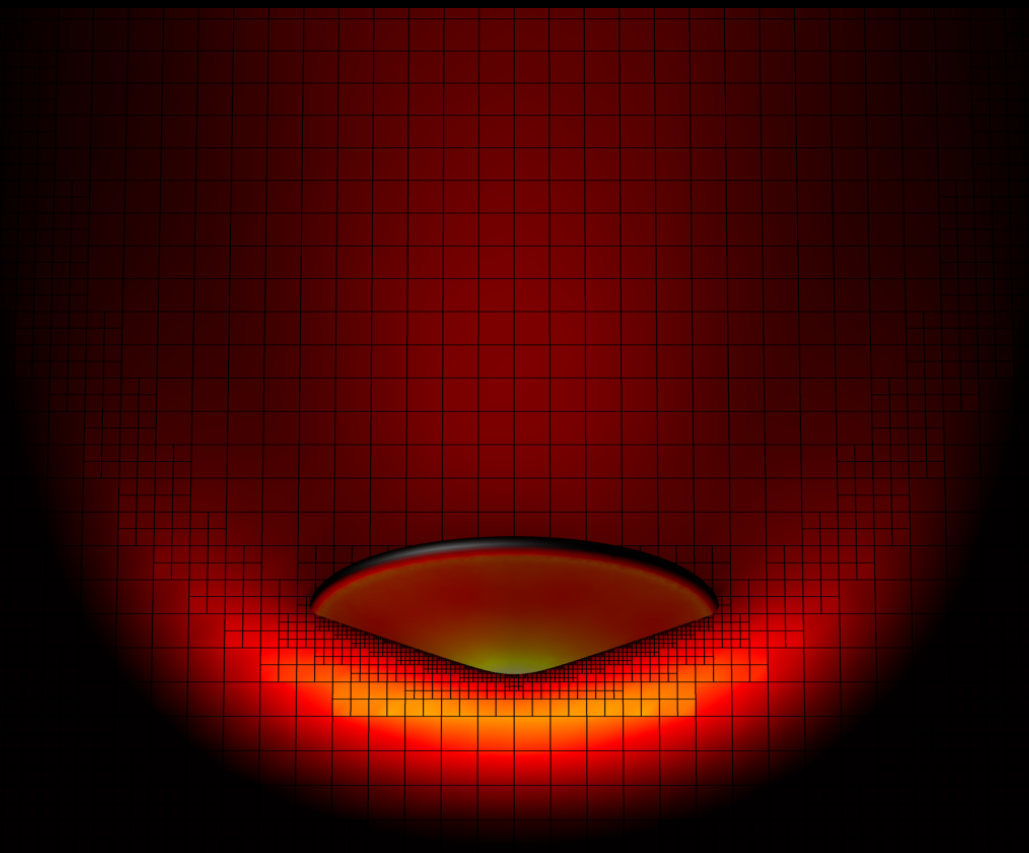
Stochastic Computation of Rarefied Gas Flows Using the Fokker-Planck-DSMC Method: Theory, Algorithms, and Parallel Implementation

Stephan Küchlin



Dissertation ETH No. 25444

DISS. ETH NO. 25444

STOCHASTIC COMPUTATION OF RAREFIED GAS FLOWS USING THE FOKKER-PLANCK-DSMC METHOD: THEORY, ALGORITHMS, AND PARALLEL IMPLEMENTATION

A thesis submitted to attain the degree of

DOCTOR OF SCIENCES of ETH ZURICH
(Dr. sc. ETH Zurich)

presented by

STEPHAN KURT KÜCHLIN
MSc. ETH Zurich

born on October 5th, 1987

citizen of
the Federal Republic of Germany
and the United States of America

accepted on the recommendation of

Prof. Dr. Patrick Jenny, examiner
Prof. Dr. Alejandro L. Garcia, co-examiner
Dr. M. Hossein Gorji, co-examiner

2018

To Eva.

Abstract

The topic of this thesis is the analysis and parallel implementation of the Fokker-Planck-DSMC algorithm for the numerical simulation of rarefied gas flows.

The most established method for this task is the Direct Simulation Monte Carlo (DSMC) technique. For gas flows in the near-continuum regime, however, its computational cost becomes intractable due to the high number of collisions of (computational) particles that need to be computed. The Fokker-Planck (FP) algorithm, on the other hand, provides accurate numerical predictions for near-continuum gas flows at computational cost independent of the number of collisions. In this method, the trajectories of the computational particles evolve independently along continuous stochastic paths. Since both DSMC and the FP algorithm are stochastic particle methods sharing the same underlying structure, they may be coupled seamlessly: the resulting FP-DSMC algorithm is capable of simulating rarefied gas flows from the near-continuum to the fully rarefied regime.

One result of this thesis is a flexible, yet computationally efficient simulation software, which uses both distributed- and shared-memory parallelization to exploit state-of-the-art high-performance computer cluster technologies. It provides the means to conduct computer simulations of flows of diatomic, rarefied gases in complex domains, using many computational particles. The new implementation is used to analyze the accuracy and performance of the FP-DSMC algorithm by means of a variety of simulations. It is shown that given a limited amount of computational resources, using FP-DSMC can provide more accurate results at lower computational cost compared to pure DSMC.

Further, the implementation is capable of performing automatic local mesh refinement, as well as parallel load balancing. This is achieved by choosing space-filling curves (SFCs) as a fundamental concept for the ordering of the computational mesh and particle data. SFCs not only allow for an elegant implementation of these features, but have the additional benefit of ensuring cache-friendly computations. The impact on computational performance of using different space-filling curves is analyzed numerically, and the implementation is demonstrated to deliver accurate simulation results for a relevant test case.

In order to maximize the efficiency gains due to the FP-DSMC algorithm, the computational mesh should be locally adapted to the flow gradients. With this goal in mind, a general theoretical framework for the estimation of mixed partial derivatives of statistics of scattered data is developed based on the concept of kernel density estimation. The new approach allows for the computation of flow gradients locally in each cell of the mesh as a simple weighted sum of the particle states, and may prove useful beyond the scope of rarefied gas flow simulations.

Zusammenfassung

Das Thema dieser Arbeit ist die Analyse und parallele Implementierung des Fokker-Planck-DSMC Algorithmus zur numerischen Simulation von Gasströmungen mit signifikanten mittleren freien Weglängen.

Die etablierteste Methode hierfür ist der *Direct Simulation Monte Carlo* (DSMC) Algorithmus. Für Gasströmungen nahe dem Kontinuumbereich erfordert DSMC jedoch die Berechnung grosser Anzahlen von Kollisionen unter den repräsentativen Partikeln, bis hin zu prohibitiv hohem Rechenaufwand. Der Fokker-Planck (FP) Algorithmus hingegen erlaubt genaue numerische Vorhersagen nahe dem Kontinuumbereich, wobei der numerische Aufwand unabhängig von der Anzahl der Kollisionen ist. Bei dieser Methode bewegen sich die Partikel entlang unabhängiger, kontinuierlicher stochastischer Trajektorien. Da sowohl DSMC als auch FP stochastische Partikelmethoden sind und dieselbe Grundstruktur aufweisen, lassen sich beide Methoden nahtlos koppeln: Es resultiert der FP-DSMC Algorithmus, welcher in der Lage ist, die Strömung von Gasen vom Kontinuum bis hin zum kollisionslosen Grenzfall zu simulieren.

Ein Ergebnis dieser Arbeit ist eine flexible, aber dennoch recheneffiziente parallele Simulationsanwendung, die sowohl verteilten als auch gemeinsam genutzten Hauptspeicher verwenden kann, was die Ausnutzung aktueller Hochleistungsgrossrechner erlaubt. Die Anwendung ermöglicht die numerische Simulation der Strömung von zweiatomigen Gasen unter geometrisch komplexen Randbedingungen, wobei eine grosse Anzahl repräsentativer Partikel verwendet werden kann. Mittels der neuen Implementierung wird die Genauigkeit und das Leistungsvermögen des FP-DSMC Algorithmus anhand mehrerer Simulationen analysiert. Für den Fall begrenzter Rechenressourcen wird gezeigt, dass FP-DSMC, bei geringerem Rechenaufwand, genauere Resultate liefert als DSMC.

Weiterhin ist die Anwendung in der Lage, das numerische Gitter automatisch lokal zu verfeinern, sowie die parallele Rechenlastverteilung dynamisch zu optimieren. Dies wird durch die Verwendung raumfüllender Kurven zur Anordnung der Daten des numerischen Gitters und der repräsentativen Partikel ermöglicht. Diese Anordnung der Datenstrukturen erlaubt nicht nur eine elegante Implementierung der oben genannten Merkmale, sondern gewährleistet darüber hinaus die effiziente Nutzung des Zugriffsspeichers. Die Auswirkung der Wahl verschiedener raumfüllender Kurven auf die Rechenzeit wird analysiert und es wird für einen relevanten Testfall gezeigt, dass die Implementierung genaue Ergebnisse liefert.

Um die Effizienzsteigerung durch die Nutzung des FP-DSMC Algorithmus zu maximieren ist es erforderlich, dass das numerische Gitter anhand der Strömungsgradienten verfeinert wird. Mit dieser Zielsetzung wird die theoretische Grundlage für die Schätzung gemischter partieller Ableitungen von Statistiken verteilter Daten auf Basis der Kerndichteschätzung entwickelt. Der neue Ansatz erlaubt die Berechnung der Strömungsgradienten lokal in jeder Zelle des numerischen Gitters als einfache gewichtete Summe der Partikeldaten und könnte sich auch über die numerische Strömungssimulation hinaus als nützlich erweisen.

Contents

List of Figures

List of Tables

1 Preface

When imagining the physical nature of gas flows, the kinetic picture might not be the first that comes to mind. The mean distance traveled by a molecule in a gas between successive collisions may seem so insignificantly small that even a rather precise mathematical description of gas flow phenomena should be able to safely neglect this range of scales. In many situations, this continuum picture is adequate. However, once the molecular mean free path becomes significant compared to the relevant flow length-scales, adequate numerical predictions of the flow field need to take into account the molecular nature of gas. Many real-world flows exhibit large variations of the molecular mean-free path length. For example, as gas escapes through a nozzle into vacuum, it becomes increasingly rarefied, and will quickly exit the continuum regime. This situation occurs naturally in space vehicle thrusters or physics experiments involving, e.g., mass spectrometry. In hypersonic flows encountered during atmospheric reentry of spacecraft, the generated shock waves have a thickness that may become small compared to the mean free path. For these phenomena, the conventional Navier-Stokes-Fourier (NSF) description, on which continuum flow solvers are based, is no longer valid. The relevant non-dimensional parameter to characterize the degree of rarefaction is the Knudsen number (Kn). Where Kn is large, the flow description needs to be directly based on the Boltzmann equation, which describes the evolution of the fluid density in a high-dimensional phase space under the influence of binary molecular collisions.

The most established numerical technique for solving the Boltzmann equation is the Direct Simulation Monte Carlo (DSMC) method. Ever since its invention in the early 1960s, DSMC has been used to make invaluable contributions to the field of rarefied gas dynamics, from the investigation of fundamental physical phenomena on the microscopic scale, to three-dimensional simulations of aerodynamic forces on the entire international space station. However, the computational cost of DSMC becomes prohibitive in the near-continuum regime. This creates a gap in the range of Knudsen numbers for which accurate numerical predictions of flows can be obtained with manageable computational effort. In other words, the previously mentioned flow situations, which are characterized by a large Kn range, are notoriously difficult to simulate numerically.

The Fokker-Planck (FP) method introduced by Jenny et al., and further developed at ETH's Institute of Fluid Dynamics by Gorji et al., is a particle Monte Carlo scheme that allows for computationally efficient simulations of rarefied gas flows in the low to moderate Kn regime. Here, continuous stochastic processes for the evolution of the gas state are integrated in time, which, for low Kn, may be orders of magnitude faster than the explicit computation of binary collisions necessary in DSMC. Moreover, it is easy to seamlessly couple DSMC and FP, since both algorithms are stochastic particle methods and share the same fundamental structure. This coupling is more consistent than the pairings in previous "hybrid" algorithms that combine partial differential equation solvers and particle methods. The combined Fokker-Planck-DSMC (FP-DSMC) algorithm has already been suc-

cessfully applied to flows covering the whole Kn range, and combines the efficiency of the FP method for small to moderate Kn with the accuracy of DSMC for large Kn.

The goal of this thesis is the development of a parallel implementation of the Fokker-Planck-DSMC algorithm that may be used to further analyze performance and accuracy of FP-DSMC for relevant test cases, ranging beyond the purely academic. Several challenges are addressed: The developed simulation software should be executable in parallel and perform well on state-of-the-art high-performance computing clusters. Without changing the code, one would like to be able to simulate many different flow scenarios, from internal flows in micro-electro-mechanical (MEMS) devices to external flows over space vehicles, not limited by the complexity of the geometry. The implementation should be specifically suited for the simulation of flows covering the whole Kn range, the staple of the FP-DSMC algorithm. Last but not least, the software should be designed from the ground up with future extensions in mind: adding a richer physical description and more numerical capabilities should necessitate as few changes as possible to existing code.

1.1 Outline

With the above as a goal in mind, the remainder of this thesis is structured as follows: Part I serves as a brief introduction to the field of rarefied gas dynamics. The key aspects of the underlying statistical modeling of rarefied gases are presented, along with a short description of the relevant algorithms for their numerical simulation. A larger section is devoted to the review of the Fokker-Planck algorithm and its coupling to DSMC. The aim of this section is to provide a unified overview of the underlying assumptions and derivations of the FP method for diatomic gases.

The original research contributions of this Ph.D. project are presented in parts II, III, and IV, which are largely identical to the corresponding articles published in the Journal of Computational Physics [KJ17; KJ18a], and the recently submitted manuscript [KJ18b], respectively.

Part II concerns the overall structure of the implementation, and the presented test cases confirm the accuracy and performance of FP-DSMC in general and of the developed code in particular.

In Part III, the extension of the code with capabilities for automatic mesh refinement and parallel load balancing are discussed. Specifically, it is demonstrated that using particle and mesh data structures ordered by space-filling curves (SFCs) allows for a (relatively) easy yet efficient implementation of these features. The data structures and algorithms presented are applicable to any particle-based simulation method, beyond the scope of FP-DSMC. The impact of using different SFCs on computational performance is studied, and again, relevant simulation test cases demonstrate the accuracy and efficiency of the implementation.

Part IV is more theoretical in nature: motivated by the need for estimates of gradients of flow variables to inform the automatic mesh refinement procedure, a

general framework for the kernel-based estimation of arbitrary-order mixed derivatives of statistics of scattered data is developed. The technique may prove useful for other particle methods, from molecular dynamics to astrophysics simulations, but also to other fields such as machine intelligence and data analysis tasks.

Based on the above, Part V describes current development steps, i.e., the application of the gradient estimation procedure to local mesh refinement, as well as the necessary algorithmic adjustments to use local variable time steps. Additionally, the advanced Fokker-Planck position integration scheme presented by Jenny et al. is reviewed.

Part VI completes the thesis with a short summary and conclusions, as well as an outlook on future algorithmic, modeling, and simulation tasks.

1.2 Acknowledgments

Several people have contributed, directly or indirectly, to this thesis:

I would like to thank my supervisor Professor Patrick Jenny for giving me the opportunity to conduct this project. I am grateful for his scientific supervision and guidance, as well as for the freedom he gave me to pursue my research interests. I am also indebted to my co-supervisor Dr. Hossein Gorji for his advice, patient answers to my questions, and for sharing his latest ideas and inspirations. I would like to thank Professor Alejandro Garcia for kindly agreeing to co-referee this thesis.

The staff of the Institute of Fluid Dynamics has been most helpful, and I would like to thank Ms. Bianca Maspero and Mr. Hans Peter Caprez. Several students conducted their projects under my supervision and their feedback helped improving the developed code. Many thanks go to my present and former colleagues at the institute, with whom it was a pleasure to work, discuss, learn, go to conferences, laugh, do sports, survive the group adventures, and spend time with.

Finishing this project would not have been possible without the love and support of my wife Eva: thank you for everything. I am grateful to my parents for their support throughout the years. Special thanks go to my father for proofreading this thesis.

Part I
Introduction

In the following, the necessary theoretical background for the subsequent chapters is provided. The section follows standard textbooks in the field, in particular [Bir94; Cer00; Str05], and the interested reader is referred to these works for more in-depth explanations and derivations.

2 Statistical Modeling of the Dynamics of Rarefied Gases

At atmospheric conditions ($p \approx 101\,\mathrm{kPa}$, $T \approx 293\,\mathrm{K}$), one cubic meter of gas contains about 2.5×10^{25} molecules. Clearly, making numerical predictions about the behavior of each of such a large number of molecules is not possible on current or any envisioned computational platforms in the foreseeable future. Using 64-bit numbers (commonly used to address computer memory), it is not even possible to assign a unique integer index to each of the 2.5×10^{25} molecules. Consequently, the microscopic description of individual gas molecules and their interactions, and the corresponding numerical methods (Molecular Dynamics), are limited to the study of very small systems.

The other end of a scale measuring different modeling approaches by the amount of detail provided, inversely related to their computational complexity (see Figure 1), is represented by continuum methods: For many flows, it is completely sufficient to treat the gas in the so-called hydrodynamic limit, and to consider partial differential equations derived from the laws of conservation of mass, momentum and energy. The modeled quantities are the macroscopic fields, such as the bulk velocity of the fluid, its pressure, density, etc. Together with the assumptions (constitutive relations) that stresses are a linear function of the gradient of velocity (law of Navier-Stokes), and that molecular heat flux is a linear function of the gradient of temperature (Fourier's law), one obtains the celebrated Navier-Stokes-Fourier (NSF) equations. Intuitively, one may say that the laws of Navier-Stokes and Fourier are justified whenever enough collisions between the particles occur on the considered time and length scale to effectively propagate information about the gradients in the flow. The average distance traveled by a molecule between successive collisions is the mean free path λ, and as long as the ratio of the mean free path to the characteristic flow length scale L is small, the NSF equations are valid. The dimensionless number given by this ratio is the Knudsen number $Kn = \lambda/L$, and the NSF equations require $Kn \ll 1$ to hold.

However, important "real-world" applications exist where the Knudsen number is not small and thus the NSF equations are no longer valid: for example, both flows in micro electro-mechanical systems (MEMS), as well as hypersonic flows

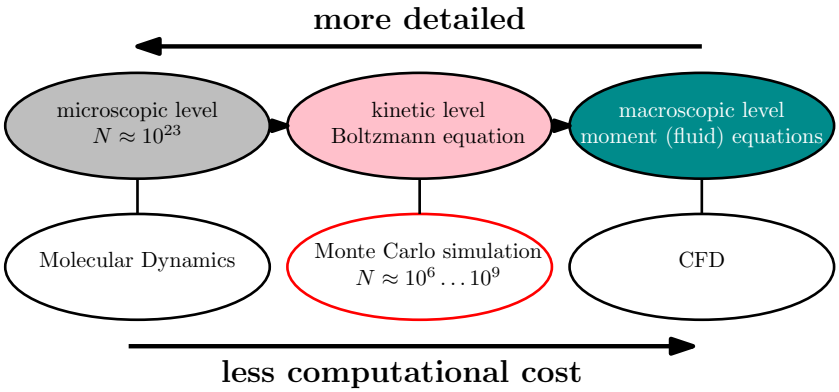


Figure 1: Three conceptual approaches to modeling gas flows, with corresponding numerical methods, arranged by the level of detail provided, inversely related to computational complexity. N is the number of particles.

encountered by space vehicles during atmospheric reentry, are characterized by small values of the characteristic length L and thus moderate to large values of Kn. The first due to the small dimensions of the device, the second due to the steep gradients of the flow properties caused by strong shock waves. In flows with low density, such as those encountered by space vehicles in orbit, or—on the laboratory scale—in physics experiments or industrial processes carried out in near vacuum, λ becomes large, and hence also Kn. Gas flows in which $Kn \gtrapprox 0.01$ are said to be in the rarefied regime. In these situations, the mathematical modeling, and consequently the numerical simulation of gas flows, must take the molecular nature of the fluid into account.

The "middle ground" between the microscopic description of each individual molecule on the one hand, and the purely macroscopic picture provided by the NSF equations on the other, is the kinetic theory of gases. Here, the "solution" quantities are again the macroscopic fields. However, these are derived from the single particle phase density, which provides information about the probability of encountering a molecule with given state at a certain point in time and space. The molecular ensemble is described on a statistical level. It turns out that modeling, and in turn simulating, the evolution in space and time of the phase density is sufficient to capture the effects attributable to the molecular nature of a gas, while still being abstract enough to provide information about moderate to large systems at manageable computational cost. The Fokker-Planck-DSMC algorithm, the topic of this thesis, is a numerical tool developed for this purpose, with the aim of expanding the boundaries of the set of problems tractable by computer simulation.

Some background on kinetic modeling, the evolution equation of the phase density, and—most importantly here—the Fokker-Planck-DSMC algorithm, is given in the following.

2.1 Phase Density and Link to Macroscopic Quantities

Let $\boldsymbol{\xi}\colon \Omega \to \mathbb{R}^{d_x}$ and $\boldsymbol{\zeta}\colon \Omega \to \mathbb{R}^{d_c}$ be absolutely continuous random vectors, where Ω denotes the sample space. $\boldsymbol{\xi}$ and $\boldsymbol{\zeta}$ describe the position of a given molecule in d_x-dimensional physical space with coordinate vector $\boldsymbol{x} \in \mathbb{R}^{d_x}$, and d_c-dimensional state space with coordinate vector $\boldsymbol{c} \in \mathbb{R}^{d_c}$, respectively. The state space coordinate vector $\boldsymbol{c}$ has components $\boldsymbol{c} = (\boldsymbol{v}, \boldsymbol{c}_{\text{int}})$, where the vector $\boldsymbol{v} \in \mathbb{R}^3$ corresponds to velocity, and for complex molecules with $d_{c,\text{int}}$ internal degrees of freedom, the vector $\boldsymbol{c}_{\text{int}} \in \mathbb{R}^{d_{c,\text{int}}}$ collects the corresponding internal states. In this thesis, only continuously varying (i.e., non-quantized) internal states are considered.

Define the (non-normalized) phase density $f\colon \mathbb{R}^{d_x} \times \mathbb{R}^{d_c} \to \mathbb{R}$, such that, at time t, the number of molecules in the infinitesimal volume-element of $d_x + d_c$-dimensional phase space, located at point $(\boldsymbol{x}, \boldsymbol{c})$, is given by [Str05, Equation (2.1)]

$$\mathrm{d}N = \frac{1}{m} f(\boldsymbol{x}, \boldsymbol{c}; t) \mathrm{d}\boldsymbol{x} \mathrm{d}\boldsymbol{c}, \tag{1}$$

where m is the molecular mass. Note that here and in the following, the abbreviated notation $\mathrm{d}\boldsymbol{x}_1 \cdots \mathrm{d}\boldsymbol{x}_n = \mathrm{d}x_{11}\mathrm{d}x_{12}\cdots\mathrm{d}x_{1d_1} \times \cdots \times \mathrm{d}x_{n1}\mathrm{d}x_{n2}\cdots\mathrm{d}x_{nd_n}$ is used to denote an infinitesimal volume-element of a $d_1 + \cdots + d_n$-dimensional space, located at point $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$.

The phase density may be factored into the gas density $\rho(\boldsymbol{x})$ and the normalized conditional state probability density function (PDF) $f_{\boldsymbol{\zeta}}(\boldsymbol{c}, \boldsymbol{x}; t)$ satisfying $\int_{\mathbb{R}^{d_c}} f_{\boldsymbol{\zeta}}(\boldsymbol{c}, \boldsymbol{x}; t)\mathrm{d}\boldsymbol{c} \equiv 1$, i.e.,

$$f(\boldsymbol{x}, \boldsymbol{c}; t) = \rho(\boldsymbol{x}; t) f_{\boldsymbol{\zeta}}(\boldsymbol{c}, \boldsymbol{x}; t). \tag{2}$$

2.1.1 Moments of the Phase Density

The phase density may be linked to the familiar variables of continuum theory via its moments, that is, weighted integrals of f over state space. Knowledge of f, together with the ideal gas law, thus determines the field quantities: density ρ, mean velocity $\boldsymbol{v}$, pressure tensor p_{ij}, heat flux vector $\boldsymbol{q}$, etc. The most frequently used relations are stated below. In the following, the explicit time dependence is dropped for brevity of notation.

The gas density ρ may be written as

$$\rho(\boldsymbol{x}) = \int_{\mathbb{R}^{d_c}} f(\boldsymbol{x}, \boldsymbol{c}')\mathrm{d}\boldsymbol{c}'. \tag{3}$$

The ith component of the mean, or bulk, velocity vector of the gas flow is given by

$$U_i(\boldsymbol{x}) = \frac{1}{\rho(\boldsymbol{x})} \int_{\mathbb{R}^{d_c}} c_i f(\boldsymbol{x}, \boldsymbol{c}')\mathrm{d}\boldsymbol{c}', \quad i = 1, 2, 3, \tag{4}$$

which in turn defines the thermal, or peculiar, velocity vector

$$\boldsymbol{V} = \boldsymbol{v} - \boldsymbol{U}. \tag{5}$$

One obtains the pressure tensor

$$p_{ij}(\boldsymbol{x}) = \int_{\mathbb{R}^{d_c}} V_i'(\boldsymbol{x}) V_j'(\boldsymbol{x}) f(\boldsymbol{x}, \boldsymbol{c}') \mathrm{d}\boldsymbol{c}', \quad i, j = 1, 2, 3, \tag{6}$$

from which the scalar pressure field $p(\boldsymbol{x})$ follows as $p = \frac{1}{3} p_{kk}$. Note that here and throughout, Einstein's summation convention is implied, that is, quantities with repeated indices are understood to be summed over the index range, unless stated otherwise. Consequently, $p = \frac{1}{3} p_{kk} \equiv \frac{1}{3} \sum_{k=1}^{3} p_{kk}$.

The trace-free part of the pressure tensor is the stress tensor

$$\sigma_{ij}(\boldsymbol{x}) = p_{\langle ij \rangle} = p_{ij} - p\delta_{ij}, \quad i, j = 1, 2, 3. \tag{7}$$

Note that here and in the following, angular brackets around tensor indices indicate the trace-free part of the respective tensor.

From the ideal gas law $p = \rho R T$, where the (specific) gas constant $R = \frac{k_B}{m}$ is defined in terms of the Boltzmann constant $k_B \approx 1.380\,648\,52 \times 10^{-23}\,\mathrm{J\,K^{-1}}$ and the molecular mass m, one may define the temperature T as

$$T(\boldsymbol{x}) = \frac{p(\boldsymbol{x})}{\rho(\boldsymbol{x}) R}, \tag{8}$$

and the temperature in "energy units" as $\theta = RT$.

Total energy density and average specific thermal energy per molecule are calculated as

$$w(\boldsymbol{x}) = w_{\mathrm{tr}}(\boldsymbol{x}) + w_{\mathrm{int}}(\boldsymbol{x}) = \frac{1}{2} \int_{\mathbb{R}^{d_c}} \|\boldsymbol{c}'\|^2 f(\boldsymbol{x}, \boldsymbol{c}') \mathrm{d}\boldsymbol{c}' \tag{9}$$

and

$$\varepsilon(\boldsymbol{x}) = \varepsilon_{\mathrm{tr}}(\boldsymbol{x}) + \varepsilon_{\mathrm{int}}(\boldsymbol{x}) = \frac{1}{\rho(\boldsymbol{x})} \int_{\mathbb{R}^{d_c}} e(\boldsymbol{x}, \boldsymbol{c}') f(\boldsymbol{x}, \boldsymbol{c}') \mathrm{d}\boldsymbol{c}', \tag{10}$$

respectively, where

$$e(\boldsymbol{x}, \boldsymbol{c}) = \frac{1}{2} \|\boldsymbol{V}(\boldsymbol{x}, \boldsymbol{v})\|^2 + e_{\mathrm{int}}(\boldsymbol{c}_{\mathrm{int}}) \tag{11}$$

is the specific thermal energy of a molecule at position $\boldsymbol{x}$ in state $\boldsymbol{c}$, comprised of the thermal kinetic energy $e_{\mathrm{tr}} = \frac{1}{2} \|\boldsymbol{V}\|^2$ and the specific energy e_{int} associated with the internal state $\boldsymbol{c}_{\mathrm{int}}$. Total energy density and average specific thermal energy per molecule are related via $w = \rho\varepsilon + \frac{1}{2}\rho \|\boldsymbol{U}\|^2$. The average specific thermal energy ε is frequently referred to as "internal" energy; here, the term "thermal" is used instead to avoid confusion with the energy associated with the internal degrees of freedom. Further, specific translational thermal energy $\varepsilon_{\mathrm{tr}}$ and the temperature in energy units θ are related via $\theta = \frac{2}{3}\varepsilon_{\mathrm{tr}}$.

