-hour measurement shown in Figure 7.2.

The simulation was performed using a dense-grid quadrature scheme with a polynomial order of $N = 5$ and 11 quadrature points per stochastic dimension, resulting in a total of $N_q = 121$ collocation points (i.e. samples). The physical and numerical parameters of the case are summarized in Table 7.1. The simulation domain is configured as a circular disk region (with radius $R = 270$ [m] and height $H = 40$ [m]. Two buildings from the city of Reutlingen are placed in the center of the domain.

Table 7.1: Physical and numerical parameters.

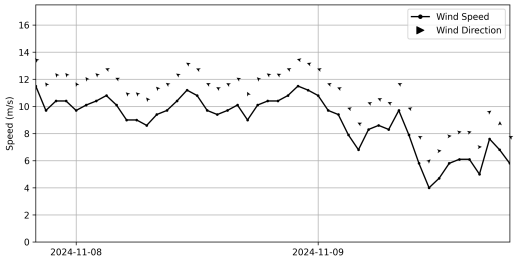
$\triangle x$	$\triangle t$	Ma	Re	T	#cells
1 [m]	0.00111 [s]	0.034641	1.8×10^6	48 [h]	10.27×10^6

All simulations were executed on the HoreKa supercomputer on CPU-only compute nodes equipped with two *Intel Xeon Platinum 8368* processors, providing 76 physical cores (152 hardware threads) per node at 2.40 GHz. At the job level, 42 samples were launched concurrently across 168 nodes, with each sample run in parallel on 304 MPI ranks. The wall-clock time per simulation was 5.4×10^4 s (about 15.0 hours), corresponding to approximately 4560 CPU core-hours per sample. In total, 121 samples were completed within a turnaround time of about 2.1 days, yielding a total computational cost of about 5.5×10^5 CPU core-hours, followed by an additional 3.6×10^3 s (one hour) for postprocessing. Theoretically, with 484 dedicated nodes, the sampling stage alone could be finished in 5.4×10^4 s (15 hours).

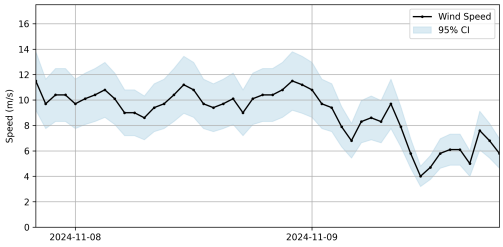
In the SC LBM framework, uncertainty is propagated by assigning each collocation node j a realization of the random multipliers $\zeta_1^{(j)}$ and $\zeta_2^{(j)}$, drawn from $\mathcal{N}(1,\, 0.1^2)$ and $\mathcal{N}(0,\, 6^2)$.

7.4 Uncertain data-assimilated simulation results

To illustrate the variability across individual samples, we highlight two representative cases from the stochastic collocation ensemble. Figure 7.3 shows results for sample 10 (out of 121), which lies farthest from the expected mean and exhibits a pronounced deviation in terms of instantaneous velocity field structures. In contrast, Figure 7.4 presents sample 60, which lies closest to the expected mean. For both cases, the time-dependent velocity magnitudes are shown at three selected instants (2024-11-07 22:00:00 CET, 2024-11-08 10:00:00 CET, and 2024-11-09 08:00:00 CET) alongside



(a) Wind speed and wind direction measurements



(b) Wind speed measurements with 95% confidence interval

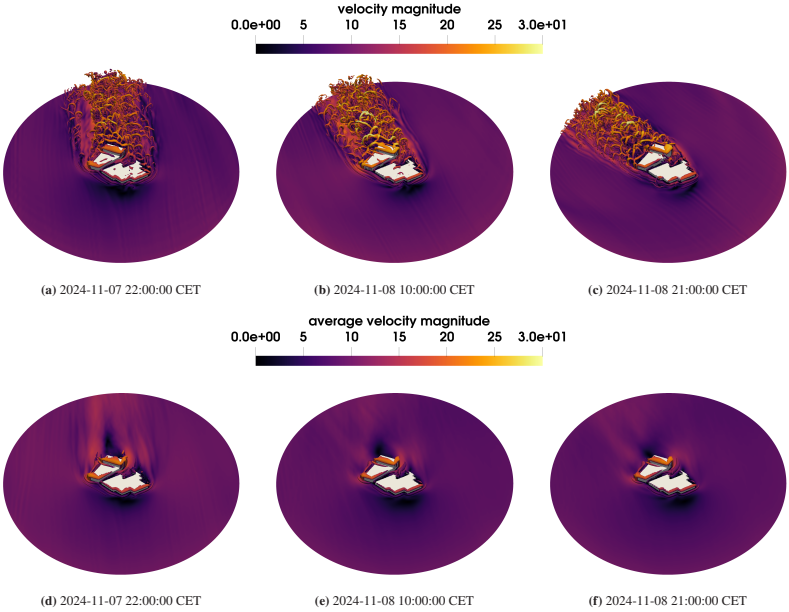
Figure 7.2: Hourly reference wind speed $U_H(t)$ recorded at the Reutlingen station (Germany, $48.4914^\circ\mathrm{N}$, $9.2043^\circ\mathrm{E}$) from 2024-11-07 20:00 to 2024-11-09 19:00 (local time, CET), obtained from `https://meteostat.net/`. Subfigure (a) shows wind speed and direction; subfigure (b) displays wind speed with the shaded 95% confidence interval.

their corresponding time-averaged fields. The isocontours of the Q-criterion ($Q = 1$) reveal coherent vortical structures, where differences between sample 10 and sample 60 highlight the spread of local flow features induced by input uncertainty.

To further quantify the uncertainty in the simulated flow field, we perform postprocessing over all 121 samples to evaluate the mean and standard deviation of the velocity magnitude.

Figure 7.5 shows the statistics of the time-dependent velocity magnitude at three representative instants (2024-11-07 22:00:00 CET, 2024-11-08 10:00:00 CET, and 2024-11-09 08:00:00 CET). The mean fields (a–c) capture the predominant flow structures around the buildings, whereas the corresponding standard deviation fields (d–f) highlight regions of elevated variability, particularly in the wakes and shear layers.

Complementing this, Figure 7.6 presents the mean and standard deviation of the time-averaged velocity magnitude. Here, the mean field emphasizes the dominant flow patterns, while the standard deviation field identifies persistent regions of high variability. Together, these statistical characterizations provide a comprehensive picture of the flow, offering insight into the reliability of the simulations and pinpointing regions most sensitive to input uncertainty.



(a) 2024-11-07 22:00:00 CET **(b)** 2024-11-08 10:00:00 CET **(c)** 2024-11-08 21:00:00 CET

(d) 2024-11-07 22:00:00 CET **(e)** 2024-11-08 10:00:00 CET **(f)** 2024-11-08 21:00:00 CET

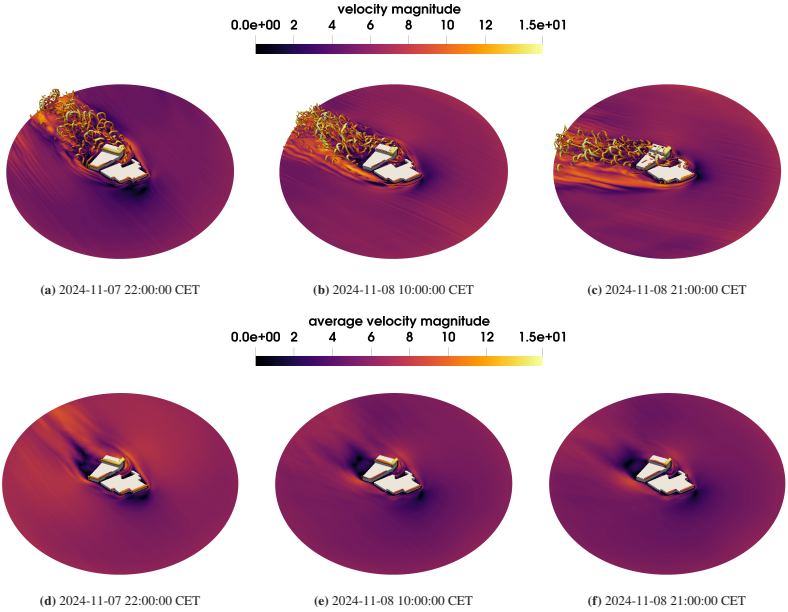
Figure 7.3: Simulated (SC LBM) single sample (number 10 out of 121) time-dependent velocity field magnitude at (a) 2024-11-07 22:00:00 CET, (b) 2024-11-08 10:00:00 CET, (c) 2024-11-09 08:00:00, and time-averaged velocity field magnitude at (d) 2024-11-07 22:00:00 CET, (e) 2024-11-08 10:00:00 CET, (f) 2024-11-09 08:00:00, respectively. Isocontours are colored in velocity magnitude and represent the Q-criterion at $Q = 1$, of the local-in-time results (a,b,c) and the time-averaged time averaged results (d,e,f), respectively.