The (translational) heat flux vector $\boldsymbol{q}_{\mathrm{tr}}$ is given by

$$q_{\mathrm{tr},i}(\boldsymbol{x}) = \frac{1}{2} \int_{\mathbb{R}^{d_c}} V_i(\boldsymbol{x}) \|\boldsymbol{V}(\boldsymbol{x}, \boldsymbol{c}')\|^2 f(\boldsymbol{x}, \boldsymbol{c}') \mathrm{d}\boldsymbol{c}', \quad i = 1, 2, 3, \tag{12}$$

and the flux vector of total thermal energy is given by

$$q_i(\boldsymbol{x}) = \frac{1}{2} \int_{\mathbb{R}^{d_c}} V_i'(\boldsymbol{x}) e(\boldsymbol{x}, \boldsymbol{c}') f(\boldsymbol{x}, \boldsymbol{c}') \mathrm{d}\boldsymbol{c}', \quad i = 1, 2, 3. \tag{13}$$

2.2 The Boltzmann Equation

In order to make predictions about the behavior of a gas, an equation for the space-time evolution of the phase density f is required. Assuming that the gas under consideration is dilute, i.e., $\lambda \gg d$, where d is the molecular diameter, and considering only binary collisions of molecules with uncorrelated velocities as the source of inter-molecular force transmission, the appropriate description is given by the equation devised by Boltzmann in 1872. Following [Kre10], a sketch of the derivation of the (generalized) Boltzmann equation is provided below: The general equation governing the evolution of the phase density (in the absence of external and long-range forces) may be written as

$$\frac{\partial}{\partial t} f(\boldsymbol{x}, \boldsymbol{c}; t) + v_i \frac{\partial}{\partial x_i} f(\boldsymbol{x}, \boldsymbol{c}; t) = S(f(\boldsymbol{x}, \boldsymbol{c}; t)), \tag{14}$$

where the left hand side describes the change of f due to the transport (convection), and the right hand side—the collision operator $S(f)$—characterizes the influence of inter-molecular forces mediated via collisions.

Any collision operator S should respect the conservation of mass, momentum and energy, since these properties are obviously conserved in individual binary collisions.

The collision operator for the Boltzmann equation ($S_{\text{Boltz.}}$) may be divided into two terms, $S^{+}_{\text{Boltz.}}$ and $S^{-}_{\text{Boltz.}}$, which are proportional to the number of collisions per unit time and volume from which molecules with state $\boldsymbol{c}$ emerge and are consumed, respectively.

In the center-of-mass frame-of-reference, a binary interaction occurs in a two dimensional plane, and may be parameterized by the impact parameter b, $0 \leq b \leq \infty$, the distance of closest approach of the undisturbed trajectories of the interacting particles in the center-of-mass frame-of-reference, and by the azimuthal angle ϵ, $0 \leq \epsilon \leq 2\pi$, the angle of the collision plane w.r.t. a reference plane.

Let two interacting particles have states $\boldsymbol{c}$ and $\boldsymbol{c}'$, respectively, and let the vector $\boldsymbol{g} = \boldsymbol{v} - \boldsymbol{v}'$ denote their relative velocity. Further, let the superscripted quantities $\boldsymbol{c}^{\star}, \boldsymbol{c}'^{\star}, \boldsymbol{g}^{\star}, b^{\star}, \epsilon^{\star}$ denote the same quantities, but for a pair of particles, that, after the collision, will be in a state characterized by the un-superscripted quantities. It is assumed that all particles obey Hamilton's equations of motion, and thus, by Liouville's theorem, volumes of phase space along Hamiltonian trajectories are conserved. In particular, for the differential $2 \times (d_c + d_x)$-dimensional volumes of phase space of the two-particle system before and after an interaction, the following relationship holds [Kre10, Equation 5.95]:

$$g^{\star} b^{\star} \mathrm{d}b^{\star} \mathrm{d}\epsilon^{\star} \mathrm{d}\boldsymbol{c}^{\star} \mathrm{d}\boldsymbol{c}'^{\star} \equiv g b \mathrm{d}b \mathrm{d}\epsilon \mathrm{d}\boldsymbol{c} \mathrm{d}\boldsymbol{c}'. \tag{15}$$

Consider the situation depicted in Figure 2 ([Str05, Figure 3.4]): all particles located in the differential phase space element $\mathrm{d}\boldsymbol{x}\mathrm{d}\boldsymbol{c}'$ will suffer collisions with all particles located in the phase space cylinder of volume $g\mathrm{d}t b\mathrm{d}b\mathrm{d}\epsilon$ during the time

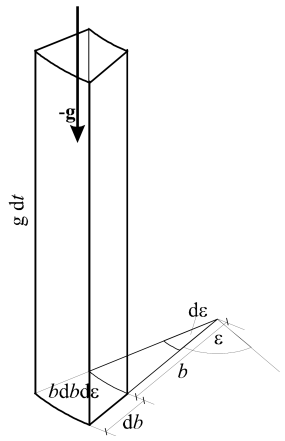


Figure 2: [Str05, Fig. 3.4]. Parametrization of a binary collision.

interval dt. Their density in phase space is $\frac{1}{m}f(\boldsymbol{x},\boldsymbol{c}')$, and the number of particles in the aforementioned cylinder is $\frac{1}{m}f(\boldsymbol{x},\boldsymbol{c})g\mathrm{d}t b\mathrm{d}b\mathrm{d}\epsilon\mathrm{d}\boldsymbol{c}$, giving the total number of binary interactions in the considered differential volume of phase space as

$$\mathrm{d}N^{-} = \frac{1}{m^2}f(\boldsymbol{x},\boldsymbol{c})f(\boldsymbol{x},\boldsymbol{c}')g\mathrm{d}t b\mathrm{d}b\mathrm{d}\epsilon\mathrm{d}\boldsymbol{x}\mathrm{d}\boldsymbol{c}\mathrm{d}\boldsymbol{c}'. \tag{16}$$

From the above, the rate of depletion of the phase density $f(\boldsymbol{x},\boldsymbol{c})$ follows by integrating over all "test"-velocities $\boldsymbol{c}'$ and over all possible values of the collision parameters b and ϵ, viz.

$$S^{-}_{\text{Boltz.}} = \frac{1}{m}\int f(\boldsymbol{x},\boldsymbol{c})f(\boldsymbol{x},\boldsymbol{c}')gb\mathrm{d}b\mathrm{d}\epsilon\mathrm{d}\boldsymbol{c}'. \tag{17}$$

Using a similar reasoning for particles located in the phase space element $\mathrm{d}t\mathrm{d}\boldsymbol{x}\mathrm{d}\boldsymbol{c}^{\star}$ $\times g^{\star}b^{\star}\mathrm{d}b^{\star}\mathrm{d}\epsilon^{\star}\mathrm{d}\boldsymbol{c}'^{\star}$, which interact to increase the phase density $f(\boldsymbol{x},\boldsymbol{c})$, one finds

$$S^{+}_{\text{Boltz.}} = \frac{1}{m}\int f(\boldsymbol{x},\boldsymbol{c}^{\star})f(\boldsymbol{x},\boldsymbol{c}'^{\star})g^{\star}b^{\star}\mathrm{d}b^{\star}\mathrm{d}\epsilon^{\star}\mathrm{d}\boldsymbol{c}'^{\star}. \tag{18}$$

Note that the quantities $\boldsymbol{g}$, $\boldsymbol{g}^{\star}$, $\boldsymbol{c}^{\star}$ and $\boldsymbol{c}'^{\star}$ are all functions of $\boldsymbol{c}$ and $\boldsymbol{c}'$, being linked by the requirement that the considered interaction should have the specified final state. Equation (15) allows $S_{\text{Boltz.}}$ to be written as

$$\begin{aligned} S_{\text{Boltz.}} &= S^{+}_{\text{Boltz.}} + S^{-}_{\text{Boltz.}} \\ &= \frac{1}{m}\int (f(\boldsymbol{x},\boldsymbol{c}^{\star};t)f(\boldsymbol{x},\boldsymbol{c}'^{\star};t) - f(\boldsymbol{x},\boldsymbol{c};t)f(\boldsymbol{x},\boldsymbol{c}';t))gb\mathrm{d}b\mathrm{d}\epsilon\mathrm{d}\boldsymbol{c}', \end{aligned} \tag{19}$$

to yield, as a final result, the generalized Boltzmann equation for a dilute gas, viz.

$$\begin{aligned} &\frac{\partial}{\partial t}f(\boldsymbol{x},\boldsymbol{c};t) + v_i\frac{\partial}{\partial x_i}f(\boldsymbol{x},\boldsymbol{c};t) = \\ &\quad\frac{1}{m}\int (f(\boldsymbol{x},\boldsymbol{c}^{\star};t)f(\boldsymbol{x},\boldsymbol{c}'^{\star};t) - f(\boldsymbol{x},\boldsymbol{c};t)f(\boldsymbol{x},\boldsymbol{c}';t))gb\mathrm{d}b\mathrm{d}\epsilon\mathrm{d}\boldsymbol{c}'. \end{aligned} \tag{20}$$

2.2.1 Equilibrium Phase Density

The equilibrium phase density f_E implied by Equation (20) is the unique function f for which $S_{\text{Boltz.}}(f_E) \equiv 0$. From Equation (20), it follows that $f_E(\boldsymbol{x}, \boldsymbol{c}^\star) f_E(\boldsymbol{x}, \boldsymbol{c}'^\star) \equiv f_E(\boldsymbol{x}, \boldsymbol{c}) f_E(\boldsymbol{x}, \boldsymbol{c}')$, and hence also

$$\ln f_E(\boldsymbol{x}, \boldsymbol{c}^\star) + \ln f_E(\boldsymbol{x}, \boldsymbol{c}'^\star) \equiv \ln f_E(\boldsymbol{x}, \boldsymbol{c}) + \ln f_E(\boldsymbol{x}, \boldsymbol{c}'). \tag{21}$$

One may prove [Kre10, Theorem 1] that any function satisfying the above relation must be a linear combination of the conserved quantities mass, momentum and energy, which implies

$$\ln f_E = A + \boldsymbol{B} \cdot \boldsymbol{v} + C e(\boldsymbol{c}), \tag{22}$$

with constants A, $\boldsymbol{B}$ and C, which may be determined by the macroscopic quantities implied by f_E. The equilibrium phase density follows as [Kre10, Equation (5.38)]

$$f_{|E}(\boldsymbol{x}, \boldsymbol{c}) = \left(\frac{1}{2\pi\theta(\boldsymbol{x})}\right)^{\frac{3}{2}} \frac{\exp(-e_{\text{int}}(\boldsymbol{c}_{\text{int}})/\theta(\boldsymbol{x}))}{\mathcal{Z}(\boldsymbol{x})} \exp\left(-\frac{\|\boldsymbol{V}(\boldsymbol{x}, \boldsymbol{v})\|^2}{2\theta(\boldsymbol{x})}\right), \tag{23}$$

where $\mathcal{Z} = \int \exp(-e_{\text{int}}(\boldsymbol{c}'_{\text{int}})/\theta) \mathrm{d}\boldsymbol{c}'_{\text{int}}$ is the partition function. For classical degrees of freedom for which $e_{\text{int}} \equiv \frac{1}{2}\|\boldsymbol{c}_{\text{int}}\|^2$, one has $\mathcal{Z} = (2\pi\theta)^{\frac{d_{c,\text{int}}}{2}}$.

2.3 Half-Range Phase Density

Especially for the specification of boundary conditions, it is of interest to know the phase density of particles crossing a directed area element $\mathrm{d}\boldsymbol{A} = \boldsymbol{n}\mathrm{d}A$ with unit normal vector $\boldsymbol{n}$. Consider the situation depicted in Figure 3: the mass of particles with states in the interval $(\boldsymbol{c}, \boldsymbol{c} + \mathrm{d}\boldsymbol{c})$ crossing the area element $\mathrm{d}\boldsymbol{A}$ during the time interval $\mathrm{d}t$, assuming that their velocity vectors $\boldsymbol{v}$ satisfy $\boldsymbol{v} \cdot \boldsymbol{n} < 0$, is $\mathrm{d}M^+(\boldsymbol{x}, \boldsymbol{c}) = \mathrm{d}t\mathrm{d}A|\boldsymbol{v} \cdot \boldsymbol{n}| f(\boldsymbol{x}, \boldsymbol{c}) \mathrm{d}\boldsymbol{c}$. The total mass of all particles crossing the directed area element $\mathrm{d}\boldsymbol{A}$ with velocities $\boldsymbol{v}'$ $v'_k n_k < 0$ during the same time interval is $\mathrm{d}M^+(\boldsymbol{x}) = \mathrm{d}t\mathrm{d}A \int |\boldsymbol{v}' \cdot \boldsymbol{n}| \mathcal{H}(-\boldsymbol{v}' \cdot \boldsymbol{n}) f(\boldsymbol{x}, \boldsymbol{c}') \mathrm{d}\boldsymbol{c}'$, where $\mathcal{H}(x)$ is the Heaviside function equal to 1 if its argument x satisfies $x \geq 0$, and 0 else. The phase density of particles crossing the area element $\mathrm{d}\boldsymbol{A}$ with velocities $\boldsymbol{v} \cdot \boldsymbol{n} < 0$ is thus

$$\begin{aligned} f^+(\boldsymbol{x}, \boldsymbol{c}) &= \rho(\boldsymbol{x}) \frac{\mathrm{d}M^+(\boldsymbol{x}, \boldsymbol{c})}{\mathrm{d}M^+(\boldsymbol{x})} \\ &= \rho(\boldsymbol{x}) \mathcal{H}(-\boldsymbol{v} \cdot \boldsymbol{n}) \frac{|\boldsymbol{v} \cdot \boldsymbol{n}| f(\boldsymbol{x}, \boldsymbol{c})}{\int |\boldsymbol{v}' \cdot \boldsymbol{n}| \mathcal{H}(-\boldsymbol{v}' \cdot \boldsymbol{n}) f(\boldsymbol{x}, \boldsymbol{c}') \mathrm{d}\boldsymbol{c}'}. \end{aligned} \tag{24}$$

In equilibrium, i.e., when $f \equiv f_{|E}$, the denominator in Equation (24) reads [Bir94, Equation (4.22)]

$$\begin{aligned} \frac{\mathrm{d}M^+(\boldsymbol{x})}{\mathrm{d}t\mathrm{d}A} &= \int |\boldsymbol{v}' \cdot \boldsymbol{n}| \mathcal{H}(-\boldsymbol{v}' \cdot \boldsymbol{n}) f_E(\boldsymbol{x}, \boldsymbol{c}') \mathrm{d}\boldsymbol{c}' \\ &= \rho(\boldsymbol{x}) \sqrt{\frac{\theta(\boldsymbol{x})}{2\pi}} \Big[\exp\Big(-s(\boldsymbol{x})^2\Big) + \sqrt{\pi} s(\boldsymbol{x}) \Big(1 + \operatorname{erf}(s(\boldsymbol{x}))\Big)\Big], \end{aligned} \tag{25}$$

where $s(\boldsymbol{x}) = \boldsymbol{U}(\boldsymbol{x}) \cdot \boldsymbol{n} \Big/ \sqrt{2\theta(\boldsymbol{x})}$.

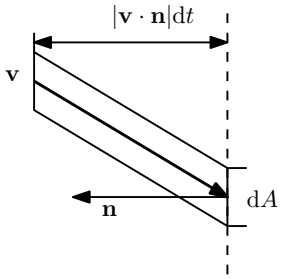


Figure 3: Illustration of phase space flux.

3 Numerical Algorithms

Various algorithms for the numerical simulation of Equation (20) exist: direct simulation, or conventional numerical discretization of the equation, is rendered prohibitively expensive in terms of computational resources for all but the simplest cases by the presence of the $d_c + 2$-dimensional integral in the collision term. The problem is ameliorated by assuming a simplified form of the collision operator, the most well-known being the Bhatnager-Gross-Krook model (BGK) (see, e.g., [Str05, p. 46ff.]).

Equations for the transport of moments may be derived by multiplying Equation (20) by functions of the state variable and analytical integration over the state space. The resulting conservation laws may be integrated in time much like the NSF equations for hydrodynamics, see, e.g., the works of Torhillon, Struchtrup, et al. [Str05; TAS03; TS04]. In order to evaluate integrals of the collision operator, either the collision dynamics must be assumed to be governed by a certain molecular model known as Maxwell molecules, or a certain functional form of f with known moments is assumed. Alternatively, one may model these so-called production terms as functions of known moments.

This thesis is concerned with a third class of methods described in more detail below, namely stochastic particle algorithms.

3.1 Generic Particle Monte Carlo Algorithm

One of the most widely used approaches for the simulation of gas flow based on the Boltzmann equation are particle Monte Carlo algorithms. As in moment methods, the quantities of interest are the macroscopic fields, which, as stated earlier, may be obtained as weighted integrals of the phase density. Particle Monte Carlo methods approximate these integrals by finite sums of random samples of the respecitve integrand, which results in the well-known convergence rate of the mean squared error of these estimates of $1/N$, where N is the number of samples. The samples, in turn, are given by functions of the state-space vectors of an ensemble of computational particles. Conceptually, each of these particles can be thought of as representing a large number of physical molecules. As an ensemble, they constitute a point-mass representation of the phase density. More formally, consider

N computational particles, each with index l, (statistical) weight w^l, random position vector $\boldsymbol{\xi}^l$ and random state vector $\boldsymbol{\zeta}^l$. The ratio of statistical weight w to molecular mass m, commonly referred to as $F_N = w/m$ (read: «eff-num») is an important parameter in particle Monte Carlo algorithms, comparable to a "resolution" of f.

In the limit $N \to \infty$, the ensemble exactly represents the phase density, viz.

$$f(\boldsymbol{x}, \boldsymbol{c}) = \lim_{N\to\infty} \sum_{l=1}^{N} w^l \delta\Big(\boldsymbol{x} - \boldsymbol{\xi}^l\Big)\delta\Big(\boldsymbol{c} - \boldsymbol{\zeta}^l\Big). \tag{26}$$

A generalized moment of f with weight function $g(\boldsymbol{c})$ may be computed in terms of its point-mass representation via

$$\int g(\boldsymbol{c}')f(\boldsymbol{x}, \boldsymbol{c}')\mathrm{d}\boldsymbol{c}' = \lim_{N\to\infty} \sum_{l=1}^{N} w^l \delta\Big(\boldsymbol{x} - \boldsymbol{\xi}^l\Big) g\Big(\boldsymbol{\zeta}^l\Big). \tag{27}$$

In the practical case of finite N, the simulated volume of physical space is disrcetized into a finite number M of non-overlapping volume elements ("grid cells") Ω^j, each with index j and volume $|\Omega^j|$. An approximation to the average value of the moment Equation (27) in cell j is then given by

$$\frac{1}{|\Omega^j|}\int_{\Omega^j}\int g(\boldsymbol{c}')f(\boldsymbol{x}', \boldsymbol{c}')\mathrm{d}\boldsymbol{c}'\mathrm{d}\boldsymbol{x}' \approx \frac{1}{|\Omega^j|}\sum_{l:\,\boldsymbol{\xi}^l\in\Omega^j} g\Big(\boldsymbol{\zeta}^l\Big). \tag{28}$$

A more rigorous treatment of the above is deferred to Part IV of this thesis.

As in many numerical simulation algorithms, time is discretized into finite time steps of length Δt. In order to simulate the Boltzmann equation, the ensemble of (computational) particles needs to be evolved in time such that the evolution of the represented density is consistent with Equation (20), to the desired level of approximation. The most basic requirements for consistent simulations are the conservation of mass, momentum and energy over each time step. In a particle algorithm, conservation of mass is trivial to fulfill by keeping the number and masses of the particles constant (except at inflows and outflows). Further, the treatment of the collision operator should lead to the correct equilibrium density.

The generic simulation algorithm for a stochastic particle method is listed as Algorithm 1 and illustrated in Figure 4.

Algorithm 1: Generic particle Monte Carlo algorithm

initialize particle system $\{\boldsymbol{x}^l, \boldsymbol{\zeta}^l\}$, $l = 1, \ldots, N$
for each timestep, **do**
    sample particles at inflow **(a)**
    for each particle p, **do**
        move p **(b)** s.t. boundary conditions **(c.1, c.2)**
    sort particles into grid cells
    for each cell C, **do**
        sample moments of particles in C
        apply collision operator to particles in C **(d)**

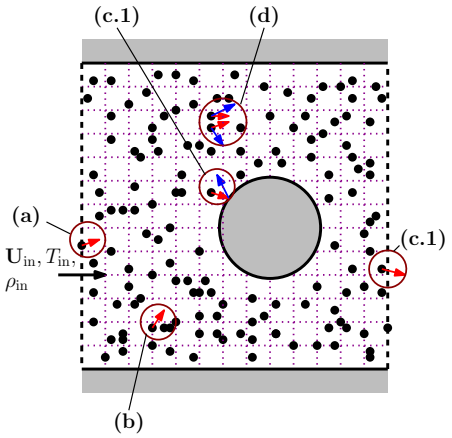


Figure 4: Illustration of generic particle Monte Carlo algorithm.

The remaining components to be discussed are the treatment of boundaries and, most importantly, of the collision term.

3.2 Boundary Conditions

Several types of boundary conditions need to be treatable by the particle algorithm to conduct meaningful simulations. The main difference to continuum methods is that boundary conditions are defined by fixing the phase density of particles entering the domain, instead of fixing values of the macroscopic fields and/or their derivatives. The specification of a boundary condition will not directly control the number of particles impinging on the boundary from within the simulation domain. A short overview of the most important boundary conditions follows.

At "stream" boundaries, new particles are sampled at each time step according to a prescribed phase density, while particles impinging from within the domain are removed from the simulation. A special case is the vacuum boundary, where the density is set to 0 and accordingly, no particles enter. The most common stream boundary assumes contact of the domain with a reservoir of molecules in equilibrium. Accordingly, density ρ, mean velocity vector $\boldsymbol{U}$ and temperature T need to be specified. The appropriate density to sample is the half-range equilibrium density given in Equation (25). A higher-order description is possible by assuming a Grad-type density, which additionally permits the specification of pressure tensor p_{ij} and heat flux vector $\boldsymbol{q}$, for a total of 13 parameters. In the present implementation, the algorithm due to Garcia and Wagner [GW06] is used to generate samples from the half-range Maxwellian. Samples from the Grad-type density are generated according to the algorithm due to Stephani et al. [SGV13]. Recall the derivation of Equation (24): the mean number of computational particles with phase density f and weights w to enter the domain through a boundary element of area A during a

time interval Δt is given by

$$\Delta\bar{N} = \frac{m}{w} A \Delta t \int |\boldsymbol{v}' \cdot \boldsymbol{n}| \mathcal{H}(-\boldsymbol{v}' \cdot \boldsymbol{n}) f(\boldsymbol{x}, \boldsymbol{c}') \mathrm{d}\boldsymbol{c}'. \tag{29}$$

The actual number of computational particles that enter during a given interval $(t, t + \Delta t)$ is a random number distributed according to the Poisson distribution [TG05]. Accordingly, during a simulation, at each time step and for each inflow are element, the number N of new particles to initialize is sampled from the probability mass function

$$f_N(n) = \frac{1}{n!} \left(\Delta\bar{N}\right)^n e^{-\Delta\bar{N}}. \tag{30}$$

At solid boundaries, particles should be prevented from penetrating the boundary, and change their state according to the specific wall model. The simplest wall model assumes perfectly elastic interactions of molecules with the wall, i.e., no energy or parallel momentum exchange takes place. This means that particles impinging on the wall are reflected specularly. Specifically, the only change to their state vector is the reversal of the wall normal component of their velocity. Specular boundary interaction is illustrated in Figure 5. One may conveniently implement this reversal in terms of a Householder reflection, that is, given the wall-normal unit vector $\boldsymbol{n}$, the particle velocity vector $\boldsymbol{v}$ is updated as

$$\boldsymbol{v} \leftarrow \boldsymbol{v} - 2(\boldsymbol{v} \cdot \boldsymbol{c})\boldsymbol{n}. \tag{31}$$

Conversely, a perfectly thermalizing wall causes diffuse reflection: here, particles impinging are assumed to undergo many collisions with the wall before reemerging with a state sampled from the prescribed PDF of the wall (typically a Maxwellian with given wall temperature). Here, the velocity of a particle after interaction with the boundary is assumed to be uncorrelated to its incoming velocity. The diffuse boundary is equivalent to an inflow boundary with zero mean velocity, oriented in the direction of the surface normal, where one particle is generated for each impinging one. Diffuse boundary interaction is illustrated in Figure 6.

A Maxwell boundary condition may be viewed as a linear combination of specular and diffuse reflection: one specifies an accomodation coefficient α, $0 \leq \alpha \leq 1$, which, on interaction of a particle with the boundary, defines the probability for diffuse reflection. Specular reflection occurs with probability $1 - \alpha$.

Finally, it is often convenient to have periodic and symmetry boundary conditions, to make use of symmetries in the simulated problem and thereby reduce the computational cost. Conveniently, a symmetry boundary is equal to a specular wall, since instead of reflecting the wall-normal component of the impinging particle's velocity, the action of the boundary may equivalently be thought of as consuming the impinging particle, while at the same time, the corresponding identical virtual particle with "mirrored" velocity crosses into the domain. Periodic boundary conditions are always proscribed on two parallel planes. When a particle impinges

on one plane, its state is left unaltered by the interaction, and it is simply displaced to the corresponding location on the other plane.

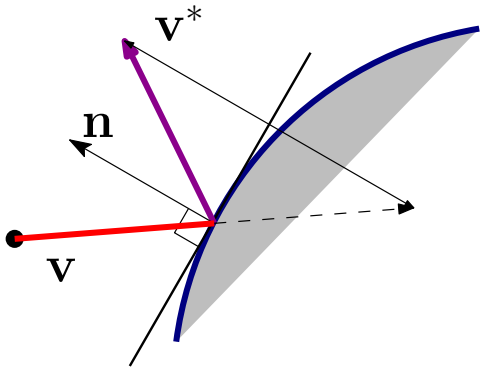


Figure 5: Specular boundary.

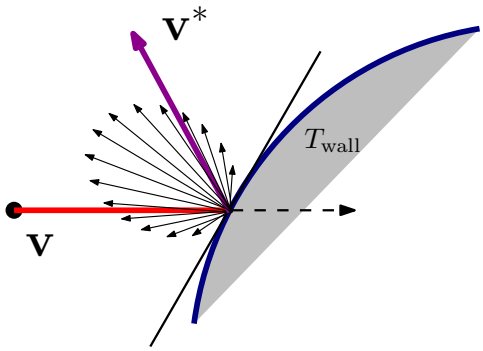


Figure 6: Diffuse boundary.

3.3 The Direct Simulation Monte Carlo (DSMC) Method

The most prominent particle Monte Carlo algorithm for the simulation of rarefied gas flows is certainly the Direct Simulation Monte Carlo (DSMC) method introduced by Graeme Bird [Bir94]. One of the fundamental assumptions underlying DSMC is that, for time steps smaller than the mean collision time, one may integrate the advection and collision terms of the Boltzmann equation sequentially rather than simultaneously. During a time step of length Δt, first, particle states are updated assuming constant position, that is, one solves a spatially homogeneous relaxation problem. Next, particle positions are updated assuming constant state, that is, collision-less flow is assumed.

In the absence of external forces, the advection term is trivial to handle, since all particles change their position simply according to their respective velocity. For particle with index l, position vector $\boldsymbol{\xi}^l$ and velocity vector $\boldsymbol{\mu}$, the position update reads $\boldsymbol{\xi}^l(t + \Delta t) = \boldsymbol{\xi}^l(t) + \boldsymbol{\mu}^l(t)\Delta t$.

To simulate the relaxation process, inter-particle collisions are simulated in a statistical way. In each grid cell, assumed to be smaller than the local mean free path, an appropriate number of particle pairs is selected for collision. For each pair, the relevant collision plane angle and impact parameter are chosen randomly. Next, the particle velocities are updated according to the dynamics of binary collisions imposed by the chosen inter-molecular potential, subject to the sampled collision parameters. Thus, DSMC conserves momentum and energy on a per-collision basis.

An attractive feature of DSMC is that various physical models for particle interactions, internal energy transfer, that is, inelastic collisions, etc., are relatively easy to implement, since the dynamics of the computational particles correspond directly to those of real molecules, albeit in a statistical sense. In this thesis, the Larsen-Borgnakke model is used to simulate inelastic collisions, see [Bir94, Chapters 11.3 and 11.4] for details.

The DSMC procedure may be directly derived from the kinetic master equation, and was proven to converge to the Boltzmann equation in the limit of infinite particles and infinitesimal time steps [IR88]. A crucial step in the algorithm is to determine the correct number of collisions to compute. Since the probability of a pair of particles to collide depends on their relative velocity, in a cell containing N_C computational particles, a naive algorithm would have to evaluate all $N_\mathrm{C}(N_\mathrm{C}-1)/2$ relative velocities. The algorithmic complexity may be reduced to $\mathcal{O}(N_\mathrm{C})$, if an acceptance rejection scheme is used instead: first, an upper bound for the collision frequency ν is determined as

$$\nu_\mathrm{max} = n(N_\mathrm{C}-1)/2 \times (\sigma_T \|\boldsymbol{g}\|)_\mathrm{max}, \tag{32}$$

where n is the number density in the cell, $\|\boldsymbol{g}\|$ is again the modulus of the relative velocity vector, and $\sigma_T = \pi d_{12}(\|\boldsymbol{g}\|)^2$ is the total collision cross-section in which d_{12} is the effective molecular diameter, which, in general, may depend on $\|\boldsymbol{g}\|$. Next, a corresponding number $N_\mathrm{C}^\mathrm{sel}$ of potential collision pairs are randomly selected among the particles in the cell. For each selected pair, the collision is only actually computed if the ratio of the product of actual relative velocity and collision cross-section of the pair to the maximum value is larger than a random number sampled uniformly from the interval $[0,1)$. Two alternatives for the computation of $N_\mathrm{C}^\mathrm{sel}$ are common, the Majorant Frequency (MF) scheme, and the No Time Counter (NTC) method:

Majorant Frequency Scheme. In this scheme [IR88], $N_\mathrm{C}^\mathrm{sel}$ is determined by successively sampling the inter-collision times τ_min^i from the appropriate exponential distribution, until the sum of sampled times exceeds the time step size. The number of select collision pairs then corresponds to the number of sampled times less one, i.e.,

$$N_\mathrm{C}^\mathrm{sel} = \underset{N}{\mathrm{argmin}} \sum_{i=1}^{N} \tau_\mathrm{min}^i, \quad \tau_\mathrm{min}^i \sim \mathrm{Exp}(\nu_\mathrm{max}), \quad \text{s.t.} \sum_{i=1}^{N+1} \tau_\mathrm{min}^i > \Delta t. \tag{33}$$

The remaining time $\Delta t - \sum_{i=1}^{N} \tau_\mathrm{min}^i$ is carried over to the next time step.

No Time Counter Method. The most widely used method for computing the number of selected pairs is the no time counter method invented by Bird [Bir94]. In its most recent form [Bir07], the number of pairs selected is simply computed as

$$N_\mathrm{C}^\mathrm{sel} = \nu_\mathrm{max} \times \Delta t. \tag{34}$$

Instead of sampling inter collision times, here, the mean value of the exponential distribution is used instead. If the number N_C of particles in the cell is larger than ≈ 10, the results produced by MF and NTC agree [IR88]. Note that, due to the stochastic nature of DSMC, $N_\mathrm{C} \gtrapprox 20$ is recommended regardless of which DSMC "flavor" is used.

3.4 The Fokker-Planck (FP) Algorithm

In situations where the mean free path is still significant, but smaller than the relevant flow length scales, DSMC becomes computationally very expensive, since the discrete nature of collisions needs to be resolved. The basic idea of the Fokker-Planck (FP) algorithm for the simulation of rarefied gas flows is to alleviate this deficiency by modeling the particle velocity evolution as a continuous stochastic process. Instead of having to resolve the discontinuous "jumps" in velocity during binary collisions, a continuous stochastic process may be integrated over much larger time scales.

Since the publication of the original FP model and simulation algorithm by Jenny et al. [JTH10] for monatomic gases, the method has undergone numerous developments and extensions. By extending the original linear drift model to a cubic model, Gorji et al. [GTJ11] improved the method to capture the correct evolution of heat fluxes. Later, Gorji and Jenny generalized the model to treat mixtures of monatomic gases [GJ12], and presented a more efficient algorithm [GJ14]. For this thesis, the model for diatomic gas flow introduced by the same authors [GJ15] was implemented. Development on the model is still ongoing, with the latest modification concerning improved accuracy for non-Maxwellian interaction potentials and the adherence to the H-theorem [GT16]. Since there exist many incremental changes to the method, the aim of this section is to provide a consistent derivation and summary of the parts of the "diatomic cubic" FP model and algorithm necessary for implementation. The subsequent sections follow, in part, Section 2 of the article [KJ17], as well as the original publications by Gorji, Jenny, et al. mentioned above.

3.4.1 The Fokker-Planck Model for the Phase Density Evolution

The Fokker-Planck model for rarefied gas flow assumes that the temporal evolution of f is governed—in the absence of external forces—by the following Fokker-Planck equation:

$$\left(\frac{D}{Dt} + \frac{\partial}{\partial c_k}\left\{A_k - \frac{1}{2}\frac{\partial}{\partial c_l}D_{kl}\right\}\right) f = 0, \tag{35}$$

where the operator

$$\frac{D}{Dt} = \frac{\partial}{\partial t} + v_i \frac{\partial}{\partial x_i} \tag{36}$$

signifies the convective derivative, and the vector $\boldsymbol{A}(\boldsymbol{x}, \boldsymbol{c}, t)$, as well as the positive definite tensor $D(\boldsymbol{x}, \boldsymbol{c}, t)$ denote drift and diffusion coefficients, respectively. This "drift and diffusion" ansatz is justified when enough collisions occur per unit of space and time, such that their individual effects on f are negligible, while their collective effect may be modeled stochastically.

In terms of Equation (15), the model assumption is that the Fokker-Planck

operator

$$S_{\mathrm{FP}} = \left(\frac{\partial}{\partial c_k} \left\{ -A_k + \frac{1}{2} \frac{\partial}{\partial c_l} D_{kl} \right\} \right) f \tag{37}$$

is a reasonable approximation to the full Boltzmann collision operator $S_{\mathrm{Boltz.}}$. Specifically, in the FP method, consistency with the Boltzmann collision operator is enforced on the level of moments:

For notational convenience, let the operator $\langle \cdot \rangle$ denote the generalized expectation w.r.t. the phase density, i.e., let

$$\langle g(\boldsymbol{c}) \rangle \equiv \int g(\boldsymbol{c}') f(\cdot, \boldsymbol{c}') \mathrm{d}\boldsymbol{c}' \tag{38}$$

denote a generalized moment of f. An evolution equation for $\langle g(\boldsymbol{c}) \rangle$ may be obtained by multiplying Equation (15) by g and integrating over the state space, that is,

$$\frac{\partial}{\partial t} \langle g(\boldsymbol{c}) \rangle + \frac{\partial}{\partial x_k} \langle g(\boldsymbol{c}) v_k \rangle = \underbrace{\int g(\boldsymbol{c}') S(\boldsymbol{c}') \mathrm{d}\boldsymbol{c}'}_{\Pi_g}, \tag{39}$$

where Π_g is referred to as the "production term" of the quantity $\langle g \rangle$. The basis of the FP method is to choose $\boldsymbol{A}$ and D such that, under certain assumptions, $\int g(\boldsymbol{c}') S_{\mathrm{FP}}(\boldsymbol{c}') \mathrm{d}\boldsymbol{c}' = \int g(\boldsymbol{c}') S_{\mathrm{Boltz}}(\boldsymbol{c}') \mathrm{d}\boldsymbol{c}'$ holds for relevant g.

Some relevant production terms of the Boltzmann equation, in the absence of internal degrees of freedom, i.e., with $d_c = 3$, assuming Maxwellian interaction potentials (which have infinite range), are given in [Str05, Equation 5.23], viz.

$$\Pi_1 \equiv 0 \qquad \text{(mass conservation)} \tag{40a}$$

$$\Pi_{v_i} \equiv 0 \qquad \text{(momentum conservation)} \tag{40b}$$

$$\Pi_{e(\boldsymbol{c})} \equiv 0 \qquad \text{(energy conservation)} \tag{40c}$$

$$\Pi_{V_{\langle i} V_{j \rangle}} = -\frac{p}{\mu} \sigma_{ij} \qquad \text{(stress tensor)} \tag{40d}$$

$$\Pi_{\|\boldsymbol{V}\|^2 V_i} = -\frac{4}{3} \frac{p}{\mu} q_i, \qquad \text{(tr. heat fluxes)} \tag{40e}$$

where $\mu = \mu(T)$ is the viscosity, which, for power law potentials, takes the form $\mu(T) = \mu_{\mathrm{ref}} (T / T_{\mathrm{ref}})^{\omega}$ with parameter ω and where μ_{ref} is the viscosity at temperature T_{ref}. While the above expressions for the production terms are exact for Maxwellian molecules, most simulations are conducted using different molecular models, such as hard spheres (HS) or the variable soft sphere (VSS) model, cf. [Bir94, Sections 2.5–2.10]. In these cases, the Equations (40) are used as an approximation, together with the correct viscosity $\mu(T)$. Additionally, a first-order correction based on a Grad-type density assuming hard spheres for $\Pi_{V_{\langle i} V_{j \rangle}}$ [GT17; GT12] is used, viz.

$$\Pi^*_{V_{\langle i} V_{j \rangle}} = -\frac{p}{\mu} \left(\sigma_{ij} - \frac{1}{28} \frac{R_{ij}}{\theta} \right), \tag{41}$$

where $R_{ij} = \langle \|\boldsymbol{V}\|^2 V_i V_j \rangle - \frac{1}{3}\langle \|\boldsymbol{V}\|^4 \rangle \delta_{ij} - 7\theta\sigma_{ij}$.

In the general case of molecules with internal degrees of freedom, closed form expressions for the Π terms do not exist, and approximations are again obtained by assuming a Grad-type density, see [GJ13], as well as [McC70], as cited in [GJ13].

The production terms implied by the Fokker-Planck operator, with generic coefficients $\boldsymbol{A}$ and D, may be evaluated by (repeated) integration by parts to bring all terms containing derivatives into divergence form, i.e., $\int \nabla_{\boldsymbol{c}}(\tilde{g}(\boldsymbol{c}, \boldsymbol{A}, D)f)\mathrm{d}\boldsymbol{c}$, which, under the usual assumption $f \to 0$ as any $c_i \to \pm\infty$, vanish due to the divergence theorem.

The resulting production terms read

$$\Pi_1^{\mathrm{FP}} = 0 \qquad \text{(mass)} \tag{42a}$$

$$\Pi_{v_i}^{\mathrm{FP}} = \langle A_i \rangle \qquad \text{(momentum)} \tag{42b}$$

$$\Pi_{e(\boldsymbol{c})}^{\mathrm{FP}} = \langle (\partial_{c_k} e) A_k \rangle + \frac{1}{2}\langle (\partial_{c_k}\partial_{c_l} e) D_{kl} \rangle \qquad \text{(energy)} \tag{42c}$$

$$\Pi_{V_{\langle i}V_{j\rangle}}^{\mathrm{FP}} = \langle A_i V_j \rangle + \langle A_j V_i \rangle + \frac{1}{2}(\langle D_{ij} \rangle + \langle D_{ji} \rangle) - \frac{1}{3}(2\langle A_s V_s \rangle + \langle D_{ss} \rangle)\delta_{ij} \qquad \text{(stress tensor)} \tag{42d}$$

$$\Pi_{\|\boldsymbol{V}\|^2 V_i}^{\mathrm{FP}} = 2\langle A_s V_s V_i \rangle + \langle A_i V_s V_s \rangle + \langle V_i D_{ss} \rangle + \langle V_s D_{si} \rangle + \langle V_s D_{is} \rangle, \qquad \text{(tr. heat fluxes)} \tag{42e}$$

where $i, j, s = 1, 2, 3$ and $k, l = 1, \ldots, d_c$.

3.4.2 The Cubic Fokker-Planck Model for Diatomic Gas Flows

Using a cubic ansatz for $A_{\mathrm{tr},i}$, the FP evolution of the moments of the phase density up to the heat fluxes can be made consistent with the evolution implied by the Boltzmann equation for Maxwellian molecules [GTJ11; GJ15], in the case of a monatomic gas, and for a Grad-type density in the case of a gas with internal degrees of freedom [GJ13].

In this thesis, the model presented in [GJ13] is employed for the simulation of flows of diatomic gas with two rotational and two vibrational internal modes, for a total state space dimension $d_c = 7$. The components of the state space coordinate vector $\boldsymbol{c} \in \mathbb{R}^7$ are designated

$$\boldsymbol{c} = \begin{pmatrix} v_1 & v_2 & v_3 & \omega_1 & \omega_2 & \chi_1 & \chi_2 \end{pmatrix}^\top. \tag{43}$$

The internal degrees of freedom are assumed to be normalized such that the specific internal thermal energy of a particle in state $\boldsymbol{c}$ may be calculated as

$$e_{\mathrm{int}}(\boldsymbol{c}) = e_{\mathrm{rot}}(\boldsymbol{\omega}) + e_{\mathrm{vib}}(\boldsymbol{\chi}) = \frac{1}{2}\|\boldsymbol{\omega}\|^2 + \frac{1}{2}\|\boldsymbol{\chi}\|^2. \tag{44}$$

Consequently, the corresponding equilibrium phase density Equation (23) reads

$$f_{|E}(\boldsymbol{x}, \boldsymbol{c}) = \rho(\boldsymbol{x})\left(\frac{1}{2\pi\theta(\boldsymbol{x})}\right)^{\frac{7}{2}} \exp\left(-\frac{\|\boldsymbol{V}(\boldsymbol{x}, \boldsymbol{v})\|^2 + \|\boldsymbol{\omega}\|^2 + \|\boldsymbol{\chi}\|^2}{2\theta(\boldsymbol{x})}\right). \tag{45}$$

The drift coefficient vector and the diffusion coefficient tensor take the following generic forms:

$$\boldsymbol{A}(\boldsymbol{x},\boldsymbol{c},t) = \Big(A_{\mathrm{tr},1}\ A_{\mathrm{tr},2}\ A_{\mathrm{tr},3}\ A_{\mathrm{rot},1}\ A_{\mathrm{rot},2}\ A_{\mathrm{vib},1}\ A_{\mathrm{vib},2}\Big)^{\top}, \tag{46}$$

$$D(\boldsymbol{x},t) = \mathrm{diag}\Big(\Big(D^2_{\mathrm{tr}}\ D^2_{\mathrm{tr}}\ D^2_{\mathrm{tr}}\ D^2_{\mathrm{rot}}\ D^2_{\mathrm{rot}}\ D^2_{\mathrm{vib}}\ D^2_{\mathrm{vib}}\Big)\Big). \tag{47}$$

Note that the diffusion is chosen to not explicitly depend on the state coordinate $\boldsymbol{c}$. For the translational modes, a cubic ansatz for the drift is made:

$$A_{\mathrm{tr},i} = \Gamma_i - \nu V_i + \tilde{c}_{ir}V_r + \gamma_i V_r V_r + \Lambda V_i V_r V_r, \tag{48}$$

with coefficients $\tilde{c} \in \mathbb{R}^{3\times 3}$, $\tilde{c}_{ij} \equiv \tilde{c}_{ji}$, $\boldsymbol{\Gamma}, \boldsymbol{\gamma} \in \mathbb{R}^3$ and $\Lambda \in \mathbb{R}$, and where the inverse relaxation time ν is given by $\nu = p/2\mu$ as a function of viscosity μ and pressure p. This choice of ν gives the correct relaxation of stresses for a linear translational drift, assuming Maxwellian molecules. The negative scalar Λ is to stabilize the drift and should vanish in equilibrium, a condition satisfied by the ad-hoc choice

$$\Lambda = -\frac{\nu}{(u^2)^4}\left|\det\left(u^0_{ij} - \frac{1}{3}u^2\delta_{ij}\right)\right|, \tag{49}$$

where here and in the following, for convenience of notation, the abbreviation

$$u^s_{i_1\ldots i_k} = \langle \|\boldsymbol{V}\|^s V_{i_1}\cdots V_{i_k}\rangle \tag{50}$$

is used. Note that $u^s_{ij} \equiv u^s_{ji}$ and $u^s_{rr} \equiv u^{s+2}$.

For the internal degrees of freedom, the drift is modeled by a linear ansatz, i.e.,

$$\begin{aligned} A_{\mathrm{rot},i} &= -\frac{1}{2}\nu_{\mathrm{rot}}\omega_i, \\ A_{\mathrm{vib},i} &= -\frac{1}{2}\nu_{\mathrm{vib}}\chi_i \end{aligned} \qquad i = 1,2. \tag{51}$$

Production Terms implied by the Cubic Model. Assuming $\boldsymbol{A}$ to be in the form given by Equations (48) and (51), the production terms (Equation (42)) read

$$\Pi^{\mathrm{FP}}_{v_i} = \langle A_{\mathrm{tr},i}\rangle = \rho\Gamma_i + \gamma_i u^2 + \Lambda u_i^2 \qquad \text{(momentum)} \tag{52a}$$

$$\begin{aligned}\Pi^{\mathrm{FP}}_{c_k c_k} &= 2\langle A_k c_k\rangle + \langle D_{kk}\rangle \\ &= -4\nu\rho\varepsilon_{\mathrm{tr}} + 2\tilde{c}_{sr}u^0_{rs} + 2\gamma_s u_s^2 + 2\Lambda u^4 \\ &\quad - 2\nu_{\mathrm{rot}}\rho\varepsilon_{\mathrm{rot}} - 2\nu_{\mathrm{vib}}\rho\varepsilon_{\mathrm{vib}} \\ &\quad + \rho\Big(3D^2_{\mathrm{tr}} + 2D^2_{\mathrm{rot}} + 2D^2_{\mathrm{vib}}\Big)\end{aligned} \qquad \text{(energy)} \tag{52b}$$

$$\begin{aligned}\Pi^{\mathrm{FP}}_{V_{\langle i}V_{j\rangle}} &= \langle A_{\mathrm{tr},i}V_j\rangle + \langle A_{\mathrm{tr},j}V_i\rangle - \frac{2}{3}\langle A_s V_s\rangle\delta_{ij} \\ &= -2\nu u^0_{ij} + \tilde{c}_{ir}u^0_{rj} + \tilde{c}_{jr}u^0_{ri} \\ &\quad + \gamma_i u_j^2 + \gamma_j u_i^2 + 2\Lambda u^2_{ij} \\ &\quad - \frac{2}{3}\Big(-\nu u^2 + \tilde{c}_{sr}u^0_{rs} + \gamma_s u_s^2 + \Lambda u^4\Big)\delta_{ij}\end{aligned} \qquad \text{(stress tensor)} \tag{52c}$$

$$\begin{aligned}\Pi^{\mathrm{FP}}_{\|\boldsymbol{V}\|^2 V_i} &= 2\langle A_{\mathrm{tr},s}V_s V_i\rangle + \langle A_{\mathrm{tr},i}V_s V_s\rangle \\ &= 2\Gamma_s u^0_{si} + \Gamma_i u^2 - 3\nu u_i^2 + 2\tilde{c}_{sr}u^0_{rsi} + \tilde{c}_{ir}u_r^2 \\ &\quad + 2\gamma_s u^2_{si} + \gamma_i u^4 + 3\Lambda u_i^4,\end{aligned} \qquad \text{(tr. heat fluxes)} \tag{52d}$$

where $w_{\mathrm{rot}} = \frac{1}{2}\langle\|\boldsymbol{\omega}\|^2\rangle$ and $w_{\mathrm{vib}} = \frac{1}{2}\langle\|\boldsymbol{\chi}\|^2\rangle$ denote the density of rotational and vibrational energy, respectively.

Evolution of Internal Energy. The evolution of rotational and vibrational energy is assumed to be governed by the Landau-Teller approximation [GJ13; NT08], i.e., homogeneous relaxation of average specific thermal rotational and vibrational energy are assumed to follow simple exponential decays, that is,

$$\frac{\mathrm{d}}{\mathrm{d}t}\varepsilon_{\mathrm{rot}}(t) = -\nu_{\mathrm{rot}}\varepsilon_{\mathrm{rot}}(t) + \nu_{\mathrm{rot}}\varepsilon^*_{\mathrm{rot}|E} \tag{53}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\varepsilon_{\mathrm{vib}}(t) = -\nu_{\mathrm{vib}}\varepsilon_{\mathrm{vib}}(t) + \nu_{\mathrm{vib}}\varepsilon^*_{\mathrm{vib}|E}, \tag{54}$$

where $\varepsilon^*_{\mathrm{rot}|E}$ and $\varepsilon^*_{\mathrm{vib}|E}$ denote the respective equilibrium values. For the rotational modes, the classical value $\varepsilon_{\mathrm{rot}|E}* = \varepsilon_{\mathrm{rot}|E} = \theta$, which is implied by Equation (45), is used, while for the vibrational modes, $\varepsilon^*_{\mathrm{vib}|E}$ is set to the equilibrium average specific energy in a heat bath of quantized harmonic oscillators, viz.

$$\varepsilon^*_{\mathrm{vib}|E} = \mathcal{L}\theta \tag{55}$$

$$\text{with} \quad \mathcal{L} = \frac{(\vartheta_v/T)}{\exp(\vartheta_v/T) - 1}, \tag{56}$$

where $\vartheta_v = h\nu_0/k_B$ is the characteristic vibrational temperature, given in terms of Planck's constant h and the fundamental frequency ν_0 of the diatomic molecule's vibrational mode.

Comparing Equations (53) and (42c) leads to the following diffusion coefficients:

$$D_{\text{rot}}^2 = \nu_{\text{rot}}\theta, \tag{57}$$
$$D_{\text{vib}}^2 = \nu_{\text{vib}}\mathcal{L}\theta. \tag{58}$$

The inverse relaxation times $\nu_{\text{rot}} = \nu_{\text{coll}}/Z_{\text{rot}}$ and $\nu_{\text{vib}} = \nu_{\text{coll}}/Z_{\text{vib}}$ are proportional to the mean collision frequency $\nu_{\text{coll}} = \frac{4}{\pi}\frac{p}{\mu}$ and the rotational and vibrational collision numbers Z_{rot} and Z_{vib}, respectively. The model equations for Z_{rot} and Z_{vib} are [Par59]

$$Z_{\text{rot}} = Z_{\text{rot}}^{\infty}\left(1 + \frac{\sqrt{\pi^3}}{2}\left(\frac{T^*}{T}\right)^{\frac{1}{2}} + \left(\frac{\pi^2}{4} + \pi\right)\left(\frac{T^*}{T}\right)\right)^{-1} \tag{59}$$

$$\text{and} \quad Z_{\text{vib}} = \left(\frac{C_1}{T^{\omega}}\right)\exp\left(C_2 T^{-\frac{1}{3}}\right), \tag{60}$$

respectively, where Z_{rot}^{∞}, T^*, C_1 and C_2 are model constants and ω is the exponent in the viscosity power law $\mu = \mu_{\text{ref}}(T/T_{\text{ref}})^{\omega}$.

It is worth noting that, while the above choices for ν_{vib} and D_{vib} lead to the desired Landau-Teller relaxation rate of vibrational energy, conserving the energy density $w = \frac{1}{2}\langle\|\boldsymbol{v}\|^2\rangle + \frac{1}{2}\langle\|\boldsymbol{c}_{\text{int}}\|^2\rangle$ will still lead to the classical equilibrium density given by Equation (45). To account for this discrepancy at inflows and diffusive boundaries, particles are generated here according to the non-equilibrium joint normal density

$$f_{|E^*}(\boldsymbol{c}) = \rho\left(\frac{1}{2\pi\theta}\right)^{\frac{5}{2}}\exp\left(-\frac{\|\boldsymbol{V}\|^2 + \|\boldsymbol{\omega}\|^2}{2\theta}\right)\left(\frac{1}{2\pi\mathcal{L}\theta}\right)\exp\left(-\frac{\|\boldsymbol{\chi}\|^2}{2\mathcal{L}\theta}\right), \tag{61}$$

instead of using Equation (45).

Constitutive Equations. The coefficients $\boldsymbol{\Gamma}$, $\tilde{c}$, $\boldsymbol{\gamma}$ and D_{tr} are determined from a system of constitutive equations, derived by comparing the production terms implied by the FP operator to those derived from the Boltzmann equation.

Conservation of momentum, i.e., comparing Equations (40b) and (52a), immediately yields

$$\Gamma_i = -\frac{1}{\rho}\left(\gamma_i u^2 + \Lambda u_i^2\right). \tag{62}$$

The desired production term for the stress tensor evolution is enforced by comparing Equations (41) and (52c), viz.

$$\begin{aligned} &-2\nu u_{ij}^0 + \tilde{c}_{ir}u_{rj}^0 + \tilde{c}_{jr}u_{ri}^0 + \gamma_i u_j^2 + \gamma_j u_i^2 + 2\Lambda u_{ij}^2 \\ &\qquad - \frac{2}{3}\left(-\nu u^2 + \tilde{c}_{sr}u_{rs}^0 + \gamma_s u_s^2 + \Lambda u^4\right)\delta_{ij} \stackrel{!}{=} \Pi_{V_{\langle i}V_{j\rangle}}^*. \end{aligned} \tag{63a}$$

A solution to Equation (63a) may be found by requiring

$$\tilde{c}_{ir}u_{rj}^0 + \tilde{c}_{jr}u_{ri}^0 + \gamma_i u_j^2 + \gamma_j u_i^2 + 2\Lambda u_{ij}^2 \stackrel{!}{=} \Pi_{V_{\langle i}V_{j\rangle}}^* + 2\nu\sigma_{ij}. \tag{63b}$$

Since the right hand side of Equation (63b) has zero trace, it follows that

$$\tilde{c}_{sr} u^0_{rs} + \gamma_s u_s^2 + \Lambda u^4 \equiv 0, \tag{64}$$

and consequently, Equation (63b) is recovered from Equation (63a).

Inserting Equations (57), (58) and (64) into Equation (52b) and comparing to Equation (40c), i.e., enforcing conservation of energy, yields the diffusion coefficient for the translational degrees of freedom as

$$D^2_{tr} = \frac{2}{3}\Big(2\nu\varepsilon_{\text{tr}} + \nu_{\text{rot}}\Big(\varepsilon_{\text{rot}} - \varepsilon^*_{\text{rot}|E}\Big) + \nu_{\text{vib}}\Big(\varepsilon_{\text{vib}} - \varepsilon^*_{\text{vib}|E}\Big)\Big), \tag{65}$$

where, as before, the rotational and vibrational stationary average specific thermal energies are $\varepsilon^*_{\text{rot}|E} = \theta$ and $\varepsilon^*_{\text{vib}|E} = \mathcal{L}\theta$, respectively.

The remaining three equations to close the system for the 9 unknowns $\tilde{c}$, $\boldsymbol{\gamma}$ are provided by comparing the FP production term for the translational heat flux vector, Equations (52d), together with the previous result for the coefficients Γ_i, Equation (62), to the corresponding term derived from the Boltzmann equation, i.e.,

$$\begin{aligned} -3\nu u_i^2 + 2\tilde{c}_{sr} u^0_{rsi} + \tilde{c}_{ir} u_r^2 \\ + 2\gamma_s\left(u^2_{si} - \frac{1}{\rho} u^0_{si} u^2\right) + \gamma_i\left(u^4 - \frac{1}{\rho} u^2 u^2\right) \\ + \Lambda\left(3u_i^4 - 2\frac{1}{\rho} u_s^2 u^0_{si} - \frac{1}{\rho} u_i^2 u^2\right) \overset{!}{=} \Pi^*_{\|\boldsymbol{V}\|^2 V_i}. \end{aligned} \tag{66}$$

For a diatomic gas, under certain assumptions (cf. [GJ13] and references therein for details), $\Pi^*_{\|\boldsymbol{V}\|^2 V_i}$ can be derived as

$$\Pi^*_{\|\boldsymbol{V}\|^2 V_i} = -\frac{2}{3}\frac{p}{\mu} u_i^2 + \frac{5}{3}(\nu_{\text{rot}} q_{\text{rot},i} + \nu_{\text{vib}} q_{\text{vib},i}) - \frac{5}{9R}(\nu_{\text{rot}} c_{\text{rot}} + \nu_{\text{vib}} c_{\text{vib}}) u_i^2, \tag{67}$$

where $q_{\text{rot},i} = \frac{1}{2}\langle \|\boldsymbol{\omega}\|^2 V_i\rangle$, $q_{\text{vib},i} = \frac{1}{2}\langle \|\boldsymbol{\chi}\|^2 V_i\rangle$ are the flux vectors of rotational and vibrational energy, respectively, and the specific heats c_{rot} and c_{vib} are given by [Kre10, Equation (5.44)]

$$\begin{aligned} c_{\text{rot}} &= R \\ \text{and} \quad c_{\text{vib}} &= R\left(\frac{(\vartheta_v/2T)}{\sinh(\vartheta_v/2T)}\right)^2. \end{aligned} \tag{68}$$

Equations (63b) and (66) together form a 9×9 linear system of equations, provided in Appendix A.1, which, at each time step and in every cell, is solved numerically for the vector of unknowns

$$\left(\tilde{\boldsymbol{c}}^\top \ \boldsymbol{\gamma}^\top\right) = \left(\tilde{c}_{11} \ \tilde{c}_{12} \ \tilde{c}_{13} \ \tilde{c}_{22} \ \tilde{c}_{23} \ \tilde{c}_{33} \ \gamma_1 \ \gamma_2 \ \gamma_3\right) \tag{69}$$

with the quantities ρ, $u^s_{i_1\dots i_n}$, θ, ε_{rot}, ε_{vib}, etc. estimated in terms of (functions of) ensemble averages of the particles located in the respective cell.