To further investigate the vertical wind profile and its uncertainty at selected positions in the urban domain, we place three virtual probes at the coordinates listed in Table 7.2 (see also pink points in Figure 7.5), representing distinct flow regions.

Table 7.2: Probe locations and associated flow regions in the urban domain (see also black line markers in Figure 7.5b). The computed vertical velocity magnitude profiles at these probe locations are shown in Figure 7.7.

Probe specifier	**Coordinates** (x, y, z)
P1	$(30, 87, z),\ z \in [0, 40]$
P2	$(78, 60, z),\ z \in [0, 40]$
P3	$(130, 15, z),\ z \in [0, 40]$

Figure 7.7 shows the vertical distribution of the velocity magnitude at each probe, including the mean, 95% CI, and individual sample realizations obtained from the SC LBM simulation. The width of the



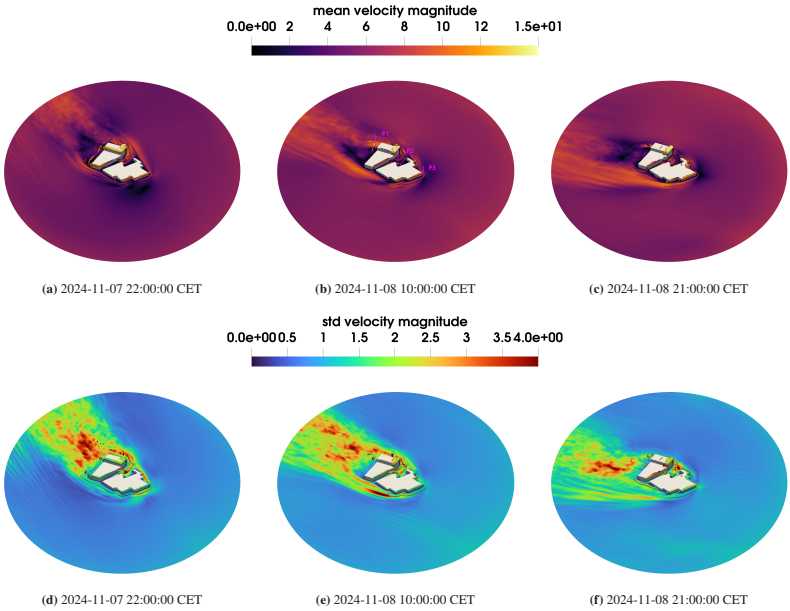
(a) 2024-11-07 22:00:00 CET **(b)** 2024-11-08 10:00:00 CET **(c)** 2024-11-08 21:00:00 CET

(d) 2024-11-07 22:00:00 CET **(e)** 2024-11-08 10:00:00 CET **(f)** 2024-11-08 21:00:00 CET

Figure 7.4: Simulated (SC LBM) single sample (number 60 out of 121, nearest to expected mean) time-dependent velocity field magnitude at (a) 2024-11-07 22:00:00 CET, (b) 2024-11-08 10:00:00 CET, (c) 2024-11-09 08:00:00, and time-averaged velocity field magnitude at (d) 2024-11-07 22:00:00 CET, (e) 2024-11-08 10:00:00 CET, (f) 2024-11-09 08:00:00, respectively. Isocontours are colored in velocity magnitude and represent the Q-criterion at $Q = 1$, of the local-in-time results (a,b,c) and the time-averaged time averaged results (d,e,f), respectively.

confidence bands varies significantly between locations, indicating different levels of flow variability depending on the local building-induced effects and turbulent mixing intensity.

Figure 7.8 shows a horizontal slice of the urban domain at a height of $2\,\mathrm{m}$ above ground level. The background colormap represents the mean velocity magnitude (time-averaged field in Figure 7.8a and time-dependent in Figure 7.8b), while two iso-contours highlight regions of reduced variability: the white contour corresponds to $\sigma_u \approx 0.8$ and the green contour to $\sigma_u \approx 0.4$, which refer to 50% and 25% of the absolute standard deviation, respectively. In this slice, the standard deviation ranges from 0 to 1.63, decreasing outside the white contour and further increasing inside the green contour. The other regions of enhanced fluctuations are primarily associated with wake zones, shear layers, and open channels between buildings. The probe locations P1–P3 are also indicated as pink dots. Notably, P1 is located in a flow region with an expected mean velocity magnitude smaller than 50% of the maximum and a standard deviation that is below 25%. Thus, based on our simulation results, we are also able to isolate flow regions that have critical features, e.g. wind velocity magnitude, up to a specific certainty.



(a) 2024-11-07 22:00:00 CET **(b)** 2024-11-08 10:00:00 CET **(c)** 2024-11-08 21:00:00 CET

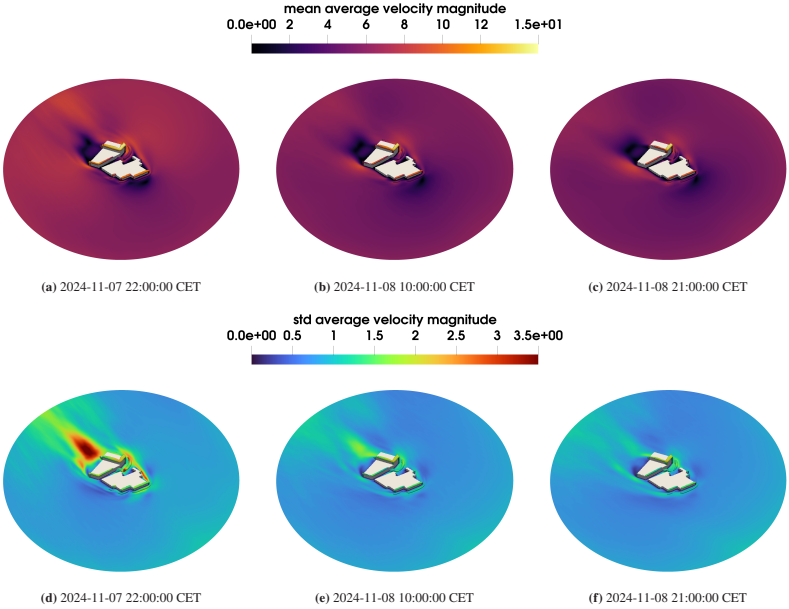
(d) 2024-11-07 22:00:00 CET **(e)** 2024-11-08 10:00:00 CET **(f)** 2024-11-08 21:00:00 CET

Figure 7.5: Simulated (SC LBM) mean of the time-dependent velocity field magnitude at (a) 2024-11-07 22:00:00 CET, (b) 2024-11-08 10:00:00 CET, (c) 2024-11-09 08:00:00, and std of the time-dependent velocity field magnitude at (d) 2024-11-07 22:00:00 CET, (e) 2024-11-08 10:00:00 CET, (f) 2024-11-09 08:00:00, respectively, both computed from 121 samples. Isocontours are computed from mean velocity magnitudes and represent the Q-criterion at $Q = 1$. The isocontours are respectively colored in mean (a,b,c) and std (d,e,f) of the time-dependent velocity magnitude. The probe probe location lines P1, P2, P3 are shown in pink (b), further specified in Table 7.2, and evaluated in Figure 7.7.

7.5 Conclusion

We presented an uncertain data assimilation workflow based on OpenLB-UQ [99], which directly injects measurement uncertainty into boundary data and propagates it through an SC LBM simulation of the urban wind flow around an isolated real building geometry in the city of Reutlingen (Germany, $48.4914°\mathrm{N}, 9.2043°\mathrm{E}$). The resulting stochastic boundary condition is mapped to a dense quadrature grid in the gPC space, and OpenLB-UQ is used to compute an ensemble of LBM solutions at the collocation nodes. From these, we recover spatio-temporal statistics of wind speed and velocity magnitude throughout the urban domain, including diagnostics tailored to urban applications. In summary, by modeling inflow uncertainty as a relative perturbation and propagating it through a non-intrusive SC LBM pipeline, we achieved the following:

- An efficiently scalable first application case of a combined UQ LES framework for urban wind flow simulations in real geometries.