To summarize, the cubic Fokker-Planck model for rarefied diatomic gas flows, due to Gorji and Jenny [GJ13], subject to the system of constitutive equations, has the following drift and diffusion coefficients:

$$\begin{aligned}
A_{\text{tr},i} &= -\nu V_i + \tilde{c}_{ir}V_r \\
&\quad + \gamma_i\Big(V_rV_r - u^2\big/\rho\Big) \\
&\quad + \Lambda\Big(V_iV_rV_r - u_i^2\big/\rho\Big) \\
A_{\text{rot},j} &= -\frac{1}{2}\nu_{\text{rot}}\omega_j \\
A_{\text{vib},j} &= -\frac{1}{2}\nu_{\text{vib}}\chi_j
\end{aligned}
\qquad
\begin{aligned}
D^2_{tr} &= \frac{2}{3}\Big(2\nu\varepsilon_{\text{tr}} + \nu_{\text{rot}}(\varepsilon_{\text{rot}} - \theta) \\
&\qquad + \nu_{\text{vib}}(\varepsilon_{\text{vib}} - \theta\mathcal{L})\Big) \\
D^2_{\text{rot}} &= \nu_{\text{rot}}\theta \\
D^2_{\text{vib}} &= \nu_{\text{vib}}\theta\mathcal{L}
\end{aligned}
\qquad (70)$$

$$i, r = 1, 2, 3 \quad j = 1, 2.$$

3.4.3 Itō Processes

Using Itō calculus, the evolution of f according to Equation (35) can be transformed into an equivalent system of stochastic differential equations (SDEs), see [Gar09, Chapter 5]. The SDEs govern the evolution of the random phase-space vector $(\boldsymbol{\xi}, \boldsymbol{\zeta}) \in \mathbb{R}^{3+7}$ with components

$$\boldsymbol{\zeta} = \Big(\mu_1\ \mu_2\ \mu_3\ \Omega_1\ \Omega_2\ \Xi_1\ \Xi_2\Big)^\top. \qquad (71)$$

Note that the probability density (PDF) of $\boldsymbol{\zeta}$, $f_{\boldsymbol{\zeta}}$, is, by definition, related to the full phase space density f as

$$f_{\boldsymbol{\zeta}}(\boldsymbol{c}; \boldsymbol{x}, t) = \frac{1}{\rho(\boldsymbol{x}, t)} f(\boldsymbol{c}, \boldsymbol{x}; t). \qquad (72)$$

The SDEs governing the evolution of $\boldsymbol{\zeta}$ and $\boldsymbol{\xi}$ read as follows:

$$\begin{aligned}
\mathrm{d}\zeta_l &= A_l(\boldsymbol{\zeta}(t), t)\mathrm{d}t + \lambda_{lk}(t)\mathrm{d}W_k \\
\text{and} \quad \mathrm{d}\xi_i &= \mu_i \mathrm{d}t,
\end{aligned} \qquad (73)$$

where $k, m, l = 1, \dots, d_c$, $i = 1, 2, 3$, λ_{lm} is the square root of the diffusion matrix $D_{lm} = \lambda_{lk}\lambda_{km}$, and $\mathrm{d}W_m$ is an increment of a stochastic Wiener process $W_m(t)$, satisfying $\mathrm{E}[\mathrm{d}W_j] \equiv 0$ and $\mathrm{E}[\mathrm{d}W_l \mathrm{d}W_m] \equiv \delta_{lm}\mathrm{d}t$. Define the fluctuating velocity vector $\boldsymbol{M} = \boldsymbol{\mu} - \mathrm{E}_\zeta[\boldsymbol{\mu}] = \boldsymbol{\mu} - \boldsymbol{U}$, with corresponding phase space coordinate

vector $\boldsymbol{V}$. For the cubic model given by Equations (70), the Equations (73) read

$$\begin{aligned} \mathrm{d}\mu_i = \mathrm{d}M_i = \Big[&-\nu M_i + \tilde{c}_{is} M_s + \gamma_i \Big(M_s M_s - u^2 \big/ \rho \Big) \\ &+ \Lambda \Big(M_i M_s M_s - u_i^2 \big/ \rho \Big) \Big] \mathrm{d}t + D_{\mathrm{tr}} \mathrm{d}W_i, \\ \mathrm{d}\Omega_j &= -\frac{1}{2} \nu_{\mathrm{rot}} \Omega_j \mathrm{d}t + D_{\mathrm{rot}} \mathrm{d}W_j, \\ \mathrm{d}\Xi_j &= -\frac{1}{2} \nu_{\mathrm{vib}} \Xi_j \mathrm{d}t + D_{\mathrm{vib}} \mathrm{d}W_j, \\ \text{and} \quad \mathrm{d}\xi_i &= \mu_i \mathrm{d}t, \end{aligned} \tag{74}$$

where $i, s = 1, 2, 3$, and $j = 1, 2$.

3.4.4 Solution Algorithm

The exact correspondence of the FP equation to the Langevin equation for individual phase-space point-masses given above (Equation (74)) is the main advantage of particle simulations based on the FP model compared to DSMC. These continuous stochastic processes may be accurately integrated in time, which can be done without having to honor collisional space and time scales as in DSMC. Instead, it is sufficient to resolve the gradients of the macroscopic properties, in order to justify the assumption that the values of the coefficients in the FP model remain constant along a particle's trajectory during a given time step.

In order to simulate the evolution of a finite ensemble of computational particles with indices l and state vectors $\boldsymbol{\zeta}^l$ according to Equations (74), the updated state vector $\boldsymbol{\zeta}^{l,\,n+1} = \boldsymbol{\zeta}^l(t = t^{n+1} = t^n + \Delta t)$ is approximated under the above assumption that the macroscopic coefficients remain fixed during the time step.

First consider the translational degrees of freedom: Let the non-linear part of the drift be abbreviated as

$$N_i = \tilde{c}_{is} M_s + \gamma_i \Big(M_s M_s - u^2 \big/ \rho \Big) + \Lambda \Big(M_i M_s M_s - u_i^2 \big/ \rho \Big), \tag{75}$$

so that the corresponding SDE for M may be written as

$$\mathrm{d}M_i = [-\nu M_i + N_i] \mathrm{d}t + D_{\mathrm{tr}} \mathrm{d}W_i. \tag{76}$$

Itō's lemma states the following [Øks03, Theorem 4.2.1]: Let

$$\mathrm{d}X_i(t) = \mu_i(t) \mathrm{d}t + \sigma_i \mathrm{d}W_i(t) \tag{77}$$

be an n-dimensional Itō process. Let $g(t, \boldsymbol{x})^\top = \Big(g_1(t, \boldsymbol{x}) \; \ldots \; g_p(t, \boldsymbol{x}) \Big)$ be a C^2 map $[0, \infty) \times \mathbb{R}^n \to \mathbb{R}^p$. Then the process

$$\boldsymbol{Y}(t, \omega) = g(t, \boldsymbol{X}(t)) \tag{78}$$

is again an Itō process, whose component number k, Y_k, is given by

$$\mathrm{d}Y_k = \partial_t g_k \mathrm{d}t + \partial_{x_l} g_k \mathrm{d}X_l + \frac{1}{2}\partial_{x_l}\partial_{x_m} g_k \mathrm{d}X_l \mathrm{d}X_m \tag{79}$$

where $\mathrm{d}W_i \mathrm{d}W_j = \delta_{ij}\mathrm{d}t$, $\mathrm{d}W_i \mathrm{d}t = \mathrm{d}t\mathrm{d}W_i = 0$, and summation over the indices $l, m = 1, \ldots, n$ is implied.

Consider the substitution $g_i := e^{\nu t} M_i$ (integrating factor method). From Equation (77), it follows that

$$\mathrm{d}g_i = e^{\nu t} N_i \mathrm{d}t + e^{\nu t} D_{\mathrm{tr}} \mathrm{d}W_i \tag{80}$$

with formal solution

$$M_i(t+\Delta t) = e^{-\nu \Delta t} M_i(t) + \underbrace{\int_t^{t+\Delta t} e^{\nu(s-t-\Delta t)} N_i \mathrm{d}s}_{I} + \underbrace{\int_t^{t+\Delta t} e^{\nu(s-t-\Delta t)} D_{\mathrm{tr}} \mathrm{d}W_i(s)}_{II}. \tag{81}$$

To estimate term I in Equation (81), it is assumed that N_i remains constant during the time interval $[t, t+\Delta t]$, resulting in

$$I \approx N_i(t) \frac{1}{\nu}\Big(1 - e^{-\nu \Delta t}\Big). \tag{82}$$

To estimate term II in Equation (81), one may use Lemma I in [Cha43], which reads as follows:

Let

$$R = \int_t^{t+\Delta t} g(s) \mathrm{d}W(s). \tag{83a}$$

Then,

$$R \sim \mathcal{N}\Big(0, \sigma_R^2\Big), \quad \text{with} \quad \sigma_R^2 = \int_t^{t+\Delta t} g(s)^2 \mathrm{d}s. \tag{83b}$$

In term II, $g(s) = e^{\nu(s-t-\Delta t)} D_{\mathrm{tr}}$, and consequently,

$$II \ \sim \mathcal{N}\left(0, \frac{D_{\mathrm{tr}}^2}{2\nu}\Big(1 - e^{-2\nu \Delta t}\Big)\right), \tag{84}$$

see, e.g., Section 4.5.6 in [Gar09].

Based on the intermediate results above, the numerical time step update of $\boldsymbol{M}$ is proposed as

$$\widetilde{M}_i^{n+1} = e^{-\nu \Delta t} M_i^n + \frac{1}{\nu} e^{-\nu \Delta t}\Big(1 - e^{-\nu \Delta t}\Big) N_i^n + \sqrt{\frac{D_{\mathrm{tr}}^2}{2\nu}(1 - e^{-2\nu \Delta t})}\, \psi_{\mathrm{tr},i}, \tag{85}$$

where $\boldsymbol{\psi}_{\mathrm{tr}}$ is a vector of independent standard normal random numbers. Note that the non-linear term contains an additional factor $e^{-\nu \Delta t}$, which is added to

recover the consistent limit $\lim_{\Delta t\to\infty}(M(t+\Delta t)-M(t)) \sim \mathcal{N}\big(0, \frac{D_{\mathrm{tr}}^2}{2\nu}\big)$. As $\Delta t \to 0$, $N_i^n \frac{1}{\nu} e^{-\nu\Delta t}\big(1-e^{-\nu\Delta t}\big) \to \Delta t N_i$, so that the forward Euler scheme is recovered.

Proceeding similarly for the internal degrees of freedom, one may derive the update equations

$$\omega_i^{n+1} = \omega_i^n e^{-\frac{1}{2}\nu_{\mathrm{rot}}\Delta t} + \sqrt{\frac{D_{\mathrm{rot}}^2}{\nu_{\mathrm{rot}}}(1-e^{-\nu_{\mathrm{rot}}\Delta t})}\,\psi_{r,i} \tag{86}$$

$$\text{and} \quad \xi_i^{n+1} = \xi_i^n e^{-\frac{1}{2}\nu_{\mathrm{vib}}\Delta t} + \sqrt{\frac{D_{\mathrm{vib}}^2}{\nu_{\mathrm{vib}}}(1-e^{-\nu_{\mathrm{vib}}\Delta t})}\,\psi_{\mathrm{v},i}, \tag{87}$$

where $\boldsymbol{\psi}_{\mathrm{r}}$ and $\boldsymbol{\psi}_{\mathrm{v}}$ are again vectors of independent standard normal distributed random numbers.

In order to make the scheme conservative, the translational degrees of freedom are scaled by a correction factor $C_f = \sqrt{\varepsilon_{\mathrm{tr}}^{n+1} \big/ \bar{\widetilde{\varepsilon}}_{\mathrm{tr}}^{n+1}}$, where the overbar over a quantity denotes the corresponding ensemble average, in contrast to the analytical value. The analytical value of the translational specific thermal energy is available from the analytic expression of the energy evolution. Specifically, the average rotational and vibrational specific thermal energies evolve as

$$\varepsilon_{\mathrm{rot}}^{n+1} = \varepsilon_{\mathrm{rot}}^{n} e^{-\nu_{\mathrm{rot}}\Delta t} + \frac{D_{\mathrm{rot}}^2}{\nu_{\mathrm{rot}}}\big(1-e^{-\nu_{\mathrm{rot}}\Delta t}\big) \tag{88}$$

$$\text{and} \quad \varepsilon_{\mathrm{vib}}^{n+1} = \varepsilon_{\mathrm{vib}}^{n} e^{-\nu_{\mathrm{vib}}\Delta t} + \frac{D_{\mathrm{vib}}^2}{\nu_{\mathrm{vib}}}\big(1-e^{-\nu_{\mathrm{vib}}\Delta t}\big), \tag{89}$$

respectively. To conserve total energy, C_f follows as

$$C_f = \left(\frac{\bar{e}^{\,n} - \bar{\varepsilon}_{\mathrm{rot}}^{n+1} - \bar{\varepsilon}_{\mathrm{vib}}^{n+1}}{\bar{\widetilde{\varepsilon}}_{\mathrm{tr}}^{n+1}}\right)^{\frac{1}{2}}, \tag{90}$$

and the translational modes are updated via

$$M_i^{n+1} = C_f \widetilde{M}_i^{n+1}. \tag{91}$$

In order to honor also the correct correlation of position and velocity, an exact position integration scheme similar to the one given in Jenny et al. [JTH10] may be used. The derivation of this scheme is relegated to Part V, Section 29. In the present implementation, the positions are updated via the Euler scheme

$$\xi_i^{n+1} = \xi_i^n + \mu_i^n \Delta t, \tag{92}$$

which corresponds to the free-flight assumption made in DSMC.

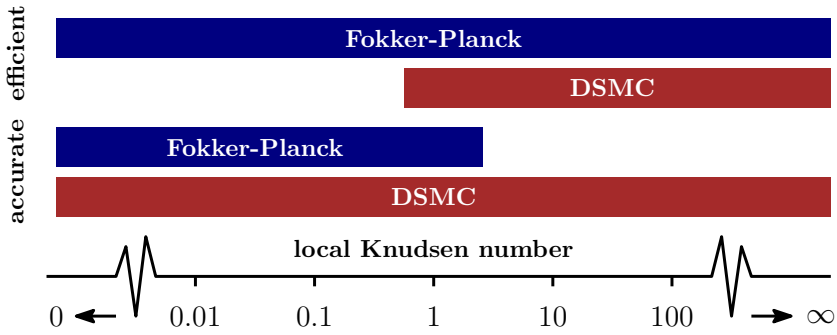


Figure 7: Ranges of accuracy and computational efficiency in terms of local Knudsen number for the Fokker-Planck and DSMC operators, assuming the relevant scales are resolved.

3.5 The Fokker-Planck-DSMC (FP-DSMC) Algorithm

As mentioned above, DSMC requires the resolution of the collision scales, that is, a grid spacing Δ smaller than the mean free path λ, as well as a time step size Δt smaller than the mean collision time τ. This means that the number of required grid cells for a resolved DSMC simulation scales as Kn^{-3}. While the computational complexity of both the NTC and Majorant Frequency schemes for DSMC is a linear function of the number of particles, the statistical nature of DSMC warrants a minimum number of computational particles per cell. This means that the computational cost to simulate a given volume of gas over a given time span also scales as Kn^{-4}. For near-continuum flows, where $Kn < 1$, DSMC quickly becomes computationally intractable.

Contrast this with the Fokker-Planck algorithm: here, the resolution requirements are far less stringent. The computational grid should resolve the relevant gradients, and the time step size may be chosen according to the Courant-Friedrichs-Lewy (CFL) condition, i.e., during one time step, particles should not travel further than the grid spacing. The computational complexity of the FP collision operator itself is linear in the number of particles N. However, a low to moderate Knudsen number is fundamental to the assumption that the FP operator adequately approximates the Boltzmann collision operator. In fact, the Fokker-Planck algorithm has been shown to be accurate for Knudsen numbers of up to ≈ 5 [JTH10; GTJ11; GJ14]. On the other hand, provided the resolution requirement can be met, DSMC is accurate irrespective of Kn, as long as the gas dynamics follow the Boltzmann equation. Assuming that the relevant scales are resolved, the ranges of accuracy and computational efficiency in terms of the local Knudsen number for the Fokker-Planck and DSMC operators are illustrated in Figure 7.

The complementary nature of the ranges of accuracy and efficiency of FP and DSMC, together with the fact that both methods are stochastic particle algorithms, differing only in the way that the particle velocity update occurs, immediately suggest a combined algorithm: Fokker-Planck-DSMC (FP-DSMC), introduced by

Gorji et. al. [GKJ13; GJ15]. At each time step of the simulation, in each cell of the computational mesh, a local criterion approximating the local Knudsen number is evaluated, and, if Kn is found to be less than one, the FP collision operator is selected, and DSMC otherwise.

From a practitioner's point of view, FP-DSMC may be thought of in two ways: either maximum accuracy given fixed computational resources, or minimal cost. For the former, one tries to achieve DSMC resolution in as large a region of the flow field as possible. The FP-DSMC procedure will then ensure that, in cells where the resolution requirements are not met, the Fokker-Planck operator is used instead, and the assumption is that in these cells, the requirements for accurate FP are satisfied. In case the mean free path is resolved throughout, DSMC is automatically used everywhere, ensuring the consistency of the FP-DSMC algorithm. Alternatively, one may wish to conduct a simulation with the least amount of computational resources possible. In this case, the spatio-temporal discretization is chosen according to the requirement of the FP operator. In cells where Kn is found to be too high for the FP operator, FP-DSMC switches to DSMC, again assuming that in this case, the relevant scales for DSMC are captured.

For the FP-DSMC algorithm, the switching criterion is the ratio of number N_{C} of computational particles in a given cell to the number $N_{\mathrm{C}}^{\mathrm{coll}}$ of collision among these particles during the given time step. Specifically, FP is chosen whenever $N_{\mathrm{C}}\big/N_{\mathrm{C}}^{\mathrm{coll}} < 1$. It is obvious that if more collisions than particles occur, the collision scales (mean free path, mean free time) are not resolved. In this case, the computational cost of DSMC scales with the number of collisions, while the cost of FP scales with N_{C} regardless. The FP-DSMC algorithm thus allows for computationally efficient simulations over the entire range of Knudsen numbers.

Following [GJ15], the FP-DSMC switching criterion is analyzed for the case of minimal resolution, that is grid spacing and time step size chosen according to gradient length scale and CFL criterion, respectively:

Defining the cell Knudsen number Kn_{C},

$$Kn_{\mathrm{C}} = \lambda/\Delta\,, \tag{93}$$

the spacial resolution requirement of DSMC, that Δ resolves the mean free path λ, may then be stated as $Kn_{\mathrm{C}} > 1$. The relevant Knudsen number for the FP operator is defined with respect to the local gradient length scale, i.e.,

$$Kn = \lambda/(Q/\|\nabla Q\|)\,, \tag{94}$$

where Q is a relevant macroscopic quantity. As mentioned above, FP simulations require $Kn \lessapprox 1$. Further, the grid spacing Δ should resolve the relevant macroscopic gradients. If Δ is assumed to be chosen according to

$$\Delta = Q/\|\nabla Q\|\,, \tag{95}$$

the cell Knudsen number Kn_{C} equals the local Knudsen number Kn, and one may conclude that the spatial resolution is sufficient for DSMC when $Kn_{\mathrm{C}} > 1$, and

the FP operator is accurate when $Kn_\mathrm{C} < 1$. It remains to be shown that the cell Knudsen number may be estimated proportional to the ratio $N_\mathrm{C}\big/N_\mathrm{C}^\mathrm{coll}$. Let the time step size Δt be chosen according to the following CFL criterion:

$$\Delta t = \frac{1}{2}\frac{\Delta}{\|\boldsymbol{U}\| + \sqrt{\theta}}, \tag{96}$$

which is sufficient temporal resolution for the FP operator. In a gas of hard spheres at equilibrium, the number of collisions among the N_C computational particles in a cell during the time Δt may be calculated as

$$N_\mathrm{C}^\mathrm{coll} = \sqrt{2/\pi}\, N_\mathrm{C}\Delta t\sqrt{\theta}\big/\lambda. \tag{97}$$

It follows that, together with Equation (96), the cell Knudsen number can be written as

$$Kn_\mathrm{C} = \frac{N_\mathrm{C}}{N_\mathrm{C}^\mathrm{coll}}\frac{1}{\sqrt{2\pi}\left(\sqrt{\gamma}Ma + 1\right)} \propto \frac{N_\mathrm{C}}{N_\mathrm{C}^\mathrm{coll}}. \tag{98}$$

In high Ma flows, care must be taken when using the $N_\mathrm{C}\big/N_\mathrm{C}^\mathrm{coll} = 1$ as the switching threshold, since the Δ may not resolve λ even though there are more particles than collisions per cell. Accordingly, in present implementation, a constant, runtime selectable scaling factor is used to adjust the switching point. Further, since the mesh nevertheless resolves the gradients, the phase density may be assumed to only vary negligibly across each grid cell, which should limit the error introduced by the too large mean collision spacing. Additionally, the use of collision sub-cells in DSMC to maintain an appropriate spacing of the collision partners could improve the accuracy in these situations. In any case, the FP algorithm already provides a significant margin of safety in terms of accuracy at moderate Knudsen numbers.

Since the number of computed collisions per cell is not available a priori, the ratio $N_\mathrm{C}\big/N_\mathrm{C}^\mathrm{coll}$ is estimated by a lower bound, that is, instead of the actual number of collisions, the number of selected collision partners in the NTC DSMC procedure is used (cf. Section 3.3). The estimate reads

$$\frac{N_\mathrm{C}}{N_\mathrm{C}^\mathrm{coll}} \approx \left[\frac{(N_\mathrm{C}-1)F_N}{2V_C}(\sigma_T\|\boldsymbol{g}\|)_\mathrm{max}\Delta t\right]^{-1}, \tag{99}$$

where $(\sigma_T\|\boldsymbol{g}\|)_\mathrm{max}$ is the maximum anticipated product of total collision cross section and relative velocity in the cell, F_N is the number of real molecules represented by one computational particle, and V_C the volume of the computational cell, respectively. This conservative estimate of the particle to collisions ratio also prevents switching to DSMC in cases where the mean free path is not fully resolved.

Part II
Parallel Implementation of the Fokker-Planck-DSMC Algorithm

This part is adapted from the article:
S. Küchlin and P. Jenny. "Parallel Fokker-Planck-DSMC algorithm for rarefied gas flow simulation in complex domains at all Knudsen numbers". In: *Journal of Computational Physics* 328 (Jan. 2017), pp. 258–277. DOI: 10.1016/j.jcp.2016.10.018.
Most of the text, figures and equations are identical to the corresponding sections of the publication.

4 Introduction

The FP-DSMC method—being a relatively recent development—had not been implemented in the framework of a general purpose 3D code before. In this part, an implementation is presented that is flexible enough to accommodate future developments and yet is efficient enough to tackle relevant problems. The code is capable of simulating 3D flows of diatomic molecules in and around complex geometry, which is demonstrated by its application to various relevant test cases. The simulation algorithm presented here—which does not require explicit specification of an interface between continuum and rarefied regions—is ideally suited for the study of complex flows. The present implementation is both flexible in terms of treatment of various flow types, from internal flows in micro devices to hypersonic reentry type flows, as well as efficient, i.e., it scales well with a large number of computational particles.

There exist many parallel implementations of the pure DSMC method. Notable examples include MONACO [DB96], SMILE [IMG98], DAC [LeB99], the code by Wu and Lian [WL03], MGDS [GZS10; GS10; GS11] and recently, SPARTA [Gal+14]. Most of these, however, are either purely based on distributed memory parallelism or on shared memory parallelism. Simulations on computer clusters with modern multi-core processors may benefit from a combination of the two. For example, the EULER cluster at ETH Zurich, on which the present implementation was tested, uses 12-core Intel Xeon processors. A key feature of the implementation is the use of both shared and distributed memory parallelism, which, on the processors mentioned above, allows for twelve times fewer sub-domains and therefore reduced communication overhead, while still being able to fully utilize the available computational resources.

The object oriented code is written in C++ and is completely modular in the sense that classical strategy patterns are employed for the implementation. The desired combination of algorithms is runtime configurable via a configuration file. Some of the run time options are:

- Molecular model, that is, Hard Sphere (HS), Variable Hard Sphere (VHS) and Variable Soft Sphere (VSS); both monatomic and diatomic.
- Collision operator, that is, Fokker-Planck, No Time Counter (NTC) DSMC, Majorant Frequency (MF) DSMC, Fokker-Planck-DSMC using either of the DSMC options.
- Geometry, that is, an arbitrary amount of primitives, i.e., any combination of stl files, nastran files, as well as explicitly defined primitives such as disks, spheres, cylinders, and parallelograms.

The rest of this part of the thesis is organized as follows: first, relevant features of the implementation are described, specifically data structures, parallelization and the treatment of complex geometries. In Section 8, results obtained with the new implementation for several test cases are presented: hypersonic flow past a corner profile, hypersonic flow around a sphere, as well as flow expansion through a 3D micro-nozzle geometry into vacuum. The results of the FP-DSMC algorithm are compared to those obtained by pure DSMC in terms of field quantities as well as computational cost.

5 Data Structures

5.1 Grid and Geometry

The implementation relies on an implicit Cartesian grid structure with immersed boundaries. In this part of the thesis, the domain is statically decomposed into Cartesian blocks for simulations leveraging distributed memory parallelism. Each block is simulated by one process, while inter-process communication is performed via the Message Passing Interface (MPI). The extension of the implementation to dynamic repartitioning of the domain is described in part III. Every process is aware of the entire geometry and of the global grid decomposition. This allows for the particle evolution step to be completed concurrently, and the communication due to particles that moved into another processor's domain is required only once. If a processor were only aware of the geometry present in its domain and only of connections to the adjacent domains, particles would potentially incur multiple communication events as they cross domain boundaries. For example, in the situation depicted in Figure 8, the particle trajectory (of one time step) labeled ① causes communication only between processes 4 and 3, while the trajectory labeled ② will not cause any communication at all.

There exist previous implementations [DB96; LeB99; WL03; GZS10] that rely on an outer loop over the move and communication routines to ensure that all particles reach their final destinations. The benefit of reduced communication has to be considered in view of additional memory required to store all geometry information on every process. For the cases studied so far, even the memory required

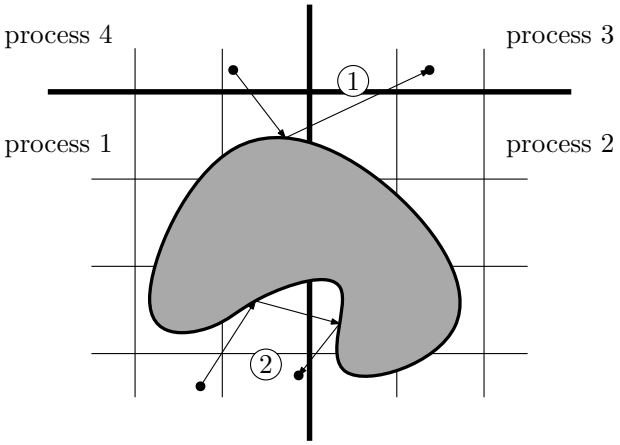


Figure 8: Particle-geometry interactions in a simulation on multiple processes.

for geometries with over 100×10^3 geometric primitives is negligible compared to the memory required for the particle data.

5.2 Particle Data

A defining characteristic of the present implementation is the handling of particle data. Instead of using a linked list of particle data structures in every cell to maintain cell-particle correspondence, the entire particle data are stored in a single matrix, in which each column holds the state vector of one particle. This is effectively an "array of structures" layout. Sorting this array by the cell index of each particle makes it easy to access all particles that belong to a given cell: Each cell only has to store the index of the column corresponding to the beginning of the particle data attributed to that cell, as well as the number of particles currently in the cell (see Figure 9).

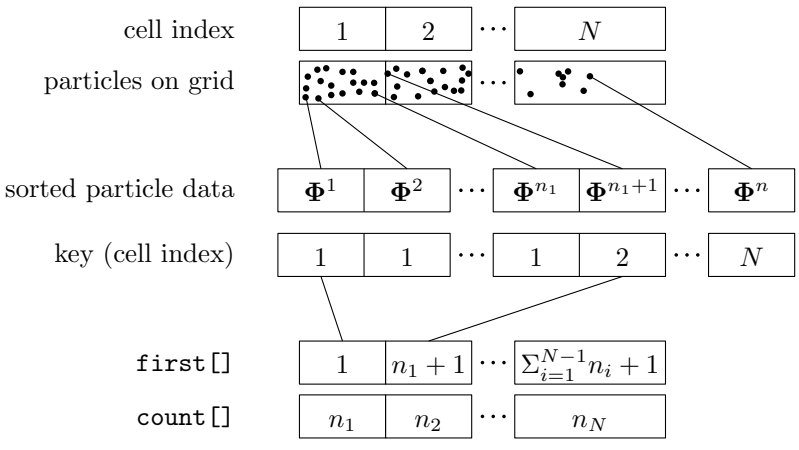


Figure 9: Particle data layout and indexing to cells.

The main motivation for contiguous storage of particle data is data locality.

Computational cost is often not dominated by floating point operations but by memory access and more importantly by the success rate of predicting memory access to avoid cache misses. In fact, many steps of the simulation algorithm are loops over the particle data belonging to each cell. Maintaining sorted particle data allows these loops to run as fast as possible. The importance of data locality becomes even greater for multi-threaded applications. Because the particle data are sorted, all data accessed by each individual thread are stored contiguously in memory, thus eliminating so called "false sharing".

A similar reasoning is discussed by Gao and Schwartzentruber [GS10; GS11]. In their approach, particle information belonging to one grid cell, but distributed in main memory, is first aggregated into a "cache-array" prior to processing, resulting in large performance gains.

For simulations on distributed memory architectures, the communication of particle data is a very costly step. It can only be performed efficiently, if the data to be sent are stored in a contiguous buffer. In the current implementation, the sorted particle matrix is the send buffer itself.

Obviously, sorting the particle data carries a non-negligible computational cost. For large numbers of particles, however, permutation of the particle data is more costly than the computation of the sorted order itself. In implementations without explicit sorting this data movement occurs as well, but prior to communication, when the send buffer is populated; and when looping over all cells. The sorting algorithm chosen for the current implementation is further detailed below.

6 Algorithms

6.1 Abstract Parallel Simulation Algorithm

As mentioned above, domain decomposition is used to parallelize the computation on distributed memory architectures. However, additional data parallelism exists as particles independently interact with boundaries and as sampling and collisions are performed independently for each grid cell. The implementation discussed here uses OpenMP[1] to execute these steps in parallel by a team of threads.

This leads to the following abstract parallel simulation algorithm:

[1] `http://openmp.org/wp/`

Algorithm 2: Abstract parallel simulation algorithm

```
1: on each MPI process, do
2:     decompose total domain → domain decomposition grid
3:     initialize local sub-domain
4:     for each timestep, do
5:         add particles at inflow
6:         for each particle p, do in parallel
7:             move p and apply boundary conditions
8:         on domain decomposition grid, do in parallel
9:             sort particles
10:        communicate particle data via MPI
11:        on local grid, do in parallel
12:            sort particles
13:        for each cell C on local grid, do in parallel
14:            sample statistics of particles in C
15:            apply collision operator to particles in C
16:        perform MPI parallel output
```

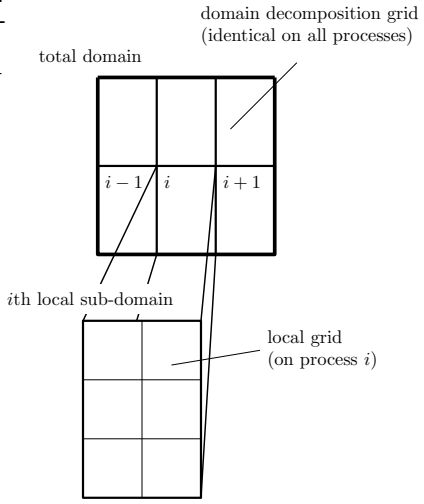


Figure 10: The grid hierarchy.

6.2 Parallel Sorting

The only non-trivially parallel portion of the algorithm is the sorting step. In the present implementation, parallel, out-of-place, most significant digit (msd) radix sort is employed (see e.g. Knuth [Knu98, pp.168]). This sorting algorithm has a run time complexity of $\mathcal{O}(m \cdot n/k \cdot p)$, where m is the number of digits in the largest occurring key, k is the number of digits in the radix, n is the number of particles to be sorted and p the number of threads participating in the sorting. Note that m is proportional to the logarithm of the number of grid cells.

Radix sort works by sorting data in m/k passes. Since copying the particle data is expensive and since this would be necessary m/k times per sorting operation, here the sorting step is split into two separate phases: First, an auxiliary array containing the cell key and array index of each particle is sorted. Second, the sorted array indices are used as a permutation vector to copy the particle data in sorted order to the final output array (see Figure 11).

In terms of memory access, moving data is more efficient when repeated memory access occurs to memory locations that are close together. This is the case when, during a time step, particles of a given cell all travel to cells that have similar keys. To improve the average proximity of cells in the key space, the cell keys are assigned in Morton order [ABM04].

6.3 Boundaries

During the movement operation, each particle trajectory has to be tested for possible boundary intersections. A common practice in DSMC codes is to index boundary geometry primitives to grid cells and to trace particles through each gird cell, while

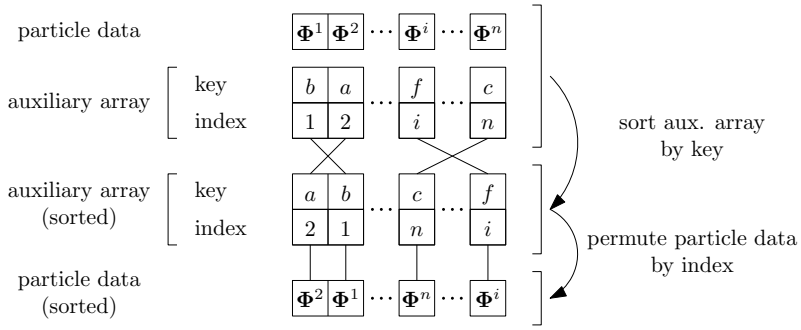


Figure 11: Indirect sorting of particle data.

testing the trajectory for intersections with those geometric primitives associated to the respective visited cell. An alternative is to use a cut-cell method, in which each cell intersected by a geometry surface is linearly deformed to approximate the boundary shape.

Here, a different approach is used, in which geometry and grid are kept completely separate. To avoid the necessity to test all trajectories against all geometry elements, the ray-tracing library Embree [Wal+14] is used to sort the geometry elements into a bounding volume hierarchy in a preprocessing step. Intersections of a trajectory with the bounding volumes can then be detected efficiently by Embree routines. The primitive tests themselves, e.g. ray-triangle intersections, are not performed by Embree, but rather by custom double-precision routines.

Since the boundary geometries are immersed in the regular Cartesian structure, cells which are intersected by the geometry have a smaller volume, which—if not accounted for—would misrepresent volume dependent averages and lead to wrong collision frequencies. Therefore, once the geometry bounding volume hierarchy (BVH) has been constructed (before the actual simulation is executed), each grid cell is tested for intersection with boundaries. If the cell is found to intersect, it is discretized into 1000 sub-cells, which are then counted towards the volume of the grid cell if their respective center lies inside the simulation domain. This volume sampling process is very efficient, since the BVH allows for fast intersection queries.

Moving the particles and checking their trajectories for boundary intersections is in itself embarrassingly parallel, since the particle trajectories are independent. However, particles advanced by different threads may collide with the same boundary, potentially causing race conditions when more than one thread tries to update the boundary sampling data at the same time. This is avoided by using appropriate atomic memory access directives provided by OpenMP.

6.4 MPI Communication

Since the particle data reside contiguously in memory and are sorted by their respective domain decomposition cell prior to communication, all particle data may be exchanged via a single all-to-all communication event. The memory occupied

by the particle data directly serves as input buffer and the starting location and element count of each message is directly available from the previous sorting step. Since the sorting algorithm is "out-of-place" (with separate input and output arrays), the input array to sorting can be reused as the output buffer for the communication. Furthermore, as mentioned previously, since all particles have been moved to their final positions for the present time steps prior to sorting, only one communication step is needed in the algorithm.

7 Parallel Performance

Strong and weak scaling behavior of the implementation are analyzed on ETH's high-performance computing cluster EULER[2]. Each of EULER's nodes is comprised of two 12-core Intel Xeon processors with shared memory.

The test case used is the same as described by Gao and Schwartzentruber[GS10] for the scaling analysis of their DSMC implementation called MGDS. That is, DSMC is used to simulate free-stream Argon flow at 50 K through a channel with dimensions 30 cm × 10 cm × 10 cm. The free stream velocity is 200 m s^{-1}. Our simulations employ between 100 and 1000 particles per cell on a 30 × 10 × 10 grid, with a time-step size of 2.5×10^{-5} s. The simulation wall clock time is measured after steady state is attained at 2000 time steps. Averages are recorded over the ensuing 8000 time steps. Note that although the results presented here are for simulations using pure DSMC, the choice of the collision operator does not affect the scaling and identical results can be expected for simulations using the FP or FP-DSMC algorithms.

Strong scaling efficiency is measured by comparing the runtime for a constant size problem divided among different numbers of cores to that of a serial run. The results for a constant problem size with 1.92×10^6 particles are shown in Figure 12. As the number of particles per core becomes low, the efficiency drops since a greater portion of the runtime is due to non-parallel tasks such as thread pool management and loop overhead.

Weak scaling efficiency is measured by comparing the runtimes per core for a problem size proportional to the number of cores. Here, weak scaling is analyzed in terms of the number of full nodes, each running one process utilizing all 24 cores. The load per core is kept constant by adding 10 cm to the length of the domain per 24 cores added (the size of one node). The results for three different core-loadings—10×10^3, 100×10^3 and 1000×10^3 particles per core—are shown in Figure 13.

The decline in efficiency is due to the increased relative contribution of the serial parts of the algorithm, including increased communication overhead. In part, the latter may be further exacerbated by the strict separation of geometry and grid: currently, sampling at inflow boundaries is distributed among all processes

[2] `http://www.top500.org/site/47773`

(with particles being communicated to the correct process during the global communication phase). In the specific problem simulated here, all particles sampled at the inflow boundaries must be communicated to the same process, thus causing an increase in per-node communication. This limitation will be removed by restricting inflow sampling to those parts of the geometry located in the domain for which the process is responsible. A further reason might be the inefficiency of the all-to-all communication operation which is applied to the particle data. The vast majority of communication occurs between processes responsible for adjacent sub domains. This means that the source and destination vectors for the all-to-all communication operation are sparse. This case might not be fully optimized by the MPI library. Switching to the more recent neighbor all-to-all communication pattern is expected to help alleviate the problem [HT09], and is discussed in part III of this thesis.

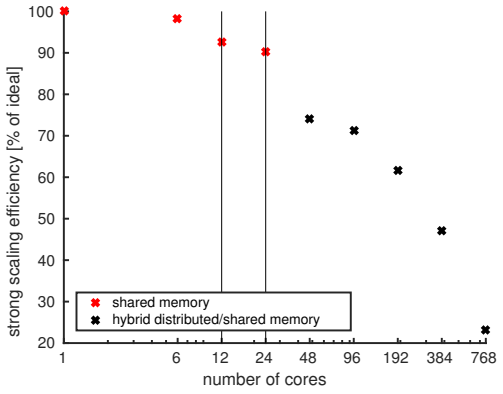


Figure 12: Strong scaling efficiency for DSMC simulation of free stream Argon flow through a channel using 1.92×10^6 particles. The vertical lines indicate the number of cores per processor (12) and per node (24), respectively. At 768 cores, each core only processes about 2500 particles, which causes the run time to be completely dominated by overhead.

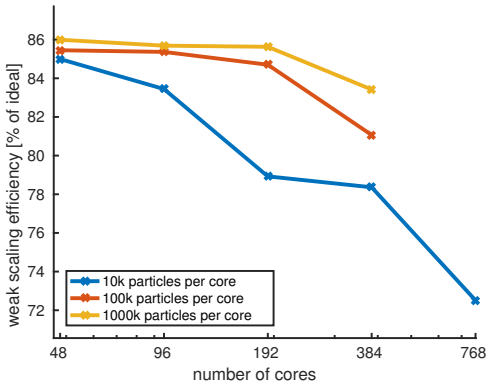


Figure 13: Weak scaling efficiency for DSMC simulation of free stream Argon flow through a channel. For each node used in the computation, the domain is extended in the x-direction by 10 cm. Each node uses all 24 available cores. The 100% reference is a simulation using a single node (24 cores) with shared memory.

8 Validation Studies

8.1 Supersonic Corner Flow

To compare DSMC and FP-DSMC results, hypersonic Argon flow over a corner profile was simulated. The geometry of the test case is similar to that used by Bird [Bir94, Chapter 16.2] and is shown in Figure 14.

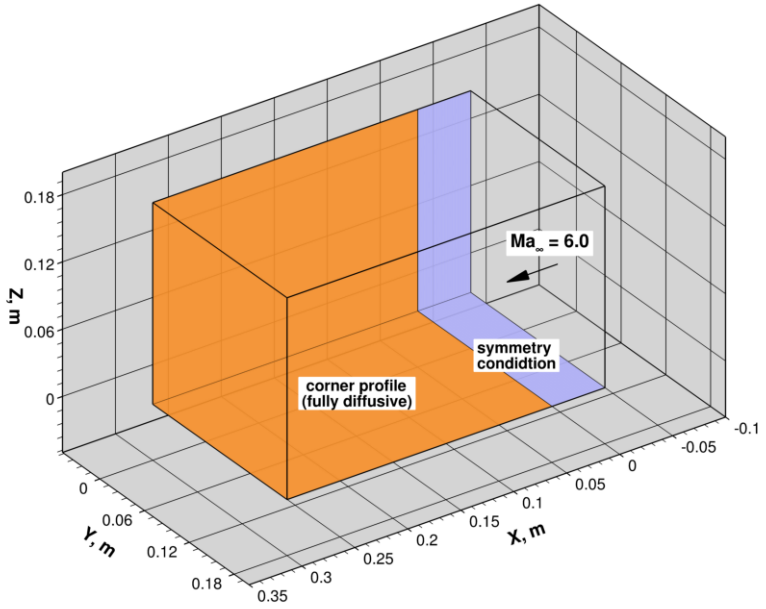


Figure 14: Schematic of the simulation volume used in the 3D corner flow simulations (Section 8.1).

Since the flow conditions simulated by Bird lead to a Knudsen number too high for simulations using the FP method, the number density was chosen ten times higher. The time step size was selected according to a CFL criterion based on the mean inflow speed and the cell width. First, pure DSMC was used to simulate the flow with 4 different grid resolutions. These simulations were then repeated—with the same parameters—using the FP-DSMC algorithm. The simulation parameters are listed in Table 3 in B.1.

Figure 15 shows a qualitative comparison of the results obtained by pure DSMC and those obtained from an FP-DSMC simulation, both with the finest grid. Pressure contours on the profile surface demonstrate that the results are in very good agreement.

A quantitative comparison of the results is provided by the plot in Figure 16, which shows a measure e of the solution errors. The latter is defined by the normalized average of the magnitude of the local difference in temperature T with respect to the pure DSMC result with the finest grid:

$$e^h_{\mathrm{co}} := \frac{1}{T_{\mathrm{ref}}} \overline{\left\| T^h_{co}(\boldsymbol{x}) - T^{h^{\min}}_{\mathrm{DSMC}}(\boldsymbol{x}) \right\|}, \quad \mathrm{co} \in \{\mathrm{DSMC}, \mathrm{FP\text{-}DSMC}\}, \tag{100}$$

where the subscript co indicates the collision operator used in the simulation, h is the cell-spacing of the grid, $T_{\mathrm{ref}} = 300\,\mathrm{K}$ is the reference temperature and the operator $\overline{(\cdot)}$ denotes averaging over all points $\boldsymbol{x}$.

The main two observations are that for a given grid the FP-DSMC simulations are both faster and more accurate than the corresponding pure DSMC simulations. Moreover, these trends become more prominent for coarser grids. For example,

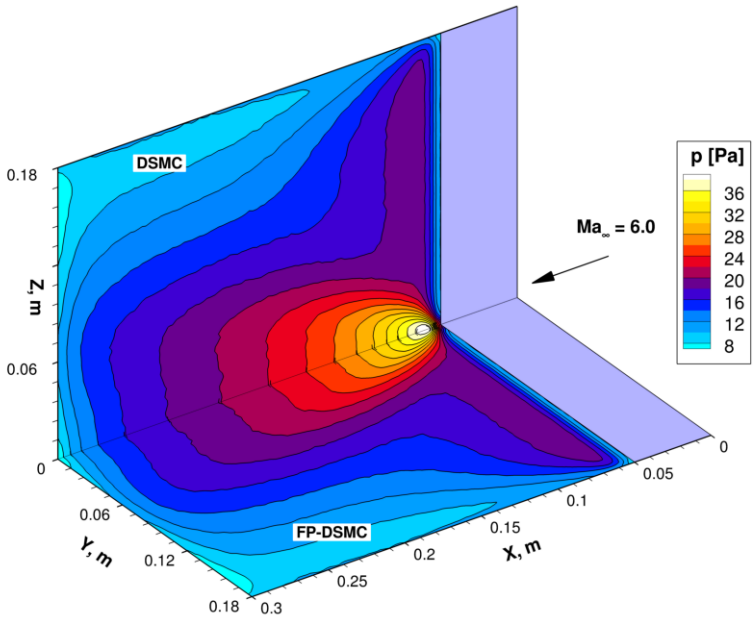


Figure 15: Qualitative comparison of the results from DSMC and FP-DSMC simulations of hypersonic Argon flow over a corner profile. The surfaces of the corner profile in the (X, Z) and (X, Y) plane are colored by contours of surface pressure obtained from pure DSMC and FP-DSMC simulations, respectively.

on the coarsest grid ($h = 0.01\,\mathrm{m}$), FP-DSMC runs over 4.5 times faster than pure DSMC, while at the same time reducing the error measure by more than 25 %.

The higher efficiency of FP-DSMC, especially for coarser grids, can be explained by the fact that in cells significantly larger than the mean free path length, each particle experiences numerous collisions per time step, which renders classical DSMC very costly. In the FP-DSMC algorithm, these cells are updated via the FP collision operator, the computational cost of which scales linearly with the number of particles, regardless of the collision frequency. If the grid resolution increases, the FP collision operator is invoked less frequently and the computational cost eventually reaches that of pure DSMC.

The observation that coarse FP-DSMC simulations are more accurate than coarse DSMC simulations stems from the reduced resolution requirements of the FP operator. More precisely, for an accurate FP simulation—provided the Knudsen number is not too large—the grid only has to be fine enough to capture the variation of the macroscopic averages. Unlike in DSMC, the collisional scales do not have to be resolved [GTJ11].

8.2 Supersonic Flow Over a Sphere

Further assessment of the FP-DSMC method and its present implementation was done via simulations of hypersonic flow of Argon over a sphere. Here, DSMC and FP-DSMC simulations were performed with three different grid resolutions. Again,

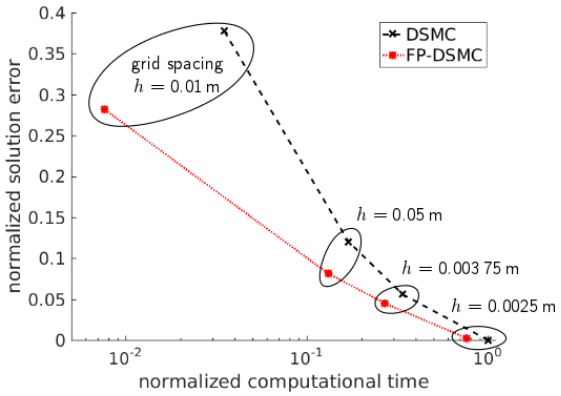


Figure 16: Results for simulations of hypersonic Argon flow over a corner profile with different grid resolutions: The normalized solution error—as defined by Equation (100)—is plotted versus normalized computational time.

except for the collision operator, the corresponding simulations were identical; the simulation parameters are listed in Table 4 in B.1.

Figure 17 offers a qualitative comparison of the results obtained by DSMC and FP-DSMC: The Mach number contours on a slice plane through the center of the flow field, as well as the pressure distribution on the sphere surface, show good agreement between the two methods.

Figure 18 shows the result of a typical FP-DSMC computation in terms of how often the particles in a given cell were updated via the FP operator. The FP operator is used in regions of high density, i.e. in the conical region between compression shock and ensuing rarefaction. In this region, high density and high temperature lead to a high collision rates. Wherever the collision rate exceeds the number of particles, the FP operator is used, which leads to a reduction of computational time.

The advantages of using the FP-DSMC scheme are further shown in Figure 19. As in Section 8.1, the normalized average of the magnitude of the local difference in temperature with respect to the pure DSMC result with the finest grid—as defined by Equation (100)—is shown for different grid resolutions, plotted against the normalized computational time. Again, for coarser grids, FP-DSMC results are more accurate than pure DSMC results and the computational savings due to the FP collision operator become more significant.

In terms of sphere drag coefficient, experimental values of about $1.1 - 1.2$ are reported in the literature for the simulated conditions [BH71]. Figure 20 shows the values obtained from the current DSMC and FP-DSMC simulations. The simulations approach the experimental value with increasing grid resolution, with the finest simulation (at a grid spacing of 0.0025 m) yielding $C_D = 1.199$. Under-resolved simulations over-predict the drag coefficient. Since the resolution requirements are less stringent for the FP than for the DSMC collision operator, under-resolved (in terms of mean free path) FP-DSMC simulations tend to be more

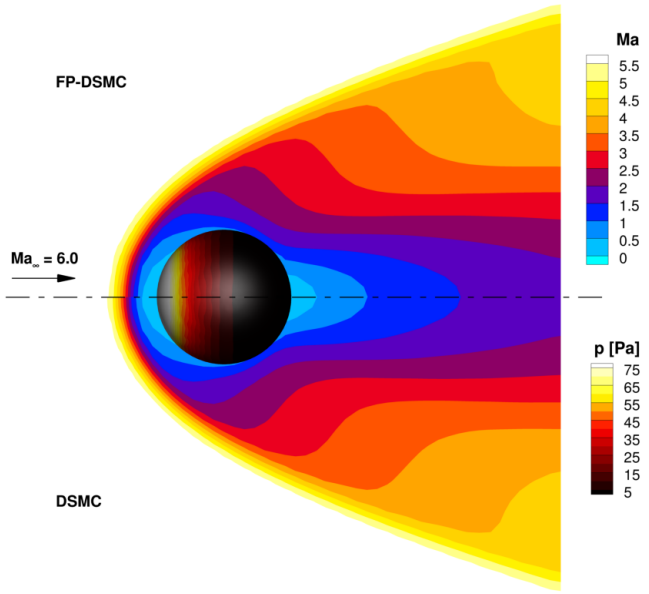


Figure 17: Qualitative flow field comparison of the results from DSMC and FP-DSMC simulations of hypersonic Argon flow over a sphere. The slice plane is colored by Mach number contours and the sphere surface by pressure contours. The small difference in the bow shock location is below the grid resolution.

accurate than corresponding pure DSMC simulations, which is confirmed by the results presented in Figures 19 and 20.

8.3 3D Micro-Nozzle Flow

Expansion of gas through a nozzle from continuum into vacuum is a challenging test case for DSMC simulations, since the entire range of Knudsen numbers—from Kn close to zero at the nozzle inlet, to Kn approaching infinity at the vacuum outlet boundaries—is present in the flow field. The Fokker-Planck-DSMC algorithm is ideally suited to deal with this kind of flow. 2D simulations of nozzle flow using FP-DSMC have been presented by Gorji and Jenny [GJ15], where large performance gains with respect to pure DSMC were demonstrated.

In the following, 3D simulation results for micro-nozzle flow are discussed, which were obtained using the new implementation. The geometry and flow conditions were chosen according to the work of Alexeenko et al. [Ale+02; Ale03]. Cold Nitrogen expands through a flat micro-nozzle with a 15° opening angle and area ratio of 10 : 1. The simulation domain contained the region from the nozzle throat to about 2.5 mm downstream of the nozzle exit. The inlet conditions were specified at the nozzle throat as Ma = 1 equilibrium flow with temperature and pressure calculated from ideal nozzle theory. The nozzle geometry used in the simulations is shown in Figure 21, and the simulation parameters are listed in Table 5 in B.1.

The resulting x-velocity fields on the nozzle center plane obtained from simu-

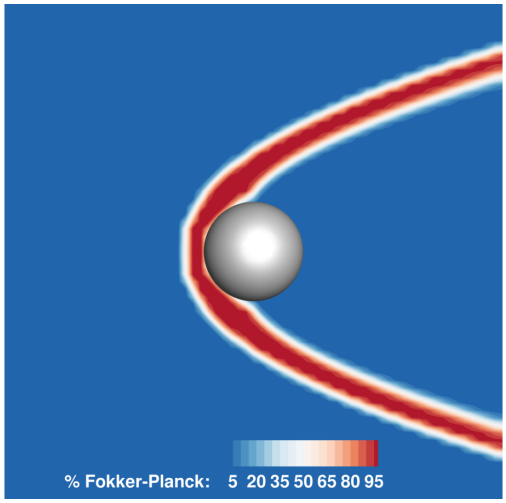


Figure 18: The slice plane is colored by the percentage of Fokker-Planck collision operator choices over the course of the simulation of hypersonic Argon flow over a sphere using the FP-DSMC scheme. The FP operator is chosen in regions where many collisions occur.

lations with different grid resolutions are shown in Figure 22. It is immediately apparent that the resolution has a great impact on the results. For comparison, Figure 23 shows the velocity contours reported by Alexeenko et al. [Ale+02, Figure 12]. Unfortunately, the exact grid resolution used in that simulation is not available. However, the velocity contours obtained from the simulation with a $900 \times 240 \times 18$ grid (Figure 22d) are in very good agreement.

Figure 24 compares the translational temperature obtained with a $900 \times 240 \times 18$ grid to the results published by Alexeenko et al. [Ale+02, Figure 13]. Again, good agreement can be observed.

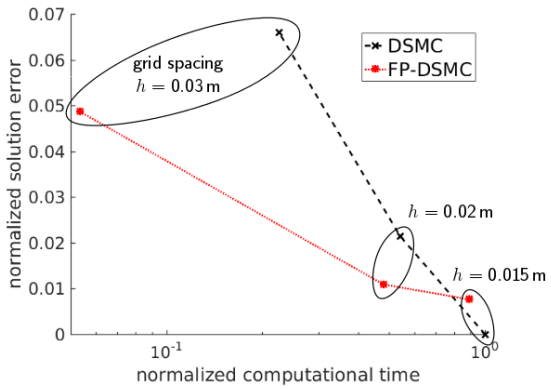


Figure 19: Results for simulations of hypersonic Argon flow over a sphere with different grid resolutions: The normalized solution error—as defined by Equation (100)—is plotted versus normalized computational time.

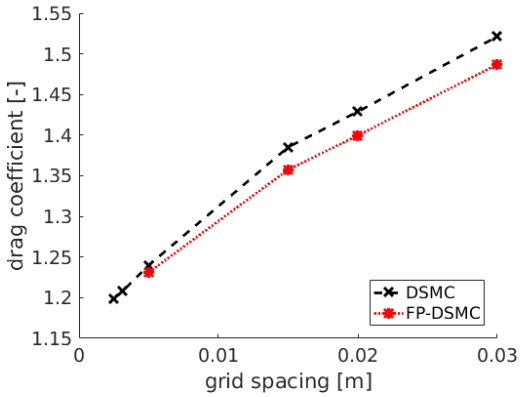


Figure 20: Sphere drag coefficient obtained from FP-DSMC and pure DSMC simulations with different grid resolutions.

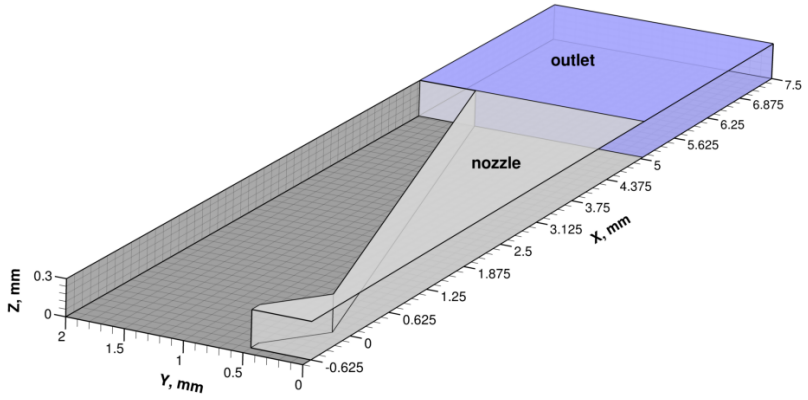
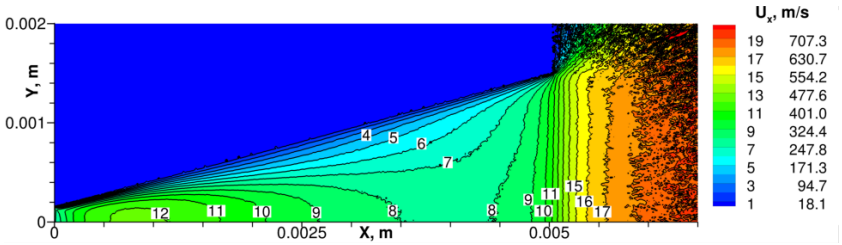
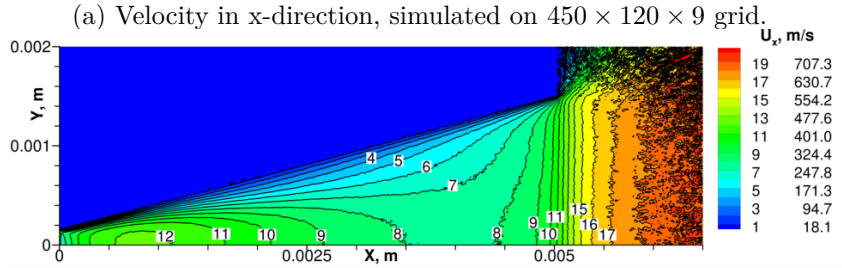


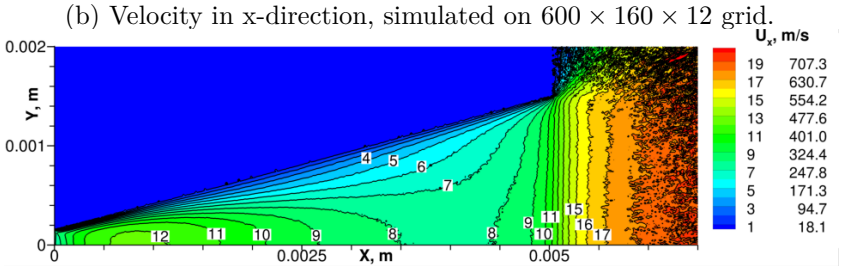
Figure 21: Geometry used in the 3D simulations of micro-nozzle flow (Section 8.3). The simulations presented here only consider the domain downstream of the nozzle throat, i.e. in the region where $x \geq 0$.