(a) 2024-11-07 22:00:00 CET **(b)** 2024-11-08 10:00:00 CET **(c)** 2024-11-08 21:00:00 CET

(d) 2024-11-07 22:00:00 CET **(e)** 2024-11-08 10:00:00 CET **(f)** 2024-11-08 21:00:00 CET

Figure 7.6: Simulated (SC LBM) mean of the time-averaged velocity field magnitude at (a) 2024-11-07 22:00:00 CET, (b) 2024-11-08 10:00:00 CET, (c) 2024-11-09 08:00:00, and std of the time-averaged velocity field magnitude at (d) 2024-11-07 22:00:00 CET, (e) 2024-11-08 10:00:00 CET, (f) 2024-11-09 08:00:00, respectively, both computed from 121 samples. Isocontours are computed from mean velocity magnitudes and represent the Q-criterion at $Q = 1$. The isocontours are respectively colored in mean (a,b,c) and std (d,e,f) of the time-averaged velocity magnitude.

- Space-time-dependent estimates of measurement data-assimilated mean and standard deviation of the velocity field in a real urban environment;
- Space-time-dependent resolved uncertainty diagnostics highlighting flow-sensitive zones (e.g., wakes, shear layers);
- Confidence intervals at selected monitoring probes.

Conclusively, our workflow respects the statistical nature of measurement-driven boundary data and provides interpretable uncertainty maps for further use in downstream applications such as urban design and planning. Promising future research directions should extend our methodology to handle multi-modal distributions, incorporate machine learning surrogate models (e.g., variational autoencoder), and enable real-time integration with sensor data streams for live urban wind assessment.

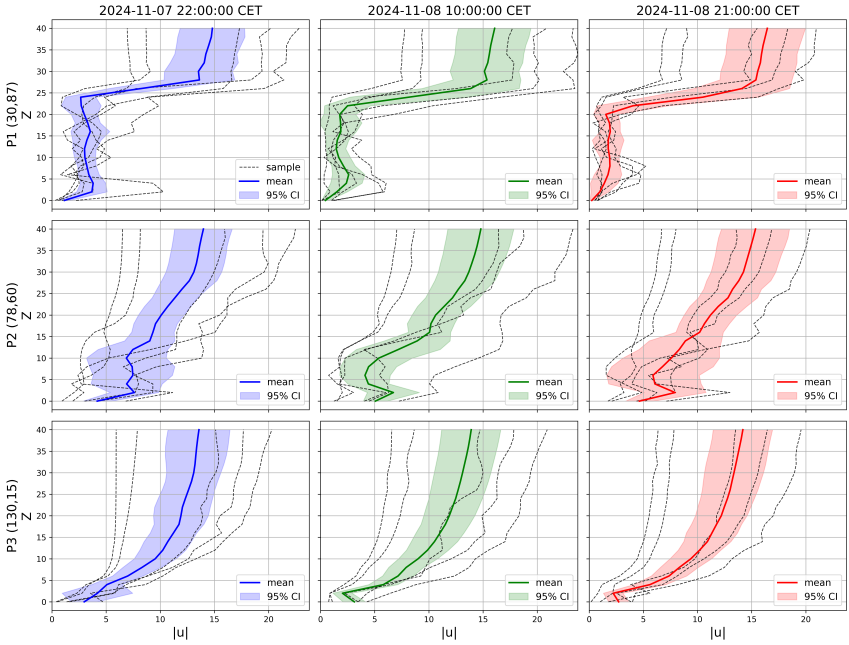
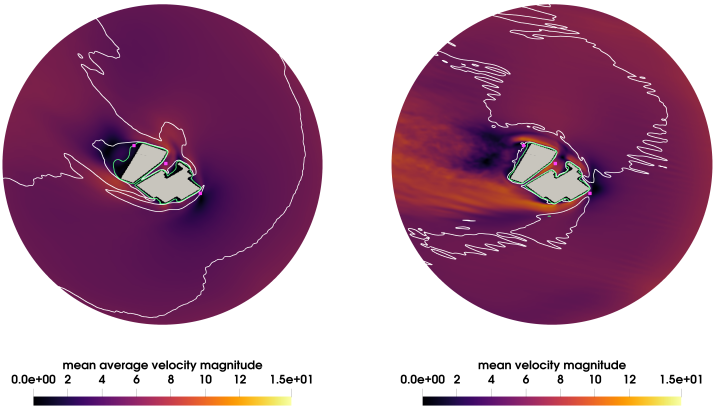


Figure 7.7: Time-local vertical velocity magnitude profiles at three probe positions: P1 $(30, 87)$ at the outer corner of the left building (shear layer), P2 $(78, 60)$ inside the channel between the buildings (channel), and P3 $(130, 15)$ at the outer corner of the right building (shear layer) (see Table 7.2 and Figure 7.5b). Each subplot corresponds to a specific time: 2024-11-07 22:00:00 CET, 2024-11-08 10:00:00 CET, and 2024-11-08 21:00:00 CET, respectively. Solid lines show the mean velocity profile, shaded regions indicate the 95% confidence intervals, and dashed lines represent five exemplary individual samples (out of 121) from the stochastic collocation LBM simulation. This figure visualizes one selected time step per subplot.



(a) mean average at 2024-11-08 21:00:00 CET

(b) mean at 2024-11-08 21:00:00 CET

Figure 7.8: Horizontal slice of the urban domain at a height of $2\,\mathrm{m}$ above ground, showing the mean velocity magnitude field (background colormap), for (a) time-averaged and (b) time-dependent results, with overlaid iso-contours of the velocity standard deviation at $\sigma_u \approx 0.8$ (50% in white) and $\sigma_u \approx 0.4$ (25% in green). The standard deviation in this slice ranges from 0 to ≈ 1.63, further decreases inside the isocounters. The probe locations P1–P3 are visualized as pink dots and further specified in Table 7.2.

8 Conclusions

The main result of this dissertation is the development of a comprehensive framework for UQ in LBM simulations of incompressible flows (see Chapters 4–7). Two complementary approaches are introduced. First, the intrusive SG LBM reformulates the LBE using gPC, achieving spectral convergence and up to six-fold efficiency gains over MCS. Second, the non-intrusive OpenLB-UQ extension integrates MC, QMC, and SC into the OpenLB library, enabling automated sampling, parallel execution, and statistical postprocessing.

Validation on canonical benchmarks confirms accuracy and scalability, while the application to urban wind flow simulations with uncertain inflow demonstrates practical relevance. Thus, the work establishes both a novel intrusive methodology and an open-source software framework that together advance efficient, uncertainty-aware CFD. Specific results are summarized, contributions are assessed with respect to the stated objectives, and directions for future research are outlined.

8.1 Summary

In Section 2.1, the deterministic incompressible NSE were extended into a stochastic framework by introducing random variables into viscosity, boundary conditions, and initial data, thereby providing the mathematical basis for analyzing how uncertainties propagate through fluid systems. Building on this, Section 2.2 recalled the kinetic theory foundations of the lattice Boltzmann method, from the Boltzmann–BGK equation to its discrete D2Q9 and D3Q19 models, with consistency to the incompressible NSE ensured through the CE expansion. A stochastic extension of the LBE was then formulated, incorporating random inputs and uncertain relaxation times while preserving the collision–streaming structure, which forms the basis for both intrusive and non-intrusive UQ strategies.

On this foundation, Section 3.1 introduced the general framework of UQ, distinguishing between aleatoric and epistemic sources and focusing on forward propagation. Section 3.2 addressed the modeling of uncertain inputs via probability distributions, noting both the advantages of orthogonal polynomial bases and limitations such as the neglect of correlations. Finally, Section 3.3 analyzed forward propagation methods: MCS as a scalable baseline with guaranteed convergence, QMC as an improvement through low-discrepancy sequences, and gPC expansions forming the basis for both non-intrusive SC and the intrusive SG projection. Special emphasis was given to sparse quadrature rules such as Genz–Keister and Clenshaw–Curtis, which mitigate the curse of dimensionality. These

developments together established the theoretical foundation for the intrusive SG LBM in Chapter 4 and the non-intrusive OpenLB-UQ framework in Chapter 5.

In Section 4.1, the SG formulation of the LBM was developed by expanding the distribution functions into a gPC basis. The collision and streaming operators were projected accordingly, yielding a coupled system for the polynomial coefficients. Particular emphasis was placed on stochastic relaxation times and equilibrium populations, where tensor contractions and quadrature rules ensured accurate coefficient couplings. A Chapman–Enskog expansion confirmed formal consistency with the incompressible NSE, marking the first such analysis for an SG LBM. Boundary conditions such as no-slip and moving walls were extended into the SG framework in Section 4.2, while Section 4.3 addressed implementation aspects, demonstrating that the standard LBM structure is preserved despite the increased dimensionality of the coefficient system. Numerical examples in Section 4.4, including Taylor–Green vortex, lid–driven cavity, and isentropic vortex convection, showed spectral convergence and five- to six-fold efficiency gains over MC LBM. Together, these results establish the SG LBM as the first fully intrusive UQ method consistent with the LBM framework.

Complementing this intrusive approach, Section 5.1 introduced the software architecture of the OpenLB-UQ extension, designed for modularity and seamless integration into the existing OpenLB code base. Section 5.2 detailed the core class design—including probability distributions, polynomial bases, sampling strategies, and quadrature rules, which enable both MCS and SC methods. A solver orchestration and data management layer was developed to automate sample generation, execution, and statistical postprocessing while preserving OpenLB's high-performance parallel infrastructure. Finally, Section 5.3 outlined the runtime workflow, illustrating how uncertainty-aware simulations can be launched with minimal changes to user applications. Overall, these developments established a practical and extensible framework that operationalizes the theoretical methods of Chapter 3 and supports scalable non-intrusive UQ workflows in LBM simulations.

In Chapter 6, the non-intrusive UQ framework was validated on canonical benchmark problems. Flow past a circular cylinder with uncertain inlet velocity assessed statistical accuracy in predicting drag and lift coefficients, while Taylor–Green vortex cases with uncertain viscosity and multi-dimensional uncertain initial conditions tested the statistical convergence of MCS, QMC, and SC methods. Section 6.2 further evaluated the parallel performance of the OpenLB-UQ module, demonstrating scalability and efficiency on distributed-memory architectures. These studies confirmed the robustness of non-intrusive approaches, highlighted the spectral convergence of SC, and provided practical guidance for selecting UQ methods depending on problem dimensionality and computational budget.

Building on these benchmarks, Chapter 7 applied the OpenLB-UQ workflow [99] to an urban wind flow simulation around a real urban geometry in Reutlingen, Germany ($48.4914°\,\mathrm{N}, 9.2043°\,\mathrm{E}$). Measurement uncertainty was injected into the inflow boundary and propagated through a non-intrusive SC LBM simulation. From the resulting ensemble, spatio-temporal statistics of velocity fields were obtained, including mean, standard deviation, and flow-sensitive diagnostics such as wakes and shear layers. This first scalable UQ LES application in real geometry demonstrates how

measurement-driven uncertainty can be translated into interpretable maps for urban planning and environmental assessment.

Conclusively, this work established both intrusive and non-intrusive uncertainty quantification strategies consistently within the lattice Boltzmann framework. The stochastic extension of the incompressible NSE and the LBM provided the theoretical foundation, while the intrusive SG LBM demonstrated spectral convergence and significant efficiency gains through a direct reformulation of the lattice Boltzmann equation. In parallel, the development of the OpenLB-UQ module extended the open-source solver with scalable non-intrusive methods, automated sampling, and parallel post-processing. Validation on canonical benchmarks and application to a building-resolved urban wind flow scenario confirmed robustness, accuracy, and practical relevance. In summary, the combination of intrusive methodology, open-source software implementation, systematic benchmark testing, and real-world application yields a coherent UQ framework that advances efficient, uncertainty-aware CFD simulations within the LBM paradigm.

While the intrusive SG LBM and the non-intrusive OpenLB-UQ framework establish complementary foundations for uncertainty-aware LBM simulation, both approaches face open challenges. Issues such as the scalability of intrusive formulations to higher dimensions and the efficiency of non-intrusive methods in complex settings motivate further research. These aspects are addressed in the following section on Limitations and Outlook.

8.2 Limitations and Outlook

The present dissertation establishes intrusive and non-intrusive frameworks for uncertainty quantification in the LBM and demonstrates their applicability to canonical benchmarks and urban wind flows. While the main objectives have been reached, several limitations remain, which at the same time motivate promising directions for future research:

- **Scalability of SG LBM and SC LBM.** The intrusive SG LBM achieves spectral convergence and efficiency gains in low-dimensional settings, yet the tensor-product structure of gPC leads to a rapid growth of modes. This restricts its applicability for high-dimensional or strongly nonlinear uncertainties. Adaptive truncation or sparse representations should be explored to mitigate this issue.
- **Stability of intrusive schemes.** The stochastic collision operator shows sensitivity in turbulent or highly transient regimes. Algorithmic refinements and stability analyses are required to extend SG LBM toward complex unsteady flows.
- **Computational cost of non-intrusive methods.** Although the `OpenLB-UQ` framework modularly integrates MC, QMC, and SC, the computational burden remains high for long time series

or high-dimensional uncertainties. Techniques such as multi-level or adaptive sparse-grid collocation would significantly reduce cost.

- **Representation of uncertainties.** The current focus is on aleatoric uncertainties in boundary data and parameters. Epistemic sources, such as incomplete geometry or model-form errors, are not yet systematically addressed. Coupling with Bayesian inference and advanced data assimilation offers a natural path forward.
- **Time-dependent and correlated inputs.** Stochastic inflows have been treated independently, whereas realistic applications demand correlated and non-stationary models. Approaches such as Gaussian processes or Karhunen–Loève expansions should be incorporated to capture temporal and spatial correlations.
- **Computational platforms.** While the present framework benefits from the parallelizability of LBM on HPC clusters, only the non-intrusive core can exploit heterogeneous architectures efficiently (e.g. GPU-accelerated exascale systems). The porting of the complete UQ framework including the postprocessing pipeline to accelerated systems remains an essential step to make large-scale uncertainty-aware simulations feasible.

In summary, further research into adaptive algorithms, high-dimensional representations, data-driven inference, and exascale deployment will broaden the scope and impact of the methods developed in this dissertation. Together, these directions promise to evolve intrusive and non-intrusive UQ for LBM into a robust platform for uncertainty-aware CFD in both science and engineering applications.

Acronyms and symbols

Some of the frequently used acronyms and symbols are summarized below. The list raises no claim to completeness. For the purpose of readability, symbols are renamed occasionally. In this case, the definition or notation is explicitly stated within the respective section and is mostly used therein.

Some of the frequently used acronyms and symbols are summarized below. For readability, symbols are occasionally renamed; in such cases the definition/notation is stated where it is used.

Frequently used acronyms

Acronym	Definition	Page
UQ	uncertainty quantification	p. 1
CFD	computational fluid dynamics	p. 1
MCS	Monte Carlo sampling	p. 1
QMC	quasi Monte Carlo	p. 1
MLMC	multi-level Monte Carlo	p. 1
SC	stochastic collocation	p. 1
gPC	generalized polynomial chaos	p. 1
SG	stochastic Galerkin	p. 1
NSE	Navier–Stokes equations	p. 1
RANS	Reynolds-averaged Navier–Stokes	p. 1
LBM	lattice Boltzmann method	p. 1
TGV	Taylor–Green vortex	p. 2
LDC	lid-driven cavity	p. 2
BE	Boltzmann equation	p. 8
BGK	Bhatnagar–Gross–Krook	p. 9
LBE	lattice Boltzmann equation	p. 9
CE	Chapman–Enskog	p. 10
QoIs	quantities of interest	p. 15

Acronym	Definition	Page
EOC	experimental order of convergence	p. 39
IVC	isentropic vortex convection	p. 27
CI	confidence interval	p. 87

Frequently used symbol

symbol	Definition	Page
$\boldsymbol{u}\colon \mathcal{X} \times \mathcal{I} \to \mathbb{R}^{d_x}$	velocity vector	p. 7
ρ	density	p. 7
ν	kinematic viscosity	p. 7
$\boldsymbol{Z} = (Z_1, ..., Z_d) \in \mathcal{Z} \subseteq \mathbb{R}^{dZ}$	a vector of independent random variables	p. 8
$(\Xi, \mathcal{F}, \mathbb{P})$	probability space	p. 8
$\boldsymbol{x} \in \mathcal{X} \subseteq \mathbb{R}^{d_x}$	spatial position	p. 8
$f\colon \mathcal{X} \times \mathbb{R}^{d_\xi} \times \mathcal{I} \to \mathbb{R}_{\geq 0}$	single-particle distribution function	p. 8
$t \in \mathcal{I} \subseteq \mathbb{R}_{>0}$	time	p. 8
$\Omega(f)$	collision operator	p. 9
τ	relaxation time	p. 9
c_s	lattice speed of sound	p. 9
$\triangle t$	time step	p. 9
$\mathcal{M} : \mathbb{R}^{dZ} \to \mathbb{R}^m$	deterministic model	p. 16
N_q	quadrature points number	p. 18
$\mathbb{E}(\cdot)$	expectation	p. 18
d_Z	dimension of uncertainty	p. 18
$\Phi_{\alpha_i}^{(i)}$	univariate orthonormal polynomial	p. 20
$\Phi_{\boldsymbol{\alpha}}$	multivariate orthonormal polynomial	p. 20
$\hat{\mathcal{M}}_\alpha$	gPC projection coefficients	p. 20
N	polynomial coefficients	p. 20
Re	Reynolds number	p. 36
ζ	uncertain variable	p. 37
δ	relative error	p. 42

List of Figures

List of Tables

Publications

Parts of this work have already been published in journals and software releases. The references are given in the main body where applicable, and own publications are also summarized below. First authorship indicates a major contribution to the work.