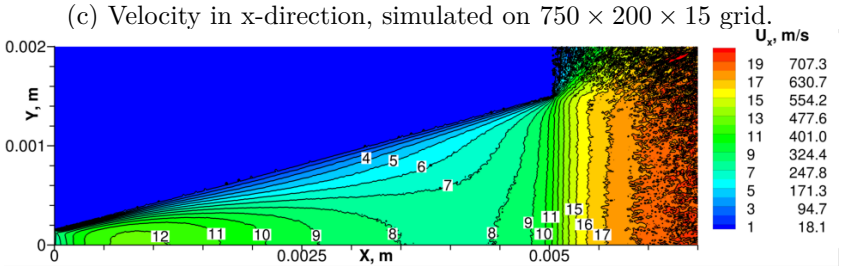
(a) Velocity in x-direction, simulated on $450 \times 120 \times 9$ grid.



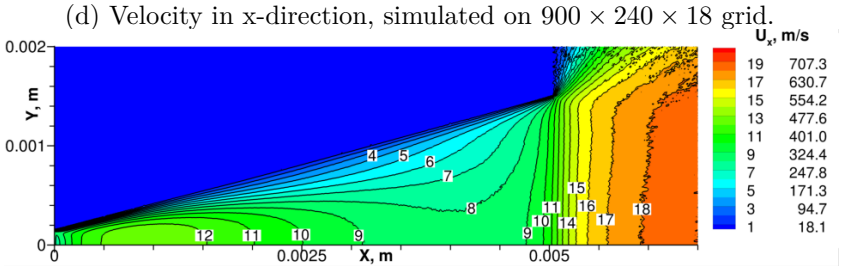
(b) Velocity in x-direction, simulated on $600 \times 160 \times 12$ grid.



(c) Velocity in x-direction, simulated on $750 \times 200 \times 15$ grid.



(d) Velocity in x-direction, simulated on $900 \times 240 \times 18$ grid.



(e) Velocity in x-direction, simulated on $1050 \times 280 \times 21$ grid.

Figure 22: Comparison of x-velocity contours on the nozzle center plane simulated with five different grids.

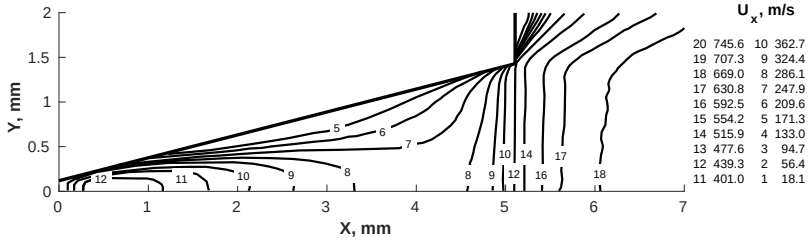
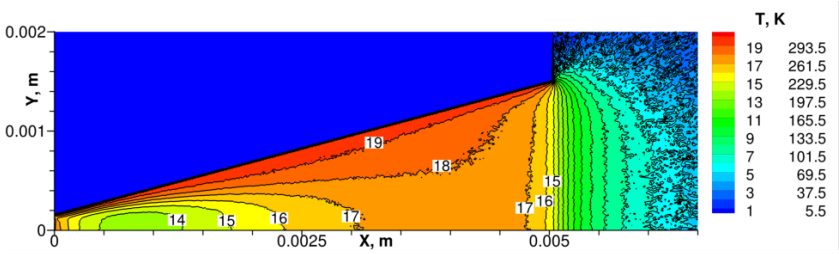
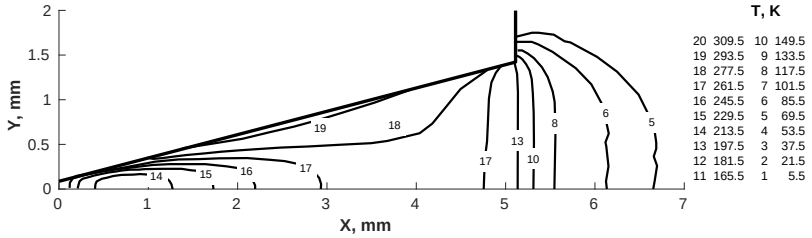


Figure 23: Data reported by Alexeenko et al. [Ale+02, Figure 12]: contours of x-velocity from 3D micro-nozzle simulation.



(a) Translational temperature on nozzle center plane, simulated on 900 × 240 × 18 grid.



(b) Data reported by Alexeenko et al. [Ale+02, Figure 13]. Translational temperature on nozzle center plane.

Figure 24: Comparison of translational temperature contours on the nozzle center plane obtained from a FP-DSMC simulation with results reported by Alexeenko et al. [Ale+02].

9 Conclusions

A general FP-DSMC algorithm for parallel computations of rarefied gas-flow in and around 3D geometries is devised and its implementation is shown to be efficient and accurate. The FP-DSMC solution algorithm, which is attractive since collision time-scales do not have to be resolved (opposed to pure DSMC), has already been presented previously for serial, 2D simulations with simple geometries. This is the first time that an efficient, parallel algorithm and implementation of FP-DSMC for 3D, complex geometries have been developed and tested.

The implementation of the new algorithm makes use of coarse- and fine-grained parallelism inherent to particle simulations by employing a hybrid OpenMP-MPI programming paradigm. While message passing is necessary when particles cross computational sub-domains, shared memory architecture is exploited to deal with the evolution of particles within the same domain. In this way, the total communication overhead can be reduced compared to a pure domain-decomposition approach. A crucial factor for a good overall efficiency is the special data structure, which ensures that the particle data of each thread are stored contiguously. This renders the implementation cache friendly and simplifies the exchange of particle data via message-passing.

To deal with complex, 3D geometries, Intel's ray-tracing library Embree is employed. Immersed surfaces are treated independently of the underlying Cartesian sampling grid, and to avoid an iterative communication procedure, the entire geometry is stored on each processor. During each time step, it has to be checked for all particle trajectories with which geometric primitives collisions occur. To avoid checking all primitives, Embree uses an efficient algorithm based on a bounding volume hierarchy.

Efficiency and accuracy of the new code could be demonstrated for a series of three-dimensional test cases, that is, simple channel flow, hypersonic flow over a corner profile, hypersonic flow over a sphere and flow through a micro-nozzle into vacuum. First, comparisons with pure DSMC (including results in the literature) confirm that the FP-DSMC method leads to at least equally accurate results at lower computational cost. Second, both strong and weak scalability of the parallel algorithm have been investigated. As a rough guideline, it can be stated at this point that at least 20 000 particles are necessary per core to achieve good parallel efficiency.

The code is implemented in C++, and due to its modular and object-oriented design, it is very flexible and easy to extend.

Part III
Automatic Mesh Refinement and Domain Decomposition via Space-Filling Curves

This part is adapted from the article:
S. Küchlin and P. Jenny. "Automatic mesh refinement and parallel load balancing for Fokker-Planck-DSMC algorithm". In: *Journal of Computational Physics* 363 (June 2018), pp. 140–157. DOI: 10.1016/j.jcp.2018.02.049.
Most of the text, figures and equations are identical to the corresponding sections of the publication.

10 Introduction

In this part of the thesis, several innovations related to the implementation are discussed with the aim to further leverage the potential of the FP-DSMC method, which has proven to be very effective in simulating flows covering the entire Kn range. So far, the presented implementation was restricted to equidistant spatial grids. To account for the multiple length scales typically involved in flows covering a wide *Kn* range, the implementation is extended to use an automatically refined, implicit octree mesh. In simulations employing locally adapted grids in particular, but in general in any simulation of flow with large spatial variation of the density of computational particles, balancing the computational load over the set of processors involved in the parallel simulation becomes very important to fully utilize the available computational resources both in terms of memory and CPU time.

It turns out that using a space-filling curve as the underlying structure for the spatial grid is an elegant solution to efficiently implement both local grid refinement and parallel load balancing. Finally, the larger the number of processors involved in the parallel execution of the simulation, the more important efficient data communication becomes. A recently developed communication pattern addresses the specific demands incurred by the code architecture.

10.1 Related Works

There exist many examples in the literature of space-filling curves (SFC) being used to facilitate scientific computing applications; see for example the book by Bader [Bad13] and references therein. Only a few can be mentioned explicitly here: For example, Aftosmis et al. [ABM04] give a detailed description of the application of SFCs to CFD, including mesh refinement and load balancing. Lin-

termann et al. [Lin+14] and Schneiders et al. [Sch+15] use a hierarchical Cartesian mesh ordered by a Hilbert curve for parallel grid generation, parallel load balancing and automatic mesh refinement in the context of CFD, and turbulent, particle-laden flows in particular. In the technical note "CUDA Particles" [Gre08], S. Green describes simulating an n-body system including long range interactions and collisions on graphic cards, in which the Morton order is used for spatial partitioning. The radix sort algorithm is used to index particles on the mesh after movement. Harlacher and co-workers [Har+12] developed a dynamic load balancing scheme for unstructured meshes based on space-filling curves, and use a heuristic to compute a new partitioning of the mesh. Jambunathan and Levin [JL15; JL16b; JL16a] have developed the DSMC code "CHAOS", which uses a forest of octrees linearized by the Morton order. In their approach, the particles are recursively grouped into a forest of octrees at each time step, and the leaf nodes are then distributed to multiple GPUs to perform load-balanced computations. Recently, Pfeiffer and Gorji [PG16] described the implementation of the FP-DSMC algorithm within the particle-in-cell framework "PICLas". The implementation uses the Peano SFC to dynamically aggregate sub cells based on a minimum required number of particles.

11 Indexing Cartesian Grids using Discrete Approximations of Space-Filling Curves

The following gives an overview of the properties of space-filling curves, and describes the implementation of SFC-based cell index computation on Cartesian grids.

11.1 Definition and Properties of Space-Filling Curves

Formally, a curve is defined as the image $f_*(\mathcal{I})$ of a mapping $f : \mathcal{I} \to \mathbb{R}^d$ from the compact set $\mathcal{I} \subset \mathbb{R}$ into $\mathbb{R}^d$. If $f : \mathcal{I} \to \mathcal{Q} \subset \mathbb{R}^d$ is surjective and $\mathcal{Q}$ has non-zero volume, then $f_*(\mathcal{I})$ is a space-filling curve. As is common in the aforementioned literature, the term *space-filling curve* will be used to refer to their discrete approximations. The purpose of a discrete SFC is then to assign a unique integer index to any two- or three-dimensional discrete coordinate, and vice versa. In particular, one is interested in bijective mappings f_N from the compact set $\mathcal{I}_N \in \mathbb{N}_0$ into $\mathcal{Q}_N \subset \mathbb{N}_0^d$. In the following, $\mathcal{I}_N$ will be referred to as the index space, $\mathcal{Q}$ as the data space, and $\mathcal{Q}_N$ as the discretized data space, respectively. For example, in the context of Cartesian meshes, numbering the grid cells by the SFC index of the respective cell vertex with the lowest coordinates induces an ordering of the grid cells in terms of their respective index $I \in \mathcal{I}_N$. Traversing the mesh in order of increasing SFC index of the cells corresponds to following the SFC through the mesh.

Space-filling curves that define useful sequentializations for simulation purposes are characterized by the following properties:

Locality: If the distance of two points in data space is small, their distance in the index space should be small as well. Compare this to the naive ordering on a regular 3D Cartesian given by $I = i + n_i j + n_i n_j k$, where n_i and n_j are the number of cells in the i and j direction, respectively. Here, neighboring cells do not have consecutive indices except in the i direction, and the differences—especially in the k direction—can become very large. The SFC-induced ordering should not favor any particular direction.

2^d-tree sequentialization: Points contained within the same orthant (i.e., quadrant, octant, etc.) of data space should map to a compact subset of index space. This property means that an SFC defines a unique ordering of the cells in an arbitrary quad/octree. Different types of SFC may partition the unit cube differently. For example, the Peano curve splits the unit cube along each dimension into three equidistant parts to form 27 sub-cubes in 3D.

Local construction: The map from data to index space should only depend on local properties, i.e. the data space coordinates as well as fixed parameters. Compare this to the sequentialization of a multilevel Cartesian mesh, where a cell index would be computed by recursive insertion starting at the top level and requiring knowledge of the specific decomposition at each level.

The following two sections provide a brief description of the two different types of SFCs that have the above properties and are used within the implementation presented here. The relevant computational task in each case is the evaluation of the inverse map $f_N^{-1} : \mathcal{Q}_N \to \mathcal{I}_N$. To this end, any point $\boldsymbol{P} \in \mathcal{Q} \subset \mathbb{R}^d$ in real data space, for which $I \in \mathcal{I}_N$ is to be evaluated, is first mapped to $\boldsymbol{P}_N \in \mathcal{Q}_N \subset \mathbb{N}_0^d$ via the component-wise relation $P_{i,N} = \left\lfloor n_{(i)} \frac{P_{(i)} - O_{(i)}}{l_{(i)}} \right\rfloor$ (with summation over indices in brackets suppressed), where O_i is the origin, l_i the maximum extent, and n_i the number of representable discrete values in coordinate direction $i \in [0, \ldots, d-1]$, respectively.

11.2 Morton Order

The Morton order (also called N- or Z-order, due to the shape of the curve in 2D) arises naturally from a depth-first traversal of a quad-/octree. It can be computed very efficiently from the d-dimensional vector of integers $\boldsymbol{P}_N$ via bit interleaving: Let the ith discrete coordinate of $\boldsymbol{P}_N$ be represented by B binary digits $b_{i,k}$ as $P_{i,N} = \sum_{k=0}^{B-1} b_{i,k} \times 2^k$. The corresponding Morton index I_M is then given by

$$I_M = \sum_{j=0}^{\lfloor B/d \rfloor - 1} \sum_{i=0}^{d-1} b_{i,j} \times 2^{i+dj}. \tag{101}$$

A C++ code for this interleaving is included in Appendix C.1. Note that any SFC index represented by B bits, regardless of which SFC is used, can only incorporate $\lfloor B/d \rfloor$ bits from each of the d coordinates of $\boldsymbol{P}_N$.

Drawing a curve through the centers of cells on a regular 2D grid in Morton order produces a self-similar hierarchy of Z shapes with $\lfloor B/d \rfloor$ levels. On every coarsening level j, the Z in each cell defines the order in which the respective cell's four quadrants on level $j-1$ are visited. Figure 25 shows a 2D example of the first 3 refinement levels of the Morton order on an 8×8 grid.

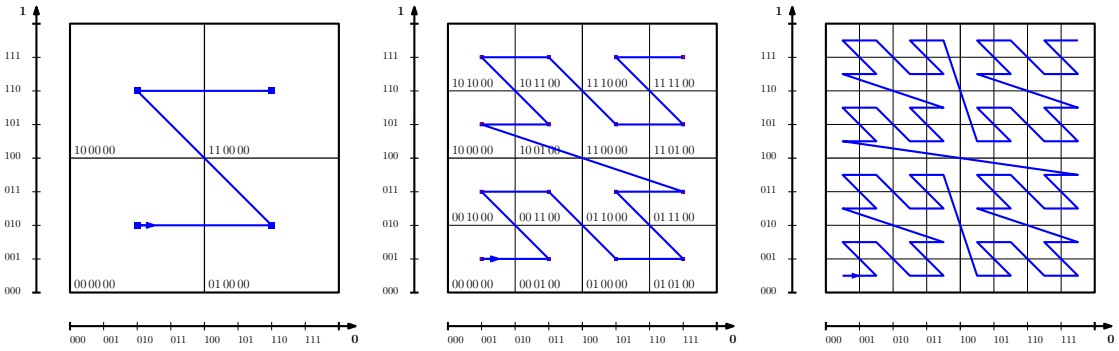


Figure 25: 2D Morton order at three levels of refinement. The values in the quadrant corners denote their respective Morton key. In this example, the maximum number of representable discrete values in each coordinate direction is 8. At the finest representable refinement level of the Morton order, shown in the right-most panel, all points within a given cell will be mapped to the same 6-digit key.

11.3 Hilbert Order

The Hilbert order has, in general, better locality than the Morton order. Furthermore, there are Hilbert orders that are face-continuous, i.e. neighboring cells in index space will always share a face in 2D/3D. This means that for a 1D problem decomposition, as discussed below in Section 13.3, the corresponding 2D/3D domain will be compact. However, the Hilbert index is more complicated to construct than the Morton index. Superiority of either ordering thus is problem dependent. As shown by Haverkort [Hav17], there are more than 10^7 unique (self-similar) Hilbert curves in three dimensions. In 2D, there exists only one Hilbert order, and drawing a curve through the centers of cells on a regular 2D grid in Hilbert order produces a self-similar hierarchy of U shapes, which are rotated and reflected appropriately (see e.g. Figure 26).

Self-similar Hilbert curves are defined by the order in which the orthants of a cube are visited (called the base order), along with the necessary transformation that specifies the traversal of the sub-orthants within each base orthant. For example, in 3D, the unit cube has 48 symmetries. Allowing also for a reverse traversal order, there are thus, in general, a total of 96 different orders in which

the octants may be visited. A given orthant order will be referred to as the "state" of the traversal in the following. In contrast to the Hilbert order, the traversal in Morton order only has one state.

To map a vector of d coordinates to their index on the chosen Hilbert curve, the corresponding Morton index is computed first. The mapping from Morton index I_M to Hilbert index I_H then proceeds via an iterative look-up table approach, similar to the procedure described by Campbell et al. [Cam+03]:

Let s_j denote the current state of the traversal, initialized at $j = \lfloor B/d \rfloor - 1$ to the state corresponding to the base order. Let $o_j \in [0, \ldots, 2^d - 1]$ denote the orthant of the unit cube at the current coarsening level j. Note that in d dimensions, o will be a d-bit number. Two tabulated functions are used that fully define each Hilbert curve: the state transition function $s_{j-1} = S(s_j, o_j)$ that indicates the transformation of the traversal when moving to the next lower coarsening level, as well as the index function $h_j = H(s_j, o_j)$, $h \in [0, \ldots, 2^d - 1]$ which defines the ordering of the orthants at the current level. Again, h is a d-bit number. To generate I_H, the $\lfloor B/d \rfloor$ d-tuples of bits in I_M are examined sequentially in order from most to least significant. The j-th d-tuple of bits represents the orthant number o_j at coarsening level j in which the encoded point is located. For each j, the corresponding d bits in I_H are set to h_j, and s is updated via the transition function S. Intuitively, each Morton "Z" is mapped to the appropriate Hilbert "U". Denoting again the individual bits of I_M as $b_{i,j}$, one may write more concisely:

$$I_H = \sum_{j=0}^{\lfloor B/d \rfloor - 1} H(s_j, o_j) \times 2^{dj}, \tag{102}$$

$$\text{where} \quad o_j = \sum_{i=0}^{d-1} b_{i,j} \times 2^i,$$

$$s_{j-1} = S(s_j, o_j),$$

$$\text{and} \quad s_{\lfloor B/d \rfloor - 1} = s_{\text{base}}.$$

For example, assuming 64 bit integers are used for both the Hilbert and Morton index—of which $21 \times 3 = 63$ bits are relevant—generating the former from the latter thus requires 42 table look-ups. A C++ code to generate the 3D Hilbert index from the Morton index is included in C.1. Figure 26 shows a 2D example of the first 3 iterations of the Hilbert curve on an 8×8 grid, and Figure 27 illustrates the construction of functions H and S for the 2D case.

For individual Hilbert curves, the sizes of the look-up tables may be reduced significantly by identifying states that can never be reached from the base state and eliminating the corresponding rows. For example, the curve nicknamed "Butz" (illustrated in Figure 28) only uses 12 states, and the two required tables thus have a combined storage cost of only 192 bytes. Since they can be kept in low-level cache, the generation of the corresponding Hilbert index is very fast. The curve nicknamed "Alfa", on the other hand, uses 48 unique states.

By identifying cycles in the transformation, the transformation Morton to

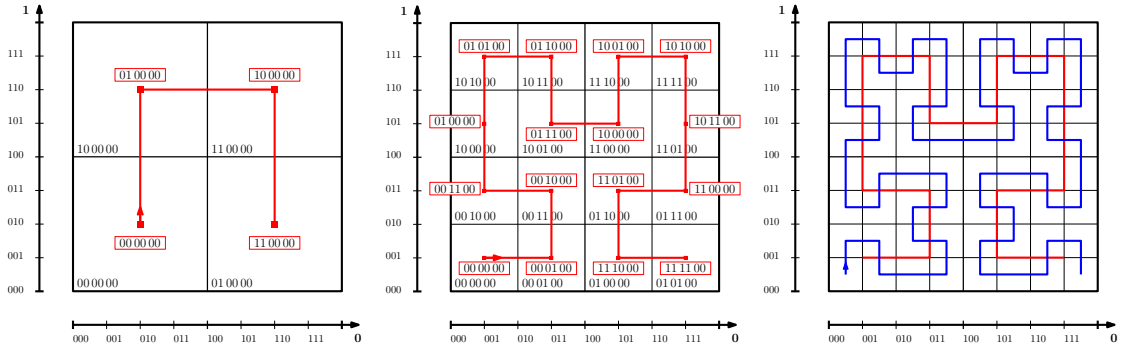


Figure 26: 2D Hilbert curve at three levels of refinement. The values in the quadrant corners denote their respective Morton key, the boxed values are the corresponding Hilbert indices. In this example, the maximum number of representable discrete values in each coordinate direction is 8. At the finest representable refinement level of the Hilbert curve, shown in blue in the right-most panel, all points within a given cell will be mapped to the same 6-digit key.

Hilbert can be further accelerated for small Morton indices with correspondingly many leading zero bits. Taking again the Butz curve as an example, it can be seen that the base state reoccurs after three consecutive visits of the zeroth octant. Thus, if the Morton index under consideration has 9 leading zero bits, the next state will be again the base state and the first 9 bits of the corresponding Hilbert index will be zero. The number of leading zero bits can be obtained at negligible computational cost directly by special CPU instructions on many processors.

Note that the reverse transform—Hilbert to Morton index—proceeds completely analogously, using the same state transition table and the orthant-to-key table which specifies the inverse permutation. The choice of which Hilbert curve to use depends on the performance in terms of locality metrics and isotropy, as well as table size. The look-up tables for two of the Hilbert curves currently implemented—"Butz" and "chI"—are provided in Appendix C.2.

12 Data Structures

12.1 Quad/Octree Cartesian Mesh

The present implementation relies on an implicit Cartesian mesh in d dimensions, $d \in [1, 2, 3]$, which is fully defined by the following properties: the extent of the discretized physical space, given by the vector $\boldsymbol{e} \in \mathbb{R}^d$, a background resolution, given by the vector $\boldsymbol{n} \in \mathbb{N}^d$, which specifies the maximum number of representable cells per dimension, a sorted list of integer indices—"keys"—on the SFC, and a corresponding list of integer cell "levels". Each entry in the list of keys corresponds to the unique point in each cell that is associated with the lowest representable SFC index value in the cell. This is the point at which the SFC enters the cell. In

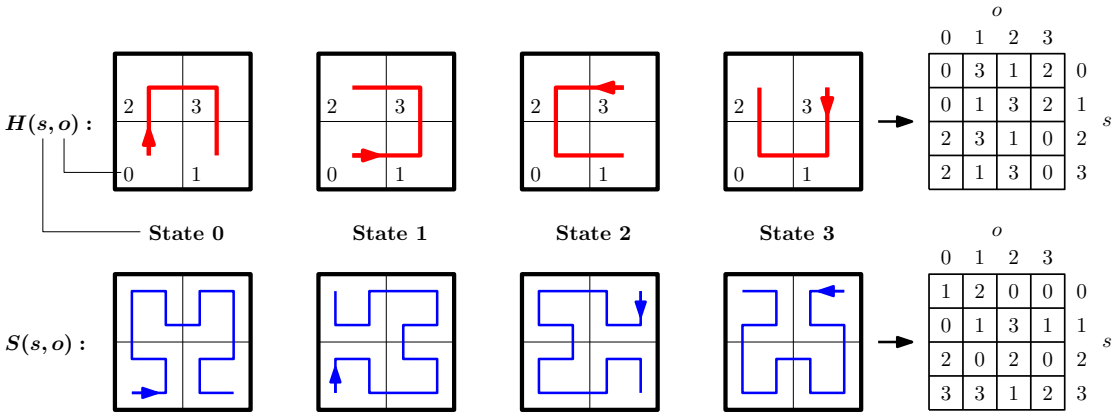


Figure 27: 2D example of Hilbert curve lookup table construction. The base order (state 0, red curve), together with its refinement (state 0, blue curve) completely define the 2D Hilbert curve construction. The encountered transformations (rotations and reflections) of the base order in each quadrant o are labeled as states s with values 0 through 3. The order in which the curve (upper row) visits the four quadrants in each state defines the function H in Equation (102). For the definition of the function S, the state of the refined curve in each of the quadrants (lower row) is tabulated for each state.

the case of "vertex gated" Hilbert curves, this is one of the cells' corners. For the Morton order, it is the cell corner with the lowest coordinate values.

The extent of the cell is determined by the corresponding entry in the list of levels: a level l implies an extent of $2^l\, e_{(i)}/n_{(i)}$, meaning that level 0 cells are the finest representable cells and a cell at level $l+1$ covers 2^d cells at level l. The two lists thus define an implicit 2^d-tree. Note that the background resolution does not need to be a power of two in any dimension and that any cell may be removed from the list, e.g., if it is outside of the fluid volume, as long as the sorted order along the SFC is maintained.

12.2 Particle Data

As described in Part II (cf. also [KJ17]), the entire particle data are stored in a single matrix, with each column corresponding to the state vector of one particle. Grid-particle correspondence is maintained by sorting the columns of the particle matrix by the SFC index corresponding to the position of each particle. Since these indices are integer values, fast sorting algorithms with linear runtime complexity ($\mathcal{O}(N)$) may be used. In the present implementation, multi-threaded, least-significant digit radix sort is used. Note that maintaining grid-particle correspondence by means of sorting was previously described by Green [Gre08] for simulations on graphics processing units. Due to the recursive nature of the SFC, a grid cell with key k_c and level l contains all positions with keys k for which $k_c <= k < k_c + 2^{d\times l}$. This means that it is sufficient to store for each cell the index of the first particle in the

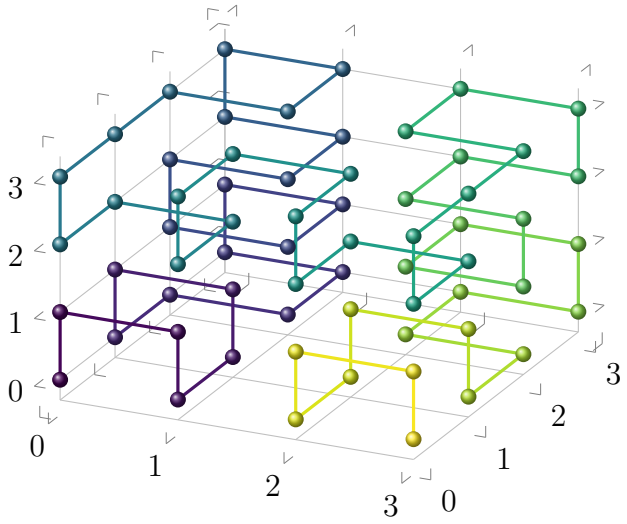


Figure 28: The Hilbert curve "Butz". Shown are points on a $4 \times 4 \times 4$ grid, connected in order of their index along the Hilbert curve.

matrix with key greater or equal to the cell's key but smaller than the key of the next cell. For convenience, also the number of particles in the cell is stored.