Peer-reviewed publications

i. **M. Zhong**, T. Xiao, M. J. Krause, M. Frank, S. Simonis. A stochastic Galerkin lattice Boltzmann method for incompressible fluid flows with uncertainties. *Journal of Computational Physics*, 517:113344, 2024. DOI: 10.1016/j.jcp.2024.113344. Reference [98].

Preprints

ii. **M. Zhong**, A. Kummerländer, S. Ito, M. J. Krause, M. Frank, S. Simonis. OpenLB-UQ: An Uncertainty Quantification Framework for Incompressible Fluid Flow Simulations. *arXiv preprint*, 2025. DOI: 10.48550/arXiv.2508.13867. Reference [99].

iii. **M. Zhong**, D. Teutscher, A. Kummerländer, M. J. Krause, M. Frank, S. Simonis. Uncertain data assimilation for urban wind flow simulations with OpenLB-UQ. *arXiv preprint*, 2025. DOI: 10.48550/arXiv.2508.18202. Reference [100].

Conference talks

iv. **M. Zhong**, M. J. Krause, M. Frank, and S. Simonis. Non-intrusive uncertainty quantification technique on Lattice Boltzmann method, "17th Asian Congress of Fluid Mechanics." In: *ACFM 2023 – 17th Asian Congress of Fluid Mechanics*. Beijing, China, 2023.

v. **M. Zhong**, M. J. Krause, M. Frank, and S. Simonis. A stochastic Galerkin lattice Boltzmann method for incompressible fluid flows with uncertainties, "The 20th International Conference

for Mesoscopic Methods in Engineering and Science." In: *ICMMES 2024 – 20th International Conference for Mesoscopic Methods in Engineering and Science*. Hammamet, Tunis, 2024.

vi. **M. Zhong**, M. J. Krause, M. Frank, and S. Simonis. Non-Intrusive and Intrusive Uncertainty Quantification for Lattice Boltzmann Method, "The 21st International Conference for Mesoscopic Methods in Engineering and Science." In: *ICMMES 2025 – 21st International Conference for Mesoscopic Methods in Engineering and Science*. Wuhan, China, 2025.

Software releases

vii. A. Kummerländer, S. Avis, H. Kusumaatmaja, F. Bukreev, D. Dapelo, S. Großmann, N. Hafen, C. Holeksa, A. Husfeldt, J. Jeßberger, L. Kronberg, J. Marquardt, J. Mödl, J. Nguyen, T. Pertzel, S. Simonis, L. Springmann, N. Suntoyo, D. Teutscher, **M. Zhong**, and M. J. Krause. OpenLB Release 1.5: Open Source Lattice Boltzmann Code. Version 1.5. 2022. DOI: 10.5281/zenodo.6469606.

viii. A. Kummerländer, T. Bingert, F. Bukreev, L. E. Czelusniak, D. Dapelo, S. Englert, N. Hafen, M. Heinzelmann, S. Ito, J. Jeßberger, F. Kaiser, E. Kummer, H. Kusumaatmaja, J. E. Marquardt, M. Rennick, T. Pertzel, F. Prinz, M. Sadric, M. Schecher, S. Simonis, P. Sitter, D. Teutscher, **M. Zhong**, and M. J. Krause. OpenLB Release 1.7: Open Source Lattice Boltzmann Code. 2024. DOI: 10.5281/zenodo.10684609.

ix. A. Kummerländer, T. Bingert, F. Bukreev, L. E. Czelusniak, D. Dapelo, S. Englert, N. Hafen, M. Heinzelmann, S. Ito, J. Jeßberger, F. Kaiser, E. Kummer, H. Kusumaatmaja, J. E. Marquardt, M. Rennick, T. Pertzel, F. Prinz, M. Sadric, M. Schecher, S. Simonis, P. Sitter, D. Teutscher, **M. Zhong**, and M. J. Krause. OpenLB Release 1.7: Open Source Lattice Boltzmann Code. DOI: 10.5281/zenodo.10684609.

x. A. Kummerländer, T. Bingert, F. Bukreev, L. E. Czelusniak, D. Dapelo, C. Gaul, N. Hafen, S. Ito, J. Jeßberger, D. Khazaeipoul, T. Krüger, H. Kusumaatmaja, J. E. Marquardt, A. Raeli, M. Rennick, F. Prinz, M. Schecher, A. Schneider, Y. Shimojima, S. Simonis, P. Sitter, P. Spelten, A. Tacques, D. Teutscher, **M. Zhong**, and M. J. Krause. OpenLB Release 1.8: Open Source Lattice Boltzmann Code. 2025. DOI: 10.5281/zenodo.15270117.

xi. A. Kummerländer, T. Bingert, F. Bukreev, L. E. Czelusniak, D. Dapelo, C. Gaul, N. Hafen, S. Ito, J. Jeßberger, D. Khazaeipoul, T. Krüger, H. Kusumaatmaja, J. E. Marquardt, A. Raeli, M. Rennick, F. Prinz, M. Schecher, A. Schneider, Y. Shimojima, S. Simonis, P. Sitter, P. Spelten, A. Tacques, D. Teutscher, **M. Zhong**, and M. J. Krause. OpenLB Release 1.8.1: Open Source Lattice Boltzmann Code. 2025. DOI: 10.5281/zenodo.15440776.

If not stated otherwise, the computer simulations used for the present results all are based upon, embedded in, or coupled to the open-source C++ library OpenLB [33]. The author contributed to the

Table 3: Selected parts of the author's contribution to OpenLB.

Keyword	Commit hash / Link	Section
`uncertain cavity2d`	olb/examples/uncertaintyQuantification/cavity2d	Section 4.4.3
`uncertain cavity3d`	olb/examples/uncertaintyQuantification/cavity3d	Section 4.4.3
`uncertain cylinder2d`	olb/examples/uncertaintyQuantification/cylinder2d	Section 6.1.1
`uncertain cylinder3d`	olb/examples/uncertaintyQuantification/cylinder3d	Section 6.1.1
`uncertain tgv2d`	olb/examples/uncertaintyQuantification/tgv2d	Section 6.1.3
`urban wind flow`	3953b0ece2caca78b10c0c8075119f0f23ce8fb1	Chapter 7
`UQclass (src/uq)`	olb/src/uq	Chapter 5

releases 1.5 [34], 1.7 [37], 1.8 [39] and 1.8.1 [40] of said library, as well as the User Guides for 1.7 [37] and 1.8 [41]. Some specific commit hashes for parts of the code (also unreleased) are listed in Table 3 together with keywords.

Bibliography

[1] R. Abgrall and S. Mishra. Uncertainty quantification for hyperbolic systems of conservation laws. In *Handbook of Numerical Analysis*, volume 18, pages 507–544. Elsevier, 2017. doi: 10.1016/bs.hna.2016.11.003.

[2] I. Babuška, F. Nobile, and R. Tempone. A stochastic collocation method for elliptic partial differential equations with random input data. *SIAM Journal on Numerical Analysis*, 45(3): 1005–1034, 2007. doi: 10.1137/100786356.

[3] P. Bansal. Numerical approximation of statistical solutions of the incompressible Navier-Stokes Equations. *arXiv*, preprint, 2021. doi: 10.48550/ARXIV.2107.06073.

[4] A. Barth, C. Schwab, and N. Zollinger. Multi-level Monte Carlo finite element method for elliptic PDEs with stochastic coefficients. *Numerische Mathematik*, 119(1):123–161, 2011. doi: 10.1007/s00211-011-0377-0.

[5] A. Beck, J. Dürrwächter, T. Kuhn, F. Meyer, C.-D. Munz, and C. Rohde. *hp*-Multilevel Monte Carlo Methods for Uncertainty Quantification of Compressible Navier–Stokes Equations. *SIAM Journal on Scientific Computing*, 42(4):B1067–B1091, 2020. doi: 10.1137/18M1210575.

[6] A. Beck, J. Dürrwächter, T. Kuhn, F. Meyer, C.-D. Munz, and C. Rohde. Uncertainty quantification in high performance computational fluid dynamics. In *High Performance Computing in Science and Engineering'19: Transactions of the High Performance Computing Center, Stuttgart (HLRS) 2019*, pages 355–371. Springer, 2021. doi: 10.1007/978-3-030-66792-4_24.

[7] P. L. Bhatnagar, E. P. Gross, and M. Krook. A Model for Collision Processes in Gases. I. Small Amplitude Processes in Charged and Neutral One-Component Systems. *Physical Review*, 94: 511–525, 1954. doi: 10.1103/PhysRev.94.511.

[8] M. Bouzidi, M. Firdaouss, and P. Lallemand. Momentum transfer of a Boltzmann-lattice fluid with boundaries. *Physics of Fluids*, 13(11):3452–3459, 2001. doi: 10.1063/1.1399290.

[9] F. Bukreev, S. Simonis, A. Kummerländer, J. Jeßberger, and M. J. Krause. Consistent lattice Boltzmann methods for the volume averaged Navier–Stokes equations. *Journal of Computational Physics*, 490:112301, 2023. doi: 10.1016/j.jcp.2023.112301.

[10] R. E. Caflisch. Monte Carlo and quasi Monte Carlo methods. *Acta Numerica*, 7:1–49, 1998. doi: 10.1017/S0962492900002804.

[11] S. Chapman and T. G. Cowling. *The mathematical theory of non-uniform gases: an account of the kinetic theory of viscosity, thermal conduction and diffusion in gases*. Cambridge University Press, 1990. doi: 10.1119/1.1942035.

[12] D. Dapelo, S. Simonis, M. J. Krause, and J. Bridgeman. Lattice-Boltzmann coupled models for advection–diffusion flow on a wide range of Péclet numbers. *Journal of Computational Science*, 51:101363, 2021. doi: 10.1016/j.jocs.2021.101363.

[13] P. Delgado and V. Kumar. A stochastic Galerkin approach to uncertainty quantification in poroelastic media. In *Fluids Engineering Division Summer Meeting*, volume 46247, page V01DT40A002. American Society of Mechanical Engineers, 2014. doi: 10.1115/FEDSM2014-21577.

[14] M. Diez, R. Broglia, D. Durante, A. Olivieri, E. Campana, and F. Stern. Validation of uncertainty quantification methods for high-fidelity CFD of ship response in irregular waves. In *55th AIAA Aerospace Sciences Meeting*, page 1655, 2017. doi: 10.2514/6.2017-1655.

[15] U. S. Fjordholm, K. Lye, and S. Mishra. Numerical approximation of statistical solutions of scalar conservation laws. *SIAM Journal on Numerical Analysis*, 56(5):2989–3009, 2018. doi: 10.1137/17M1154874.

[16] S. Fu, R. So, and W. W. F. Leung. Stochastic finite difference lattice Boltzmann method for steady incompressible viscous flows. *Journal of Computational Physics*, 229(17):6084–6103, 2010. doi: 10.1016/j.jcp.2010.04.041.

[17] C. Garcia-Sanchez, D. Philips, and C. Gorlé. Quantifying inflow uncertainties for CFD simulations of the flow in downtown Oklahoma City. *Building and Environment*, 78:118–129, 2014. doi: 10.1016/j.buildenv.2014.04.013.

[18] J. Garcke. Sparse grids in a nutshell. In *Sparse grids and applications*, pages 57–80. Springer, 2012. doi: 10.1007/978-3-642-31703-3_3.

[19] A. Genz and B. D. Keister. Fully symmetric interpolatory rules for multiple integrals over infinite regions with gaussian weight. *Journal of Computational and Applied Mathematics*, 71 (2):299–309, 1996. doi: 10.1016/0377-0427(95)00232-4.

[20] G. Geraci, M. S. Eldred, and G. Iaccarino. A multifidelity multilevel Monte Carlo method for uncertainty propagation in aerospace applications. In *19th AIAA Non-deterministic Approaches Conference*, page 1951, 2017. doi: 10.2514/6.2017-1951.

[21] U. Ghia, K. N. Ghia, and C. Shin. High-Re solutions for incompressible flow using the Navier-Stokes equations and a multigrid method. *Journal of Computational Physics*, 48(3):387–411, 1982. doi: 10.1016/0021-9991(82)90058-4.

[22] M. B. Giles. Multilevel Monte Carlo methods. *Acta Numerica*, 24:259–328, 2015. doi: 10.1017/S096249291500001X.

[23] D. Gottlieb and D. Xiu. Galerkin method for wave equations with uncertain coefficients. *Communications in Computational Physics*, 3(2):505–518, 2008. doi: 2008-CiCP-7864.

[24] Z. Guo and C. Shu. *Lattice Boltzmann method and its application in engineering*, volume 3. World Scientific, 2013. doi: 10.1142/8806.

[25] N. Hafen, A. Dittler, and M. J. Krause. Simulation of particulate matter structure detachment from surfaces of wall-flow filters applying lattice Boltzmann methods. *Computers & Fluids*, 239: 105381, 2022. doi: 10.1016/j.compfluid.2022.105381.

[26] M. Haussmann, F. Ries, J. B. Jeppener-Haltenhoff, Y. Li, M. Schmidt, C. Welch, L. Illmann, B. Böhm, H. Nirschl, M. J. Krause, et al. Evaluation of a near-wall-modeled large eddy lattice Boltzmann method for the analysis of complex flows relevant to IC engines. *Computation*, 8(2): 43, 2020. doi: 10.3390/computation8020043.

[27] M. Haussmann, P. Reinshaus, S. Simonis, H. Nirschl, and M. J. Krause. Fluid–Structure Interaction Simulation of a Coriolis Mass Flowmeter Using a Lattice Boltzmann Method. *Fluids*, 6(4):167, 2021. doi: 10.3390/fluids6040167.

[28] S. Hosder, R. Walters, and M. Balch. Efficient sampling for non-intrusive polynomial chaos applications with multiple uncertain input variables. In *48th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, page 1939, 2011. doi: 10.2514/6.2007-1939.

[29] J. Jacob, L. Merlier, F. Marlow, and P. Sagaut. Lattice Boltzmann method-based simulations of pollutant dispersion and urban physics. *Atmosphere*, 12(7):833, 2021. doi: 10.3390/atmos12070833.

[30] S. Jin, J.-G. Liu, and Z. Ma. Uniform spectral convergence of the stochastic Galerkin method for the linear transport equations with random inputs in diffusive regime and a micro–macro decomposition-based asymptotic-preserving method. *Research in the Mathematical Sciences*, 4 (1):15, 2017. doi: 10.1186/s40687-017-0105-1.

[31] M. Junk, A. Klar, and L.-S. Luo. Asymptotic analysis of the lattice Boltzmann equation. *Journal of Computational Physics*, 210(2):676–704, 2005. doi: 10.1016/j.jcp.2005.05.003.

[32] O. M. Jérôme Jacob and P. Sagaut. A new hybrid recursive regularised Bhatnagar–Gross–Krook collision model for Lattice Boltzmann method-based large eddy simulation. *Journal of Turbulence*, 19(11-12):1051–1076, 2018. doi: 10.1080/14685248.2018.1540879.

[33] M. Krause, A. Kummerländer, S. Avis, H. Kusumaatmaja, D. Dapelo, F. Klemens, M. Gaedtke, N. Hafen, A. Mink, R. Trunk, J. Marquardt, M. Maier, M. Haussmann, and S. Simonis. OpenLB–Open source lattice Boltzmann code. *Computers & Mathematics with Applications*, 81:258–288, 2021. doi: 10.1016/j.camwa.2020.04.033.

[34] A. Kummerländer, S. Avis, H. Kusumaatmaja, F. Bukreev, D. Dapelo, S. Großmann, N. Hafen, C. Holeksa, A. Husfeldt, J. Jeßberger, L. Kronberg, J. Marquardt, J. Mödl, J. Nguyen, T. Pertzel, S. Simonis, L. Springmann, N. Suntoyo, D. Teutscher, M. Zhong, and M. Krause. OpenLB Release 1.5: Open Source Lattice Boltzmann Code, Nov. 2022. URL `https://doi.org/10.5281/zenodo.6469606`.

[35] A. Kummerländer, M. Dorn, M. Frank, and M. J. Krause. Implicit propagation of directly addressed grids in lattice Boltzmann methods. *Concurrency and Computation: Practice and Experience*, e7509, 2022. doi: 10.1002/cpe.7509.

[36] A. Kummerländer, S. Avis, H. Kusumaatmaja, F. Bukreev, M. Crocoll, D. Dapelo, N. Hafen, S. Ito, J. Jeßberger, J. E. Marquardt, J. Mödl, T. Pertzel, F. Prinz, F. Raichle, M. Schecher, S. Simonis, D. Teutscher, and M. J. Krause. OpenLB Release 1.6: Open Source Lattice Boltzmann Code, Apr. 2023. URL `https://doi.org/10.5281/zenodo.7773497`.

[37] A. Kummerländer, T. Bingert, F. Bukreev, L. E. Czelusniak, D. Dapelo, S. Englert, N. Hafen, M. Heinzelmann, S. Ito, J. Jeßberger, F. Kaiser, E. Kummer, H. Kusumaatmaja, J. E. Marquardt, M. Rennick, T. Pertzel, F. Prinz, M. Sadric, M. Schecher, S. Simonis, P. Sitter, D. Teutscher, M. Zhong, and M. J. Krause. OpenLB User Guide 1.7, Aug. 2024. URL `https://doi.org/10.5281/zenodo.13293033`.