To summarize, sorting the particle matrix and counting the number of particles per cell allows for $\mathcal{O}(N)$ complexity insertion of all N particles into an arbitrarily refined quad/octree mesh. There is no need to explicitly trace the particles through the mesh, since the SFC index calculation is mesh-independent. Besides the convenience of being able to identify the parent cell of a given particle solely based on local information (i.e., the particle's coordinates), the ordering of the grid cells along an SFC with good locality property may also benefit the computational performance of the sorting step: when particles move through the mesh, a small change in coordinates also corresponds to only a small change in the SFC index. During the particle sorting step, this means that particle data stored in close proximity in computer memory will be copied to locations that are again close together. In the present multi-threaded implementation, this avoids interference effects ("false sharing") between the threads performing the sorting and data movement.

13 Algorithms

13.1 Parallel Simulation

The parallel simulation algorithm, which was presented in Part II (cf. Algorithm 1 in [KJ17]), remains unchanged by the introduction of a locally refined mesh. In particular, no grid-topology aware processing is required for particle movement or communication. This underlines the elegance of the SFC approach. Below,

the main points of the parallel simulation algorithm are briefly summarized, with emphasis on the parts relevant to the new SFC-based data structures:

Domain decomposition is used to distribute the simulation workload onto multiple processes for the parallel computation on distributed memory architectures. The processes exchange data via the Message Passing Interface (MPI). Each process advances the simulation in a given sub-volume of the total simulation domain. The relevant sub-volume is defined by a contiguous sub-set of the list of cells, sorted by SFC index (cf. Section 13.3 below). Processes only store mesh data associated with its sub-set. A sorted list of the key of the first cell of each sub-set is present on each process and thus specifies the domain decomposition grid. At no point during the simulation do data for a given cell reside on more than a single process, the decomposition is performed fully in parallel.

MPI parallel file output operations are used to write simulation results to disk, without having to gather the data on the root process. To this end, simulation data are output in SFC order using the 1D "subarray" MPI datatype.

As the particles move through the simulation volume, they are communicated to the process responsible for the simulation of their spatial location. Although the sub-volumes formed by the SFC sub-sets may be highly irregular, this step is trivial and constitutes one of the main advantages of the presented simulation architecture: since the particles are sorted by their respective SFC index, they may be simply binned on the domain decomposition grid, and—because they reside contiguously in memory—directly communicated via a single message to the relevant process. Recall that for (FP-)DSMC, no mesh data have to be communicated in order to complete a time step, since the algorithm does not depend on information in neighboring cells.

Further, the fine-grained data parallelism inherent in the stochastic particle simulation is exploited: particles are sampled independently at inflow boundaries and interact independently with solid boundaries and outflows. During sampling of particle distribution moments and application of the collision operator, all operations are independent for each cell. OpenMP is used to execute the independent steps in parallel by a team of threads. As mentioned above (Section 12.2), the necessary sorting of the particle data on each process before and after communication is also done in parallel.

13.2 Automatic Mesh Refinement

Refining a grid cell amounts to replacing the corresponding entry of the cell with index c, key k_c and level $l_c > 0$ in the key and level lists by 2^d entries, each with level $l' = l_c - 1$ and keys beginning at k_c and spaced $2^{d \times l'}$. The corresponding cell data (velocity moments, averages and macroscopic properties) are copied and scaled appropriately in the case of extrinsic properties.

Note that all other cells' keys and levels remain unchanged, since the keys k of the newly inserted cells are guaranteed to have values in the range $k_c \leq k < k_{c+1}$. A 2D example of the implicit refinement by key list modification is shown in Figure 29.

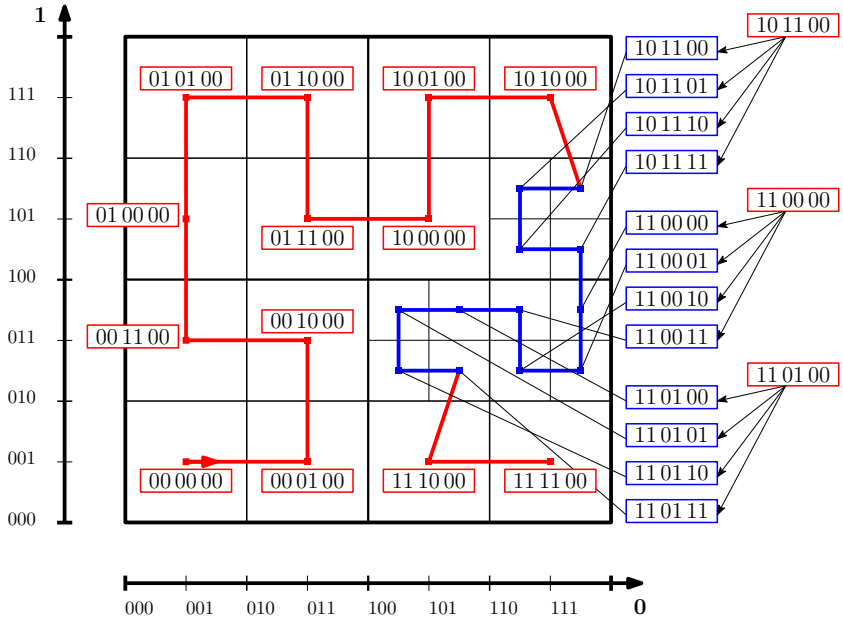


Figure 29: 2D example of mesh refinement in implicit, Hilbert-ordered quad tree. The cells with keys 10 11 00, 11 00 00 and 11 01 00 are refined by replacing their respective key in the list by $2^2 = 4$ new entries, marked in blue.

Various automatic refinement criteria are easy to implement: any map of cell data (e.g., particle distribution moments, macroscopic quantities, cell geometry, etc.) to a true/false value may be passed to the AMR routine as a function handle, in which it is used to flag cells for refinement. Here static and dynamic criteria are distinguished, the former being independent of the simulation result. At present, one static and one dynamic criterion are implemented: before the actual simulation, the cells may be automatically refined depending on their location with respect to (potentially non-participating) geometry, i.e., one may define an arbitrary volume in which the cells should be refined to a certain level prior to the simulation, or specify that cells intersected by geometry are to be refined to a given level. During the simulation, cells can be flagged for refinement based on the ratio of their size to the local equilibrium mean free path λ_0. For pure DSMC simulations, one should choose $d_{\text{cell}} > \lambda_0$ as refinement criterion to ensure mean free path resolution. As explained above, FP-DSMC simulations require less stringent thresholds.

Automatic refinement is carried out for solution independent criteria once before the simulation, and for dynamic criteria later every Δn_{amr}'th time step, between time steps numbered $n_{\text{amr}}^{\text{start}}$ and $n_{\text{amr}}^{\text{end}}$, all three parameters being user input. Typically, one would set $n_{\text{amr}}^{\text{end}} < n_{ta}$, where n_{ta} is the time step after which the flow is assumed to be stationary and time averaging begins. The refinement interval Δn should be chosen low enough to prevent spurious bifurcations of the solution that may otherwise occur due to under-resolving of relevant flow regions, yet high enough to allow for sufficient sample accumulation to mitigate the influence of the

Monte-Carlo noise.

An automatic refinement step proceeds as follows: on each rank, the list of cells is partitioned onto participating threads, which evaluate the refinement criterion for each cell in parallel. A bit vector is set to true at the corresponding index if a cell is marked for refinement. The values of this vector corresponding to cells on the rank domain boundary are communicated to neighboring ranks. Taking these values into account, a second pass over the cell list on each rank modifies the bit vector to ensure a maximum level difference of one level between neighboring cells. Note that cell neighbors may be located efficiently by binary search in the sorted cell key array. Finally, the bit vector is passed to a serial routine that performs the actual splitting/merging of the cells. The total number of modified cells on each rank is summed up on the root rank. As long as the total number of modified cells is greater than zero, the above procedure continues in a loop.

After the refinement, all changed cells are marked for cell volume correction: cells outside the fluid volume are removed, and the respective volumes of cells intersected by geometry are estimated via a Monte Carlo procedure.

13.3 Parallel Load Balancing

The decomposition of the simulation domain discretized by N cells onto P different processes is specified by a list S of $P+1$ cell indices, here referred to as separators, since they separate the SFC index space into compact intervals. Each entry S_i in the separator list corresponds to the index of the first cell along the SFC belonging to process i, and thus process i will advance the simulation in the domain spanned by cells $[S_i, \ldots, S_{i+1})$. A decomposition is only compatible with the presented simulation algorithm if the cell list remains sorted across all processes. This means that re-decomposing the domain can only be accomplished by shifting separator cell indices along the SFC, and the value of the first and last separators S_0 and S_P are fixed at $S_0 \equiv 0$ and $S_P \equiv N$, respectively.

The associated computational problem of finding the optimal separator cells is known as the "Chain-on-Chains Partitioning" (CCP) problem, i.e. "[finding] a sequence of $P-1$ separators to divide a chain of N tasks with associated computational weights $[w]$ into P *consecutive* parts" [PTA08]. A partitioning is optimal, if the load of the maximally loaded part (called the bottleneck value B) is minimized. It is obvious that the ideal bottleneck value B^* is the average load and that the optimal bottleneck value is bounded from above by $B_{\max} = B^* + w_{\max}$. Instead of computing the partitioning via a heuristic, a version of the "exact bisection" (EB) algorithm according to Pinar and Aykanat [PA04; PTA08] is implemented.

EB is an exact algorithm guaranteed to find the optimal partitioning in polynomial run time. It is efficient because it performs binary search on the discrete set of *realizable* bottleneck values $\mathfrak{B}$ in the interval $\mathcal{B} = [B^*, \ldots, B_{\max}]$, instead of having to search the (infinite) set of all bottleneck values $\mathcal{B}$. The computational cost of EB itself is negligible; however, it is not local, i.e., it requires the indi-

vidual computational load of all cells as input. In the present implementation, a distributed, cooperative algorithm is used to avoid having to store this information on the master process.

The computation itself is carried out on the root process, and required values are requested from the other processes, which spin in a loop and use the non-blocking probe MPI function to receive requests from the root process on demand. The appropriate function handle to execute to serve the request is then determined by a unique tag attached to the message. The root process exploits the fact that the cells are distributed in sorted order across the processes, and can thus use binary search to determine which process owns a cell with given global cell index. Once the root process has completed the computation of the optimal separator positions, it notifies the other processes to exit the loop.

Because the original formulation of the EB algorithm does not prevent empty partitions, an additional step is carried out by the root process, where one of two duplicate separator indices is shifted by one position to have at least one cell assigned to each process. Note that this procedure will never alter an optimal bottleneck value, since only the load of previously empty partitions is increased.

The new separator array is then sent from the root process to all other processes and subsequently used to determine which data need to be communicated in order to complete the re-partitioning.

In the present implementation, the total storage cost in bytes incurred by a cell is used as load metric. This corresponds to a fixed value for the storage of the moments of the velocity distribution and the state variables and a dynamic value proportional to the number of computational particles in the cell. The latter value may be run-time configured to be computed from an exponentially weighted time average in order to avoid over-adaptation of the partitioning to the fluctuations in particle numbers. Using only the number of particles in a given cell as load metric would in principle result in faster simulations. Balancing total storage, however, allows for simulation runs with a larger amount of particles and avoids exceeding the available memory on a given process.

Note further that EB is trivially extensible to heterogeneous systems [PTA08] in that it allows to compute optimal partitions also when the actual computational power of the individual machines on which the processes run is taken into account. The performance of each process could be dynamically estimated at run time by the measurement of wall clock time per particle per time step, which is already done in the code for profiling.

In the present implementation, the EB algorithm is also used locally on each MPI process in its “native” (undistributed) form to compute an optimal partitioning of the mesh onto the available OpenMP threads for sampling and collisions. Here, the number of particles per cell is used directly as a load metric, which gives an optimal partitioning for the FP collision operator, since its computational cost is directly proportional to the number of particles. The necessary prefix sum of the load weights is already available from the preceding sorting step in the form of the

array containing the index of the first particle in each cell. For DSMC, the number of collisions in each cell would be a more relevant metric, but would necessitate a further preprocessing step.

13.4 Point-to-Point Communication

Because the domain decomposition is not given by planes aligned with the Cartesian axes, but rather by indices along an SFC, it is not a priori clear which processes will communicate particle information with each other, and every time step may result in a different communication pattern. Since data of particles from a given range of cells along the SFC are kept contiguous in memory, simply counting the number of particles within each partition of the SFC allows to establish the number, size and target processes for particle data exchange on each process. A single "all-to-all" communication allows for the correct transfer of the particle data. When the number of processes is large, however, the fraction of processes that exchange non-empty messages compared to the total number will diminish. A similar situation arises when mesh elements have to be redistributed after a load-balancing step. Most processes will only have to communicate with direct neighbors along the SFC when the domain decomposition does not change much, but this cannot be guaranteed a priori. This situation is known as the "Dynamic Sparse Data Exchange Problem" (DSDE).

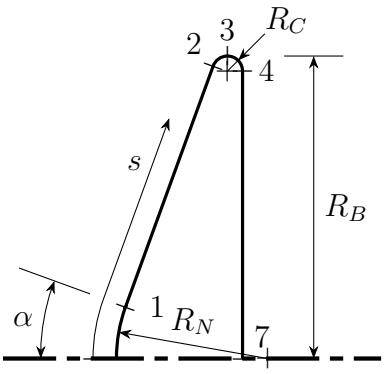
In order to avoid sub-optimal sparse all-to-all communication, the "Nonblocking Consensus" (NBX) communication algorithm due to Hoefler et al. [HSL10; Gro+14] has been implemented. NBX uses non-blocking MPI routines which are functions that return immediately, while the completion of the initiated operation may be checked by calling small test routines. The send operations are furthermore synchronous, i.e., only considered complete when the recipient has finished receiving the message. After issuing all synchronous, non-blocking sends, each process spins in a loop and dynamically receives any incoming message. Once a process has verified that all its initiated sends have completed, it enters a non-blocking barrier, but continues to receive messages dynamically. All send and receive operations are thus guaranteed to have completed once all processes have entered the barrier.

NBX achieves efficient point-to-point data exchange in situations where the communication neighborhood is small compared to the total number of processes. Note that Harlacher et al. [Har+12] already proposed to use, but did not implement, this communication pattern for the redistribution of mesh elements after load balancing.

14 Numerical Example: Flow Over a Planetary Probe Geometry

Reentry flow over a 70 degree blunted cone resembling a simplified planetary probe geometry [ABL97a] is a common test case for rarefied gas flow simulation (see, for example, [DB96; GZS10; GZS11; PG16]), because experimental data are available.

Figure 30 shows the definition of the rotationally symmetric geometry. The flow conditions and simulation parameters are listed in Table 6 in Appendix C.3.



Location	s/R_N
1	0.349
2	2.043
3	2.166
4	2.323
7	4.223

$\alpha = 20°$
$R_B = 2.5\,\text{cm}$
$R_B/R_N = 2.0$
$R_C/R_N = 0.1$

Figure 30: Planetary probe geometry and definition of surface coordinate s.

14.1 Automatic Mesh Refinement and Result Comparison

Figure 31 shows a cutaway of the automatically refined mesh during a typical simulation run. Mesh elements residing on the same processor have the same color. The probe geometry is shown in orange, together with the surface triangulation. The overall structure of the refined mesh is similar to that reported in Figure 5 in [GZS11].

In Figure 32, the simulation results in terms of surface heat flux to the probe geometry are compared to experimental [ABL97b] and DSMC results [GZS11] in the literature, as well as to a reference DSMC simulation, which was obtained using the same parameters as the FP-DSMC simulation (listed in Table 6), but with the time step reduced to $\Delta t_{\text{DSMC}} = 5.0 \times 10^{-8}\,\text{s}$ and more aggressive refinement criterion of $\left(\Delta x/\lambda_0\right)_{\text{crit.}} = 0.5$ to ensure that the collisional scales were resolved. The figure shows the circumferentially averaged values of the heat flux to the surface, plotted versus the non-dimensional surface coordinate s/R_n, as defined in Figure 30. One may observe a slight underprediction of the heat fluxes on the windward side by FP-DSMC compared to pure DSMC, while the overall level of agreement with experimental data is equally satisfactory. The larger differences compared to the results of Gao et al. at $s/R_n \approx 2$ are probably due in part to the much finer surface mesh in the shoulder area, cf. Figure 5 in [GZS11]. Note that the total wall clock

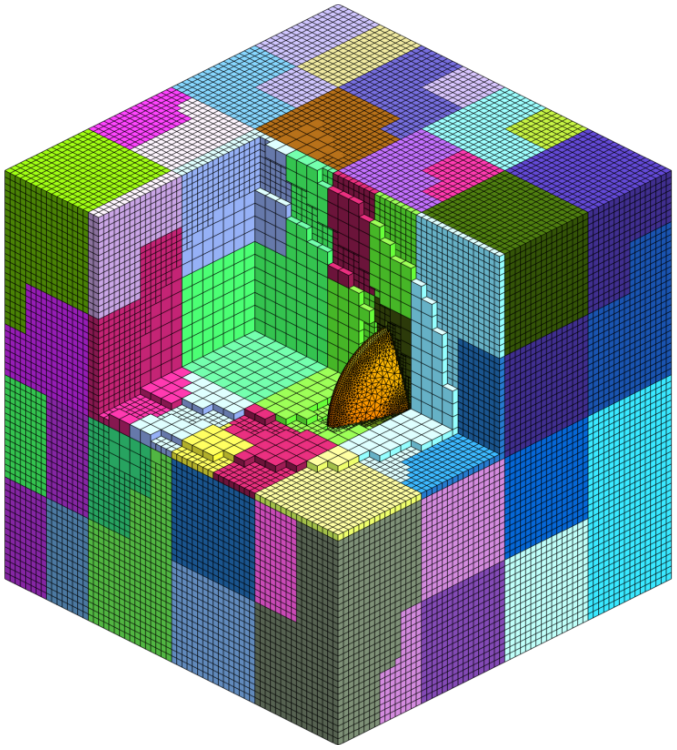


Figure 31: Cutaway of the automatically refined mesh during a typical run of a simulation of flow over a planetary probe geometry. Mesh elements residing on the same processor have the same color. The probe geometry is shown in orange, together with the surface triangulation.

time required for the pure DSMC simulation to simulate the same physical time as the FP-DSMC simulation was higher by a factor of more than 11.

14.2 Performance Comparison of Simulations using different Space-Filling Curves

Normalized total simulation run times are presented for seven runs for each of five different SFCs: the classical Morton order, as well as the Hilbert curves nicknamed "Sasburg", "Alfa", "Butz" and "ChI" in [Hav17]. Recall that computations using different Hilbert curves are performed by simply using different lookup tables. All simulations were carried out on 48 nodes of ETH's "Euler III" cluster[3]. Each node is equipped with a single Intel Xeon E3-1285Lv5 processor and 32 GB of memory. In terms of parallel decomposition, on each node, a single MPI process is run using 8 OpenMP threads on the available 4 processor cores. The Euler III cluster is not equipped with a fast interconnect between the nodes, which means that efficient communication is even more important. Figure 33 shows the normalized lowest run time achieved for each of the SFCs. The results show that only Hilbert curves

[3] `https://scicomp.ethz.ch/wiki/Euler`

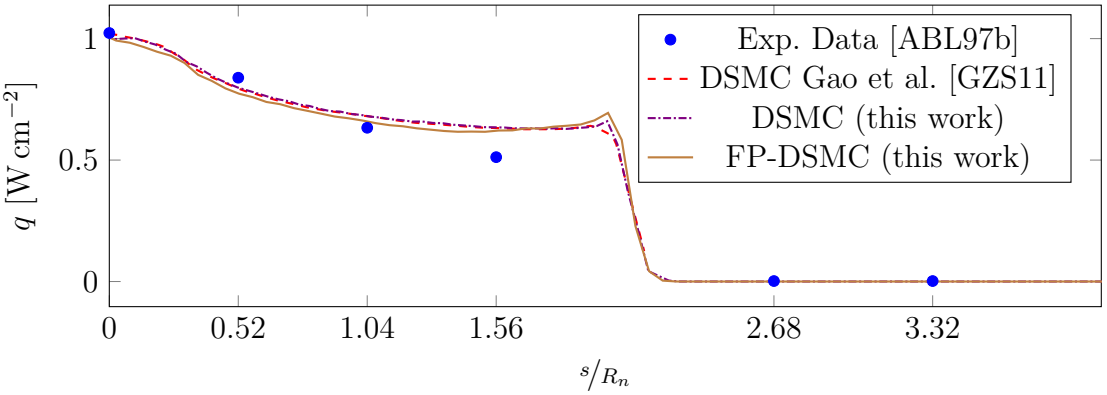


Figure 32: Comparison of heat flux to geometry surface, circumferentially averaged and plotted over the non-dimensional surface coordinate s/R_n, where R_n is the nose radius of the geometry.

"Butz" and "ChI" perform equal or better than the Morton ordering, with the fastest run using the "ChI" order being $\sim 5\,\%$ faster than the fastest run using the Morton ordering. Although the "Sasburg" and "Alfa" curves have very good theoretical properties in terms of space isotropy and locality [Hav17, Table 6], they perform worse than the Morton order. This is probably due to the larger size of the required tables more than offsetting any potential gains from improved communication and memory locality, cf. Section 14.5. The two Hilbert curves offering comparable performance to the Morton order have the smallest lookup tables of those studied.

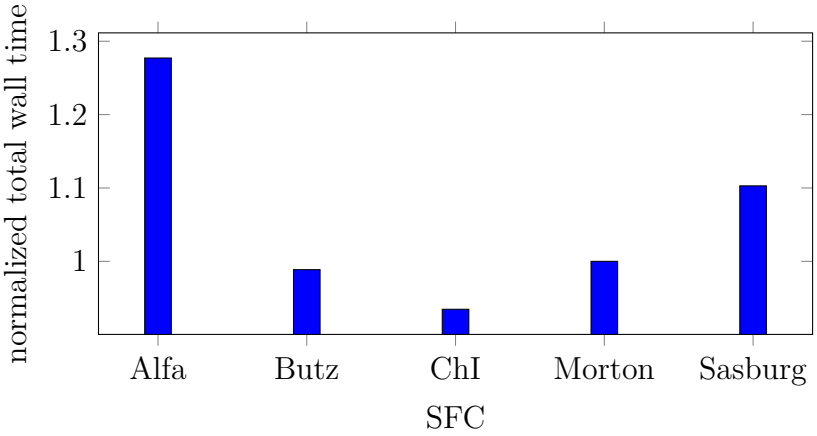


Figure 33: Comparison of lowest total simulation run time over seven runs for each of five different SFCs. The values are normalized by the lowest run time incurred using the Morton order. All simulations were carried out on 48×4 processor cores.

14.3 Evolution of the Load Balance Metric

As mentioned in Section 13.3, the load metric is defined as the combined storage requirement of particle and grid cell data, and measure load balance as the ratio of average load to the maximum load of any process, such that a value of one constitutes perfect load balancing. Figure 34 shows the evolution of the load balance metric during the transient simulation phase (the first 1000 time steps), averaged over seven simulations, when using different space filling curves. The load balance is evaluated every 25 time steps, once before and once after any re-partitioning, which is carried out when the metric falls below 0.98. The results show a strong tendency toward imbalance during the transient simulation phase. The load balancing algorithm, however, achieves near perfect re-balancing. This is evident in the large jumps in the individual curves, which occur when the re-balancing is carried out, and which bring the metric back close to the ideal value of 1.00. Towards the end of the transient phase, the load balance stays approximately constant within the specified corridor of 0.98 to 1.00 , as is to be expected due to the stationary density field. One may further observe that, especially during the beginning of the simulation, the Hilbert orderings tend to yield better load balancing than the Morton order.

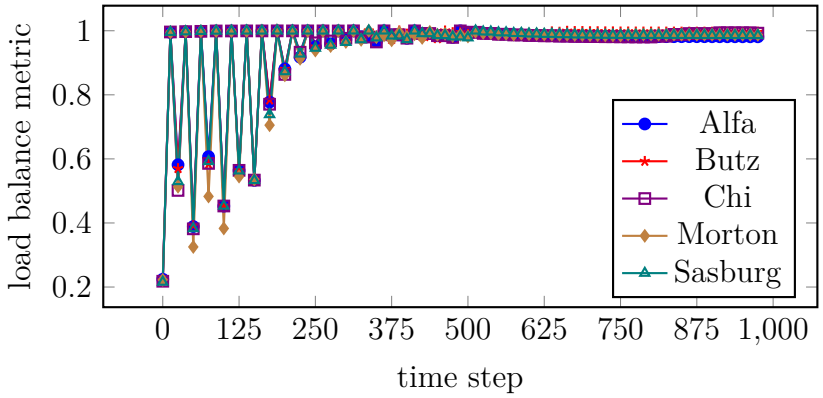


Figure 34: Evolution of the load balance metric during transient simulation phase, averaged over seven runs, using different SFCs. The metric is evaluated twice every 25 time steps, once before and once after any re-partitioning. The large jumps in the curves occur at time steps where the re-partitioning was carried out.

14.4 Parallel Scaling Analysis for the Hilbert Curve “Butz”

Flow over a planetary probe geometry is simulated to examine strong scaling of the implementation. The same simulation is repeated on varying numbers of cluster nodes, each running a single MPI process using 8 threads. The flow and geometrical setup are the same as for the other simulations presented in this work; however, the number of particles is reduced by a factor of four and the total number of time

steps is reduced. The results in Figure 35 show the typical diminishing returns as each thread processes fewer particles and communication overhead starts to dominate. Note that the chosen test case could not complete on a single node due to lack of memory. Further, the runtime cost of file output was subtracted in the shown results.

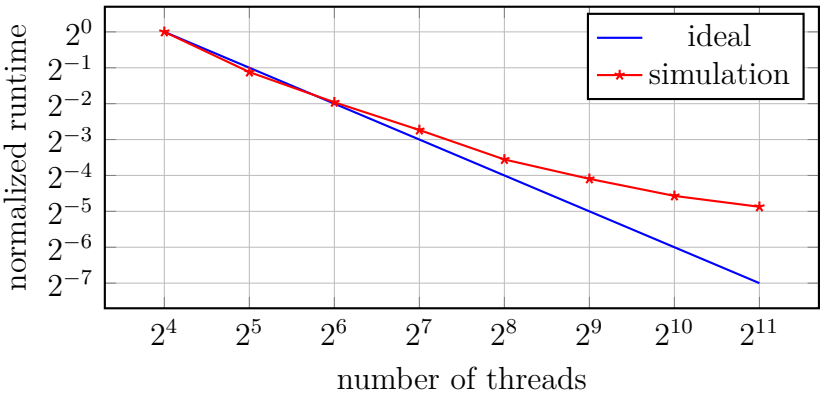


Figure 35: Strong scaling analysis using the Hilbert curve "Butz". All reported runtimes concern simulations of flow over a planetary probe using identical parameters on different numbers of processor cores.

14.5 Relative Computational Cost of the Simulation Subroutines

A representative profiling of the relative computational cost in terms of wall clock time of the different parts of the simulation algorithm is provided in Table 1. The results were obtained using the Hilbert curves "Butz" and "Alfa", as well as the Morton order. Note that automatic load balancing and mesh refinement procedures are only active during the first 1000 of the 2500 time steps, in accordance with the assumption of a stationary density field after that point. Load-balancing was evaluated every 25 time steps and re-balancing was carried out when the metric was below 0.98 (see Section 14.3). Automatic mesh refinement was carried out every 50 time steps. Regardless of the respective intervals of evaluation, it is evident from Table 1 that the automatic refinement and load balancing functions incur negligible computational cost compared to the main simulation routines.

The results further indicate that the SFC look-up table size is indeed a major contributor to computational cost, and thus explains the relatively poorer performance of simulations using the "Alfa" curve. Computing the indices on the "Butz" curve is, however, only slightly more expensive than only computing the Morton index. Its superior locality properties compared to pure Morton ordering are mainly evident in the particle communication step.

Table 1: Representative profiling of relative computational time attributed to different parts of the simulation algorithm, using three different SFCs (simulation parameters listed in Table 6).

subroutine	% of total wall clock time		
	Butz	Alfa	Morton
particle communication	23.66	17.26	25.61
sorting	23.45	16.47	23.64
sampling & collisions	19.89	13.99	20.04
SFC computation	8.62	32.88	6.30
movement & boundaries	6.44	4.68	6.54
inflow	6.41	4.50	6.47
file output	2.07	1.34	1.79
dynamic mesh refinement	0.57	0.53	0.60
particle-to-cell indexing	0.50	0.35	0.50
load balancing	0.20	0.17	0.20
other overhead	8.20	7.82	8.32

15 Numerical Example: Laval Nozzle Flow

In order to investigate the influence of surface compared to volume refinement on SFC related performance, expansion of Argon at near-continuum conditions through a Laval nozzle into a receiving chamber and subsequent interaction with a skimmer geometry is simulated. Similar to the analysis in Section 14.2, the performance impact of using different SFCs is compared. To save computational time, only a quarter of the axisymmetric flow was simulated and symmetry conditions were imposed on the relevant planes. In the receiving chamber, a background pressure of 40 Pa is set as initial condition, as well as a reservoir boundary condition at the cylindrical outer simulation boundary. Figure 36 shows the simulated geometry together with the boundary mesh. The conditions at the nozzle inlet cross section with diameter $d_{\text{in}} = 10\,\text{mm}$ are calculated from the stagnation temperature $T_0 = 293\,\text{K}$ and pressure $p_0 = 3\,\text{kPa}$ using isentropic flow relations. The equilibrium Knudsen number, based on the critical cross section diameter $d_{\text{c}} = 6.24\,\text{mm}$, ranges from $\approx 2.7 \times 10^{-4}$ near the nozzle inlet to $\approx 2.1 \times 10^{-2}$ outside the jet, and $\gtrapprox 5$ after the skimmer. This wide range of Kn covered by the simulation would pose a significant challenge to pure DSMC. The FP-DSMC algorithm, together with a locally refined mesh, makes the simulation tractable. For this study, only static refinement was used: to resolve the axisymmetric flow on the Cartesian grid, all cells in a cylindrical volume around the flow center line were refined. Further, all cells intersected by the nozzle and skimmer geometry were refined as well. Note that the flow through the skimmer itself were not resolved by the mesh chosen for this study. All flow conditions and simulation parameters are listed in Table 7 in Appendix C.3. Figure 37 shows the stationary solution on the symmetry planes in terms of Mach number, together with the mesh.