[38] A. Kummerländer, F. Bukreev, S. F. R. Berg, M. Dorn, and M. J. Krause. Advances in Computational Process Engineering using Lattice Boltzmann Methods on High Performance Computers. In W. E. Nagel, D. H. Kröner, and M. M. Resch, editors, *High Performance Computing in Science and Engineering '22*, pages 233–247, Cham, 2024. Springer Nature Switzerland. doi: 10.1007/978-3-031-46870-4_16.

[39] A. Kummerländer, T. Bingert, F. Bukreev, L. E. Czelusniak, D. Dapelo, C. Gaul, N. Hafen, S. Ito, J. Jeßberger, D. Khazaeipoul, T. Krüger, H. Kusumaatmaja, J. E. Marquardt, A. Raeli, M. Rennick, F. Prinz, M. Schecher, A. Schneider, Y. Shimojima, S. Simonis, P. Sitter, P. Spelten, A. Tacques, D. Teutscher, M. Zhong, and M. J. Krause. OpenLB Release 1.8: Open Source Lattice Boltzmann Code , Apr. 2025. URL `https://doi.org/10.5281/zenodo.15270117`.

[40] A. Kummerländer, T. Bingert, F. Bukreev, L. E. Czelusniak, D. Dapelo, C. Gaul, N. Hafen, S. Ito, J. Jeßberger, D. Khazaeipoul, T. Krüger, H. Kusumaatmaja, J. E. Marquardt, A. Raeli, M. Rennick, F. Prinz, M. Schecher, A. Schneider, Y. Shimojima, S. Simonis, P. Sitter, P. Spelten, A. Tacques, D. Teutscher, M. Zhong, and M. J. Krause. OpenLB Release 1.8.1: Open Source Lattice Boltzmann Code, May 2025. URL `https://doi.org/10.5281/zenodo.15440776`.

[41] A. Kummerländer, T. Bingert, F. Bukreev, L. E. Czelusniak, D. Dapelo, C. Gaul, N. Hafen, S. Ito, J. Jeßberger, D. Khazaeipoul, T. Krüger, H. Kusumaatmaja, J. E. Marquardt, A. Raeli, M. Rennick, F. Prinz, M. Schecher, A. Schneider, Y. Shimojima, S. Simonis, P. Sitter, P. Spelten, A. Tacques, D. Teutscher, M. Zhong, and M. J. Krause. OpenLB User Guide 1.8, Apr. 2025. URL `https://doi.org/10.13140/RG.2.2.26901.64486`.

[42] J. Kusch, R. G. McClarren, and M. Frank. Filtered stochastic Galerkin methods for hyperbolic equations. *Journal of Computational Physics*, 403:109073, 2020. doi: 10.1016/j.jcp.2019.109073.

[43] P. Lallemand and L.-S. Luo. Lattice Boltzmann method for moving boundaries. *Journal of Computational Physics*, 184(2):406–421, 2003. doi: 10.1016/S0021-9991(02)00022-0.

[44] P. Lallemand, L.-S. Luo, M. Krafczyk, and W.-A. Yong. The lattice Boltzmann method for nearly incompressible flows. *Journal of Computational Physics*, 431:109713, 2021. doi: 10.1016/j.jcp.2020.109713.

[45] S. Lanthaler, S. Mishra, and C. Parés-Pulido. Statistical solutions of the incompressible Euler equations. *Mathematical Models and Methods in Applied Sciences*, 31(02):223–292, 2021. doi: 10.1142/S0218202521500068.

[46] S. Lanthaler, S. Mishra, and C. Parés-Pulido. Statistical solutions of the incompressible Euler equations. *Mathematical Models and Methods in Applied Sciences*, 31(02):223–292, 2021. doi: 10.1142/S0218202521500068.

[47] J. Latt, B. Chopard, O. Malaspinas, M. Deville, and A. Michler. Straight velocity boundaries in the lattice Boltzmann method. *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics*, 77(5):056703, 2008. doi: 10.1103/PhysRevE.77.056703.

[48] F. Leonardi, S. Mishra, and C. Schwab. Numerical approximation of statistical solutions of planar, incompressible flows. *Mathematical Models and Methods in Applied Sciences*, 26(13):2471–2523, 2016. doi: 10.1142/S0218202516500597.

[49] D. Lucor, C. Enaux, H. Jourdren, and P. Sagaut. Stochastic design optimization: Application to reacting flows. *Computer Methods in Applied Mechanics and Engineering*, 196(49-52):5047–5062, 2007. doi: 10.1016/j.cma.2007.07.003.

[50] L. Mathelin, M. Y. Hussaini, and T. A. Zang. Stochastic approaches to uncertainty quantification in CFD simulations. *Numerical Algorithms*, 38(1):209–236, 2005. doi: 10.1007/BF02810624.

[51] J. W. McCullough and P. V. Coveney. Uncertainty quantification of the lattice Boltzmann method focussing on studies of human-scale vascular blood flow. *Scientific Reports*, 14(1):11317, 2024. doi: 10.1038/s41598-024-61708-w.

[52] A. Medaglia, L. Pareschi, and M. Zanella. Stochastic galerkin particle methods for kinetic equations of plasmas with uncertainties. *Journal of Computational Physics*, 479:112011, 2023. doi: 10.1016/j.jcp.2023.112011.

[53] R. Mei, L.-S. Luo, and W. Shyy. An accurate curved boundary treatment in the lattice Boltzmann method. *Journal of Computational Physics*, 155(2):307–330, 1999. doi: 10.1006/jcph.1999.6334.

[54] R. Mei, L.-S. Luo, P. Lallemand, and D. d'Humières. Consistent initial conditions for lattice Boltzmann simulations. *Computers & Fluids*, 35(8-9):855–862, 2006. doi: 10.1016/j.compfluid.2005.08.008.

[55] A. Mink, K. Schediwy, C. Posten, H. Nirschl, S. Simonis, and M. J. Krause. Comprehensive Computational Model for Coupled Fluid Flow, Mass Transfer, and Light Supply in Tubular Photobioreactors Equipped with Glass Sponges. *Energies*, 15(20), 2022. doi: 10.3390/en15207671.

[56] R. Molinaro, S. Lanthaler, B. Raonić, T. Rohner, V. Armegioiu, S. Simonis, D. Grund, Y. Ramic, Z. Y. Wan, F. Sha, S. Mishra, and L. Zepeda-Núñez. Generative AI for fast and accurate statistical computation of fluids. *arXiv preprint*, 2025. doi: 10.48550/arXiv.2409.18359.

[57] W. J. Morokoff and R. E. Caflisch. Quasi-Monte Carlo integration. *Journal of Computational Physics*, 122(2):218–230, 1995. doi: 10.1006/jcph.1995.1209.

[58] H. N. Najm. Uncertainty quantification and polynomial chaos techniques in computational fluid dynamics. *Annual Review of Fluid Mechanics*, 41:35–52, 2009. doi: 10.1146/annurev.fluid.010908.165248.

[59] T. A. Oliver and R. D. Moser. Bayesian uncertainty quantification applied to RANS turbulence models. In *Journal of Physics: Conference Series*, volume 318, number 4, page 042032. IOP Publishing, 2011. doi: 10.1088/1742-6596/318/4/042032.

[60] S. D. Ryan and C. J. Arisman. Uncertainty quantification of steady and transient source term estimation in an urban environment. *Environmental Fluid Mechanics*, 21(3):713–740, 2021. doi: 10.1007/s10652-021-09794-6.

[61] S. Simone, F. Montomoli, F. Martelli, K. S. Chana, I. Qureshi, and T. Povey. Analysis on the effect of a nonuniform inlet profile on heat transfer and fluid flow in turbine stages. *Journal of Turbomachinery*, 2012. doi: 10.1115/1.4003233.

[62] S. Simonis. *Lattice Boltzmann Methods for Partial Differential Equations*. PhD thesis, Karlsruhe Institute of Technology (KIT), 2023. doi: 10.5445/IR/1000161726.

[63] S. Simonis and M. J. Krause. Forschungsnahe Lehre unter Pandemiebedingungen. *Mitteilungen der Deutschen Mathematiker-Vereinigung*, 30(1):43–45, 2022. doi: 10.1515/dmvm-2022-0015.

[64] S. Simonis and M. J. Krause. Limit consistency of lattice Boltzmann equations. *ESAIM: Mathematical Modelling and Numerical Analysis*, 59(3):1271–1299, 2025. doi: 10.1051/m2an/2025026.

[65] S. Simonis and S. Mishra. Computing statistical Navier–Stokes solutions. In R. Abgrall, M. Garavello, M. Lukáčová-Medvid'ová, and K. Trivisa, editors, *Hyperbolic Balance Laws: Interplay between Scales and Randomness*, volume 21, number 1 of *Oberwolfach Reports*, pages 634–637. EMS Press, 2024. doi: 10.4171/OWR/2024/10.

[66] S. Simonis, M. Frank, and M. J. Krause. On relaxation systems and their relation to discrete velocity Boltzmann models for scalar advection–diffusion equations. *Philosophical Transactions of the Royal Society A*, 378(2175):20190400, 2020. doi: 10.1098/rsta.2019.0400.

[67] S. Simonis, M. Haussmann, L. Kronberg, W. Dörfler, and M. J. Krause. Linear and brute force stability of orthogonal moment multiple-relaxation-time lattice Boltzmann methods applied to homogeneous isotropic turbulence. *Philosophical Transactions of the Royal Society A*, 379: 20200405, 2021. doi: 10.1098/rsta.2020.0405.

[68] S. Simonis, D. Oberle, M. Gaedtke, P. Jenny, and M. J. Krause. Temporal large eddy simulation with lattice Boltzmann methods. *Journal of Computational Physics*, 454:110991, 2022. doi: 10.1016/j.jcp.2022.110991.

[69] S. Simonis, M. Frank, and M. J. Krause. Constructing relaxation systems for lattice Boltzmann methods. *Applied Mathematics Letters*, 137:108484, 2023. doi: 10.1016/j.aml.2022.108484.

[70] S. Simonis, J. Nguyen, S. J. Avis, W. Dörfler, and M. J. Krause. Binary fluid flow simulations with free energy lattice Boltzmann methods. *Discrete and Continuous Dynamical Systems - Series S*, 2023. doi: 10.3934/dcdss.2023069.

[71] S. Simonis, N. Hafen, J. Jeßberger, D. Dapelo, G. Thäter, and M. J. Krause. Homogenized lattice Boltzmann methods for fluid flow through porous media – part I: kinetic model derivation. *ESAIM: M2AN*, 59(2):789–813, 2025. doi: 10.1051/m2an/2025005.

[72] M. Siodlaczek, M. Gaedtke, S. Simonis, M. Schweiker, N. Homma, and M. J. Krause. Numerical evaluation of thermal comfort using a large eddy lattice Boltzmann method. *Building and Environment*, 192:107618, 2021. doi: 10.1016/j.buildenv.2021.107618.

[73] R. C. Smith. *Uncertainty quantification: Theory, Implementation, and Applications*, volume 12. SIAM, 2013. doi: 10.5555/2568154.

[74] S. Smolyak. Quadrature and interpolation formulas for tensor products of certain classes of functions. In *Soviet Math. Dokl.*, volume 4, pages 240–243, 1963.

[75] S. Succi. *The lattice Boltzmann equation: for fluid dynamics and beyond*. Oxford University Press, 2001. doi: 10.1063/1.1537916.

[76] T. J. Sullivan. *Introduction to uncertainty quantification*, volume 63. Springer, 2015. doi: 10.1007/978-3-319-23395-6.

[77] D. Teutscher, F. Bukreev, A. Kummerländer, S. Simonis, P. Bächler, A. Rezaee, M. Hermansdorfer, and M. J. Krause. A digital urban twin enabling interactive pollution predictions and enhanced planning. *Building and Environment*, page 113093, 2025. doi: 10.1016/j.buildenv.2025.113093.

[78] K. Timm, H. Kusumaatmaja, A. Kuzmin, O. Shardt, G. Silva, and E. Viggen. *The lattice Boltzmann method: principles and practice*. Cham, Switzerland: Springer International Publishing AG, 2016. doi: 10.1007/978-3-319-44649-3.

[79] L. N. Trefethen. Is Gauss quadrature better than Clenshaw–Curtis? *SIAM review*, 50(1):67–87, 2008. doi: 10.1137/060659831.

[80] R. Trunk, T. Weckerle, N. Hafen, G. Thäter, H. Nirschl, and M. J. Krause. Revisiting the Homogenized Lattice Boltzmann Method with Applications on Particulate Flows. *Computation*, 9(2), 2021. doi: 10.3390/computation9020011.

[81] L. M. van den Bos, B. Koren, and R. P. Dwight. Non-intrusive uncertainty quantification using reduced cubature rules. *Journal of Computational Physics*, 332:418–445, 2017. doi: 10.1016/j.jcp.2016.12.011.

[82] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17: 261–272, 2020. doi: 10.1038/s41592-019-0686-2.

[83] H. Wang and D. A. Sheen. Combustion kinetic model uncertainty quantification, propagation and minimization. *Progress in Energy and Combustion Science*, 47:1–31, 2015. doi: 10.1016/j.pecs.2014.10.002.

[84] P. Wang, H. Chen, X. Meng, X. Jiang, D. Xiu, and X. Yang. Uncertainty quantification on the macroscopic properties of heterogeneous porous media. *Physical Review E*, 98(3):033306, 2018. doi: 10.1103/PhysRevE.98.033306.

[85] N. Wiener. The homogeneous chaos. *American Journal of Mathematics*, 60(4):897–936, 1938. doi: 10.2307/2371268.

[86] T. Xiao. An investigation of uncertainty propagation in non-equilibrium flows. *International Journal of Computational Fluid Dynamics*, 36(4):294–318, 2022. doi: 10.1080/10618562.2022.2104262.

[87] T. Xiao and M. Frank. A stochastic kinetic scheme for multi-scale flow transport with uncertainty quantification. *Journal of Computational Physics*, 437:110337, 2021. doi: 10.1016/j.jcp.2021.110337.

[88] T. Xiao and M. Frank. A stochastic kinetic scheme for multi-scale plasma transport with uncertainty quantification. *Journal of Computational Physics*, 432:110139, 2021. doi: 10.1016/j.jcp.2021.110139.

[89] D. Xiu. Fast numerical methods for stochastic computations: a review. *Communications in Computational Physics*, 5(2-4):242–272, 2009. doi: 2009-CiCP-7732.

[90] D. Xiu. *Numerical methods for stochastic computations: A spectral method approach*. Princeton University Press, 2010. doi: 10.2307/j.ctv7h0skv.

[91] D. Xiu and J. S. Hesthaven. High-order collocation methods for differential equations with random inputs. *SIAM Journal on Scientific Computing*, 27(3):1118–1139, 2005. doi: 10.1137/040615201.

[92] D. Xiu and G. E. Karniadakis. Modeling uncertainty in steady state diffusion problems via generalized polynomial chaos. *Computer Methods in Applied Mechanics and Engineering*, 191 (43):4927–4948, 2002. doi: 10.1016/S0045-7825(02)00421-8.

[93] D. Xiu and G. E. Karniadakis. Modeling uncertainty in flow simulations via generalized polynomial chaos. *Journal of Computational Physics*, 187(1):137–167, 2003. doi: 10.1016/S0021-9991(03)00092-5.

[94] W. Zhao, J. Huang, and W.-A. Yong. Lattice Boltzmann method for stochastic convection-diffusion equations. *SIAM/ASA Journal on Uncertainty Quantification*, 9(2):536–563, 2021. doi: 10.1137/19M1270665.

[95] G. Zhao-Li, Z. Chu-Guang, and S. Bao-Chang. Non-equilibrium extrapolation method for velocity and pressure boundary conditions in the lattice boltzmann method. *Chinese Physics*, 11 (4):366, 2002. doi: 10.1088/1009-1963/11/4/310.

[96] M. Zhong, S. Zou, D. Pan, C. Zhuo, and C. Zhong. A simplified discrete unified gas kinetic scheme for incompressible flow. *Physics of Fluids*, 32(9):093601, 2020. doi: 10.1063/5.0021332.

[97] M. Zhong, S. Zou, D. Pan, C. Zhuo, and C. Zhong. A simplified discrete unified gas–kinetic scheme for compressible flow. *Physics of Fluids*, 33(3):036103, 2021. doi: 10.1063/5.0033911.

[98] M. Zhong, T. Xiao, M. J. Krause, M. Frank, and S. Simonis. A stochastic Galerkin lattice Boltzmann method for incompressible fluid flows with uncertainties. *Journal of Computational Physics*, 517:113344, 2024. doi: 10.1016/j.jcp.2024.113344.

[99] M. Zhong, A. Kummerländer, S. Ito, M. J. Krause, M. Frank, and S. Simonis. OpenLB-UQ: An Uncertainty Quantification Framework for Incompressible Fluid Flow Simulations. *arXiv preprint*, 2025. doi: 10.48550/arXiv.2508.13867.

[100] M. Zhong, D. Teutscher, A. Kummerländer, M. J. Krause, M. Frank, and S. Simonis. Uncertain data assimilation for urban wind flow simulations with OpenLB-UQ. *arXiv preprint*, 2025. doi: 10.48550/arXiv.2508.18202.