Five simulations each per space-filling curve “Morton”, “Sasburg”, “Alfa”,

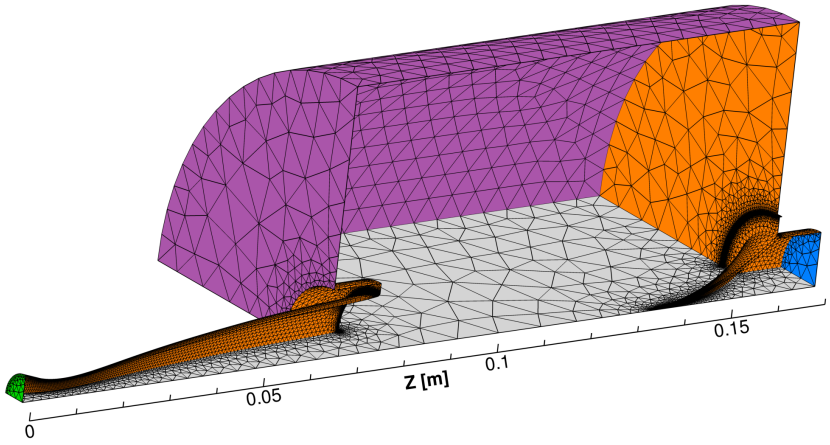


Figure 36: Surface mesh for Laval nozzle flow simulation. The inflow (from the left) is colored green, diffusive surfaces are orange, reservoir boundaries are purple, and the low-pressure reservoir boundary is colored blue. The symmetry boundary in the XZ plane is shown in gray, and the symmetry boundary in the YZ plane is omitted for clarity.

"Butz" and "ChI" were performed. All simulations were carried out on 8 nodes of ETH's "Euler II"cluster, each equipped with two Intel Xeon E5-2680v3 12-core processors and 32GB memory. On each processor, two MPI processes are run using 12 OpenMP threads each, for a total of 32 MPI ranks. The normalized lowest run time achieved for each curve over five runs is reported in Figure 38. Curves "Butz", "ChI" and "Morton" again performed similarly (using "ChI" was 2 % faster than "Morton" or "Butz"), while "Alfa" and "Sasburg" incurred a much higher overhead.

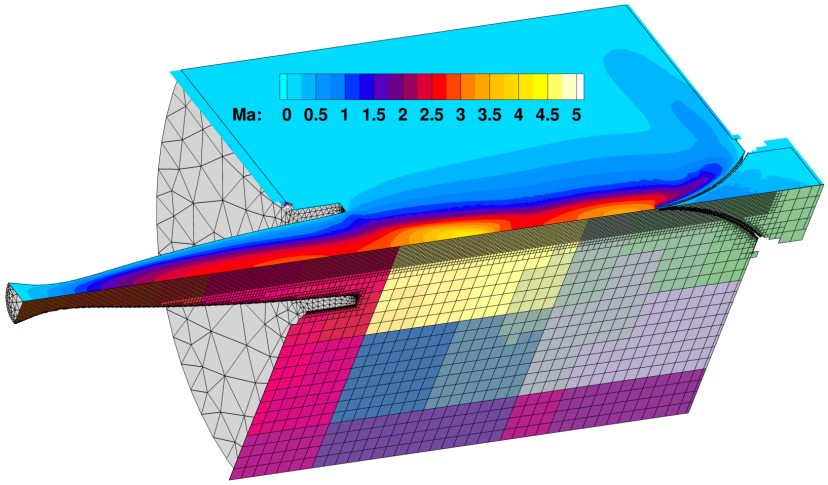


Figure 37: Stationary solution obtained from Laval nozzle flow simulation. The upper symmetry plane is colored by Mach number, the lower plane shows the volume mesh, colored by process.

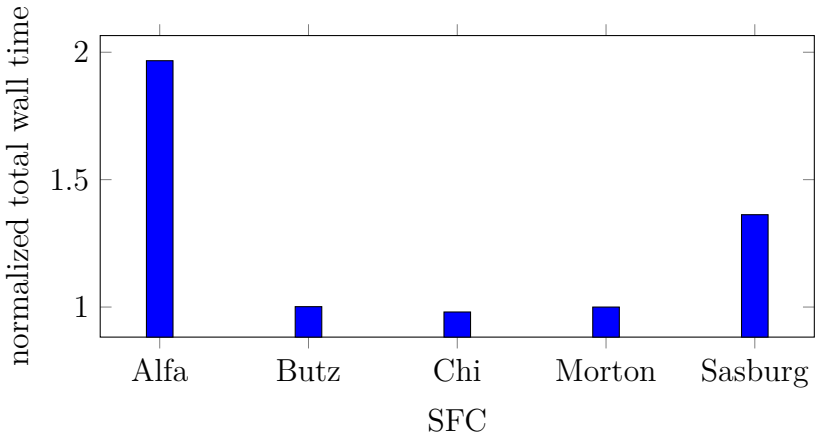


Figure 38: Comparison of lowest total simulation run time over five simulations of Laval nozzle flow for each of five different SFCs. The values are normalized by the lowest run time incurred using the Morton order. All simulations were carried out on 32×6 processor cores.

16 Conclusions

In this part of the thesis, the extension of the parallel Fokker-Planck-DSMC implementation with automatic mesh refinement and load balancing is presented. The use of an implicit octree grid in conjunction with an ordering of the grid cells based on space filling curves allows for an efficient and comparatively succinct implementation of both these features.

Various beneficial properties of using space filling curves in the context of DSMC are discussed, and an efficient implementation of the curve index computation is presented. The mapping of the mesh topology to 1D allows to use the exact load balancing algorithm developed by Pinar and Aykanat [PA04] for both process level domain decomposition and thread level partitioning of the collision and sampling step. The dynamic sparse data exchange problem that arises from the chosen domain decomposition is addressed via the implementation of the non-blocking consensus algorithm due to Hoefler et al. [HT09] for both particle exchange and repartitioning.

Numerical simulation of reentry flow over a planetary probe geometry demonstrates the validity of the implementation via comparison to numerical and experimental results in the literature. Comparing the computational performance of the implementation when using different space filling curves reveals that certain Hilbert curves can offer sufficient advantages in terms of locality of the domain decomposition and the particle data to offset the additional processing step required for their computation. Further, a strong scaling study is presented, which uses the same flow scenario as a test case. Profiling the execution time of the different subroutines of the new simulation algorithm reveals that automatic mesh refinement and load (re-)balancing carry negligible computational cost. However, due to the serial nature of these algorithms, their relative contribution to the overall computational cost will eventually become significant as the number of parallel partitions becomes large.

The findings in terms of the performance impact of using different space filling curves are confirmed for a second test case with higher surface-to-volume mesh refinement ratio: the expansion of Argon through a Laval nozzle and downstream wall interaction of the flow are simulated. This study further demonstrates the ability of the implementation to simulate gas flow covering a wide range of Knudsen numbers, from the near continuum to the rarefied regime.

To the best of the author's knowledge, the parallel implementation presented here is the first (FP-)DSMC code that is fundamentally based on space-filling curves. Furthermore, the comparison of the performance impact on (FP-)DSMC from using different SFCs is unique.

Part IV
Kernel Estimation of Moment Derivatives for Particle Methods

This part is adapted from the manuscript:
S. Küchlin and P. Jenny. "Kernel Estimation of Mixed Spatial Derivatives of Statistics of Scattered Data". Manuscript submitted for publication. 2018.
Most of the text, figures and equations are identical to the corresponding sections of the manuscript.

17 Introduction

The subject of non-parametric density (derivative) estimation has received much attention in the statistics and—more recently—machine intelligence literature, see, for example, [Sin76] and [CM02]. When given as data N realizations x^l, $l = 1, \dots, N$, of a random variable with associated probability density $\varphi(x)$, one is interested in estimating $\varphi(x)$ via the kernel density estimate

$$\widehat{\varphi}(x) = \frac{1}{hN} \sum_{l=1}^{N} K\left(\frac{x - x^l}{h}\right),$$

see, e.g., the seminal work by Epanechnikov [Epa69]. Here, a related problem is considered: estimation of the νth-order ($\nu = 0, 1, \dots$) mixed spatial derivative of the generalized state space moments $\mathrm{E}_\zeta[g(\boldsymbol{\zeta}; \boldsymbol{x})]$ of a joint probability density $f_{\xi\zeta}(\boldsymbol{x}, \boldsymbol{c})$ at a given location $\boldsymbol{x}$, i.e., $\partial_{x_1}^{\alpha_1} \partial_{x_2}^{\alpha_2} \cdots \partial_{x_{d_x}}^{\alpha_{d_x}} \mathrm{E}_\zeta[g(\boldsymbol{\zeta}; \boldsymbol{x})]$ with $\sum_{i=1}^{d_x} \alpha_i = \nu$. To this end, the aim is to derive optimal kernels and their respective asymptotic bias and mean squared error expressions for the estimation of various mixed spatial derivatives of the state space moments. The suggested approach may prove useful for particle-based simulation methods ranging from molecular dynamics, direct simulation Monte Carlo and related rarefied gas dynamics simulation algorithms to astro-physics simulations; as well as for many data analysis tasks.

The remainder of this part of the thesis is organized as follows: after introducing notation (Section 18), the kernel estimator for mixed spatial derivatives of density weighted generalized state space moments is derived, along with necessary consistency conditions for the kernels (Section 19). Next, the convergence properties of the estimator are studied by deriving the relevant asymptotic bias and variance expressions (Section 20), as well as the asymptotically optimal kernel bandwidth (Section 21). Estimation of the unweighted generalized state space moments in terms of the estimates of their density weighted counterparts is considered in Section 22. Furthermore, it is shown in Section 23 that kernel interpolation of mixed spatial derivatives of a (potentially unknown) function may be considered as a special case of the results obtained for the generalized state space moments.

Section 24 deals with the explicit construction of kernel functions: three classes of kernels are derived, and a novel regularization of the associated variational problem is proposed, which generalizes previous results in the literature. A Monte Carlo study (Section 25) illustrates the root mean squared error behavior of various estimates as a function of kernel type and sample size for a relevant test case. Section 26 concludes.

18 Multi-Index Notation

For notational convenience, multi indices are used throughout this part of the thesis. The relevant definitions from Appendix A in [Sei10] are restated (with slight modification) here: a multi index is an ordered n-tuple $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_n) \in \mathbb{N}_0^n$ of integers. For $\boldsymbol{\alpha} \in \mathbb{N}_0^n$, define the length $|\boldsymbol{\alpha}| = \alpha_1 + \alpha_2 + \cdots + \alpha_n$ and factorial $\boldsymbol{\alpha}! = \alpha_1!\alpha_2! \cdots \alpha_n!$, respectively. Addition and subtraction, multiplication by a scalar, and comparison of multi indices are defined component-wise, i.e., for $\boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathbb{N}_0^n$, $a \in \mathbb{N}$: $\boldsymbol{\alpha} \pm \boldsymbol{\beta} = (\alpha_1 \pm \beta_1, \alpha_2 \pm \beta_2, \ldots, \alpha_n \pm \beta_n)$, $\boldsymbol{\alpha} \leq \boldsymbol{\beta}$ if $(\alpha_1 \leq \beta_1, \alpha_2 \leq \beta_2, \ldots, \alpha_n \leq \beta_n)$ and $a\boldsymbol{\alpha} = (a\alpha_1, a\alpha_2, \ldots, a\alpha_n)$. Note that subtraction $\boldsymbol{\alpha} - \boldsymbol{\beta}$ is only defined for $\boldsymbol{\beta} \leq \boldsymbol{\alpha}$.

Further, for a vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$, define $\boldsymbol{x}^{\boldsymbol{\alpha}} = x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_n^{\alpha_n}$. The mixed partial derivative operator w.r.t. coordinates $\boldsymbol{x}$ is abbreviated as $\boldsymbol{\partial}^{\boldsymbol{\alpha}} = \partial_1^{\alpha_1} \partial_2^{\alpha_2} \cdots \partial_n^{\alpha_n} = \frac{\partial^{|\alpha|}}{\partial_{x_1}^{\alpha_1} \partial_{x_2}^{\alpha_2} \cdots \partial_{x_n}^{\alpha_n}}$. Some special multi indices are $\mathbf{0} = (0, \ldots, 0)$, $\mathbf{1} = (1, \ldots, 1)$ and $\boldsymbol{\delta}_i = (\delta_{1i}, \delta_{2i}, \ldots, \delta_{ni})$, where δ_{ij} is the Kronecker delta. Consequently, $|\mathbf{0}| = 0$, $\mathbf{0}! = 1$, $\boldsymbol{x}^{\mathbf{0}} = 1$, $\boldsymbol{\partial}^{\mathbf{0}} f(\boldsymbol{x}) = f(\boldsymbol{x})$, $|\mathbf{1}| = n$, $\mathbf{1}! = 1$, $\boldsymbol{x}^{\mathbf{1}} = \prod_{j=1}^n x_j$ and $\boldsymbol{x}^{\boldsymbol{\delta}_i} = x_i$, where here and in the following, in expressions such as $\boldsymbol{x}^{\mathbf{1}}$, the dimension of the multi index in the exponent is implied by the dimension of the vector.

19 Kernel Estimate of Mixed Spatial Derivatives of Generalized State Space Moments

19.1 Definitions

Recall the definition of the phase density from Part I, Section 2.1, restated here with some modifications: multi indices are used to remain consistent with the notation introduced above, the system under consideration is no longer restricted to consist of an ensemble of molecules, and the explicit time dependence is dropped for notational convenience.

Given data in a $d_x + d_c$-dimensional phase space, comprised of d_x-dimensional configuration space with coordinate vector $\boldsymbol{x} \in \mathbb{R}^{d_x}$, and d_c-dimensional state space with coordinate vector $\boldsymbol{c} \in \mathbb{R}^{d_c}$, this part of the thesis considers the case when said data may be assumed to be realizations of the absolutely continuous random vectors $\boldsymbol{\xi} \colon \Omega \to \mathbb{R}^{d_x}$ and $\boldsymbol{\zeta} \colon \Omega \to \mathbb{R}^{d_c}$, respectively, where Ω denotes the sample space. Let $\mathcal{B}(\mathbb{R}^n)$ be the Borel σ-algebra of subsets of $\mathbb{R}^n$. Define the (non-normalized) phase

density $f\colon \mathbb{R}^{d_x} \times \mathbb{R}^{d_c} \to \mathbb{R}$, such that the probability of finding a data point in the region $B \in \mathcal{B}\Big(\mathbb{R}^{d_x} \times \mathbb{R}^{d_c}\Big)$ is given by

$$\Pr(\{\boldsymbol{\xi}, \boldsymbol{\zeta}\} \in B) = \frac{1}{\iint_{\mathbb{R}^{d_x}\times\mathbb{R}^{d_c}} f(\boldsymbol{x}', \boldsymbol{c}')\,\mathrm{d}\boldsymbol{x}'^{\mathbf{1}}\mathrm{d}\boldsymbol{c}'^{\mathbf{1}}} \iint_B f(\boldsymbol{x}', \boldsymbol{c}')\,\mathrm{d}\boldsymbol{x}'^{\mathbf{1}}\mathrm{d}\boldsymbol{c}'^{\mathbf{1}}, \tag{103}$$

where $\mathrm{d}\boldsymbol{x}'^{\mathbf{1}}\mathrm{d}\boldsymbol{c}'^{\mathbf{1}}$ denotes $\mathrm{d}x'_1\mathrm{d}x'_2\cdots\mathrm{d}x'_{d_x}\mathrm{d}c'_1\mathrm{d}c'_2\cdots\mathrm{d}c'_{d_c}$. Hence, $f(\boldsymbol{x}, \boldsymbol{c})$ must be non-negative and a Borel function. The phase density may be factored into the (configuration) density $\rho(\boldsymbol{x})$ and the normalized conditional state probability density $f_{\boldsymbol{\zeta}}(\boldsymbol{c}, \boldsymbol{x})$ satisfying $\int_{\mathbb{R}^{d_c}} f_{\boldsymbol{\zeta}}(\boldsymbol{c}, \boldsymbol{x})\mathrm{d}\boldsymbol{c}^{\mathbf{1}} \equiv 1$, i.e.,

$$f(\boldsymbol{x}, \boldsymbol{c}) = \rho(\boldsymbol{x}) f_{\boldsymbol{\zeta}}(\boldsymbol{c}, \boldsymbol{x}). \tag{104}$$

Define the normalization constant in Equation (103) as

$$\mathcal{M} = \iint_{\mathbb{R}^{d_x}\times\mathbb{R}^{d_c}} f(\boldsymbol{x}', \boldsymbol{c}')\,\mathrm{d}\boldsymbol{x}'^{\mathbf{1}}\mathrm{d}\boldsymbol{c}'^{\mathbf{1}} = \int_{\mathbb{R}^{d_x}} \rho(\boldsymbol{x}')\mathrm{d}\boldsymbol{x}'^{\mathbf{1}}. \tag{105}$$

In the case where $\rho(\boldsymbol{x})$ corresponds to a physical density with units of mass per volume, $\mathcal{M}$ is equal to the total mass under consideration. The joint probability density of the random vectors $\{\boldsymbol{\xi}, \boldsymbol{\zeta}\}$ follows as

$$f_{\boldsymbol{\xi}\boldsymbol{\zeta}}(\boldsymbol{x}, \boldsymbol{c}) = \frac{1}{\mathcal{M}} f(\boldsymbol{x}, \boldsymbol{c}). \tag{106}$$

Finally, the expectation operator acting on Borel measurable functions $g\colon \mathbb{R}^{d_x} \times \mathbb{R}^{d_c} \times \mathbb{R}^{d_y} \to \mathbb{R}$ of $\{\boldsymbol{\xi}, \boldsymbol{\zeta}; \boldsymbol{y}\}$, where the vector $\boldsymbol{y} \in \mathbb{R}^{d_y}$ denotes non-random elements in the domain of g, is defined as usual:

$$\mathrm{E}_{\boldsymbol{\xi}\boldsymbol{\zeta}}[g(\boldsymbol{\xi}, \boldsymbol{\zeta}; \boldsymbol{y})] = \iint_{\mathbb{R}^{d_x}\times\mathbb{R}^{d_c}} g(\boldsymbol{x}', \boldsymbol{c}', \boldsymbol{y}) f_{\boldsymbol{\xi}\boldsymbol{\zeta}}(\boldsymbol{x}', \boldsymbol{c}')\mathrm{d}\boldsymbol{x}'^{\mathbf{1}}\mathrm{d}\boldsymbol{c}'^{\mathbf{1}}. \tag{107}$$

In the following, only functions g for which $\mathrm{E}[|g|] < \infty$ and $\mathrm{E}[g^2] < \infty$ exist are considered. Further, the discussion will be restricted to the case where g is not a function of $\boldsymbol{\xi}$, but may depend on the measure space coordinate $\boldsymbol{x}$. Accordingly, one may set $\boldsymbol{y} \equiv \boldsymbol{x}$ and $g = g(\boldsymbol{\zeta}; \boldsymbol{x})$. In this case,

$$\mathrm{E}_{\boldsymbol{\xi}\boldsymbol{\zeta}}[g(\boldsymbol{\zeta}; \boldsymbol{x})] = \mathrm{E}_{\boldsymbol{\zeta}}[g(\boldsymbol{\zeta}; \boldsymbol{x})] = \int_{\mathbb{R}^{d_c}} g(\boldsymbol{c}', \boldsymbol{x}) f_{\boldsymbol{\zeta}}(\boldsymbol{c}', \boldsymbol{x})\mathrm{d}\boldsymbol{c}'^{\mathbf{1}}. \tag{108}$$

Define $F_g\colon \mathbb{R}^{d_x} \to \mathbb{R}$ as the density weighted, generalized state space moments with spatial dependence, viz.

$$\begin{aligned} F_g(\boldsymbol{x}) &= \mathrm{E}_{\boldsymbol{\zeta}}[\rho(\boldsymbol{x}) g(\boldsymbol{\zeta}; \boldsymbol{x})] \\ &= \int_{\mathbb{R}^{d_c}} \rho(\boldsymbol{x}) g(\boldsymbol{c}', \boldsymbol{x}) f_{\boldsymbol{\zeta}}(\boldsymbol{c}', \boldsymbol{x})\mathrm{d}\boldsymbol{c}'^{\mathbf{1}} \\ &= \int_{\mathbb{R}^{d_c}} g(\boldsymbol{c}', \boldsymbol{x}) f(\boldsymbol{c}', \boldsymbol{x})\mathrm{d}\boldsymbol{c}'^{\mathbf{1}}. \end{aligned} \tag{109}$$

The following sections are concerned with the estimation of the mixed spatial derivatives $\boldsymbol{\partial}^{\boldsymbol{\alpha}} F_g$ from sample data.

19.2 Derivation of the Kernel Estimate

To proceed, the integrand in Equation (109) is rewritten in terms of a convolution in the measure space of $\boldsymbol{\xi}$ with the limit of a sequence of suitable functions $\phi_{\boldsymbol{h}}\colon \mathbb{R}^{d_x} \to \mathbb{R}$ with $\boldsymbol{h} \in \mathbb{R}^{d_x},\ h_i > 0$, as follows ([DeV78, Chapter 7, Lemma 1]): for $\phi\colon \mathbb{R}^{d_x} \to \mathbb{R}$ absolutely integrable and appropriately differentiable, with compact support on the d_x-dimensional interval $[-1,1]^{d_x}$, satisfying $\int_{\mathbb{R}^{d_x}} \phi(\boldsymbol{x})\mathrm{d}\boldsymbol{x}^{\mathbf{1}} = 1$,

$$g(\boldsymbol{c}',\boldsymbol{x})f(\boldsymbol{c}',\boldsymbol{x}) = \lim_{\boldsymbol{h}\to\mathbf{0}^+} \int_{\mathbb{R}^{d_x}} g(\boldsymbol{c}',\boldsymbol{x}')f(\boldsymbol{c}',\boldsymbol{x}') \underbrace{\frac{1}{\boldsymbol{h}^{\mathbf{1}}}\phi\Big(\mathrm{diag}(\boldsymbol{h})^{-1}(\boldsymbol{x}-\boldsymbol{x}')\Big)}_{\phi_{\boldsymbol{h}}} \mathrm{d}\boldsymbol{x}'^{\mathbf{1}}, \quad (110)$$

where $\boldsymbol{h}^{\mathbf{1}} = \boldsymbol{h}^{(1,1,\ldots,1)} = \prod_{j=1}^{d_x} h_j$. Note that here and throughout, $[a,b]^n$ denotes the set $\{\boldsymbol{x} \in \mathbb{R}^n : a \le x_i \le b,\ i = 1,\ldots,n\} \in \mathcal{B}(\mathbb{R}^n)$, i.e., an n-dimensional cube with edge length $b-a$. Inserting Equation (110) into Equation (109), one finds

$$\begin{aligned} F_g(\boldsymbol{x}) &= \lim_{\boldsymbol{h}\to\mathbf{0}^+} \iint_{\mathbb{R}^{d_x}\times\mathbb{R}^{d_c}} g(\boldsymbol{c}',\boldsymbol{x}')f(\boldsymbol{c}',\boldsymbol{x}')\frac{1}{\boldsymbol{h}^{\mathbf{1}}}\phi\Big(\mathrm{diag}(\boldsymbol{h})^{-1}(\boldsymbol{x}-\boldsymbol{x}')\Big)\mathrm{d}\boldsymbol{x}'^{\mathbf{1}}\mathrm{d}\boldsymbol{c}'^{\mathbf{1}} \\ &= \mathcal{M} \lim_{\boldsymbol{h}\to\mathbf{0}^+} \mathrm{E}_{\boldsymbol{\xi}\boldsymbol{\zeta}}\Big[g(\boldsymbol{\zeta},\boldsymbol{\xi})\frac{1}{\boldsymbol{h}^{\mathbf{1}}}\phi\Big(\mathrm{diag}(\boldsymbol{h})^{-1}(\boldsymbol{x}-\boldsymbol{\xi})\Big)\Big]. \end{aligned} \quad (111)$$

The νth-order mixed spatial derivative $\boldsymbol{\partial}^{\boldsymbol{\alpha}}F_g$ of F_g, with $|\boldsymbol{\alpha}| = \nu$, follows as

$$\begin{aligned} \boldsymbol{\partial}^{\boldsymbol{\alpha}}F_g(\boldsymbol{x}) &= \mathcal{M} \lim_{\boldsymbol{h}\to\mathbf{0}^+} \mathrm{E}_{\boldsymbol{\xi}\boldsymbol{\zeta}}\Big[g(\boldsymbol{\zeta},\boldsymbol{\xi})\ \boldsymbol{\partial}^{\boldsymbol{\alpha}}\frac{1}{\boldsymbol{h}^{\mathbf{1}}}\phi\Big(\mathrm{diag}(\boldsymbol{h})^{-1}(\boldsymbol{x}-\boldsymbol{\xi})\Big)\Big] \\ &= \mathcal{M} \lim_{\boldsymbol{h}\to\mathbf{0}^+} \mathrm{E}_{\boldsymbol{\xi}\boldsymbol{\zeta}}\Big[\frac{1}{\boldsymbol{h}^{\boldsymbol{\alpha}+\mathbf{1}}}\phi^{(\boldsymbol{\alpha})}\Big(\mathrm{diag}(\boldsymbol{h})^{-1}(\boldsymbol{x}-\boldsymbol{\xi})\Big)g(\boldsymbol{\zeta},\boldsymbol{\xi})\Big], \end{aligned} \quad (112)$$

where the notation $\phi^{(\boldsymbol{\alpha})}$ signifies $\frac{\partial^{|\alpha|}}{\partial_{r_1}^{\alpha_1}\partial_{r_2}^{\alpha_2}\cdots\partial_{r_n}^{\alpha_n}}\phi(\boldsymbol{r})$.

An estimate $\widehat{\boldsymbol{\partial}^{\boldsymbol{\alpha}}F_g}(\boldsymbol{x})$ of $\boldsymbol{\partial}^{\boldsymbol{\alpha}}F_g(\boldsymbol{x})$ may be obtained by fixing $\boldsymbol{h}$ and estimating the expected value in Equation (112) via the weighted average of a finite number N of samples, each with index l, $l = 1,\ldots,N$, weight w^l, position $\boldsymbol{x}^l$, and with state vector $\boldsymbol{c}^l$, viz.

$$\widehat{\boldsymbol{\partial}^{\boldsymbol{\alpha}}F_g}(\boldsymbol{x}) = \frac{1}{\boldsymbol{h}^{\boldsymbol{\alpha}+\mathbf{1}}}\sum_{l=1}^{N} w^l K_{\boldsymbol{\alpha}}\Big((\mathrm{diag}\,\boldsymbol{h})^{-1}\Big(\boldsymbol{x}-\boldsymbol{x}^l\Big)\Big)g(\boldsymbol{c}^l,\boldsymbol{x}^l), \quad (113)$$

where the *kernel* $K_{\boldsymbol{\alpha}}$ is the νth-order mixed partial derivative $\phi^{(\boldsymbol{\alpha})}$, with $|\boldsymbol{\alpha}| = \nu$, of the function ϕ, and the weights are assumed to satisfy

$$w^l > 0, \quad \sum_{l=1}^{N} w^l \equiv \mathcal{M}, \quad \text{and} \quad \lim_{N\to\infty} w^l = 0, \quad (114)$$

which also implies $\sum_{l=1}^{N}\Big(w^l\Big)^2 = 0$. Define the random variable $\eta\colon \Omega \to \mathbb{R}$, which depends on the random vectors $\boldsymbol{\zeta}$ and $\boldsymbol{\xi}$ as follows:

$$\eta = \frac{1}{\boldsymbol{h}^{\boldsymbol{\alpha}+\mathbf{1}}}K_{\boldsymbol{\alpha}}\Big((\mathrm{diag}\,\boldsymbol{h})^{-1}(\boldsymbol{x}-\boldsymbol{\xi})\Big)g(\boldsymbol{\zeta},\boldsymbol{\xi}). \quad (115)$$

The expected value of η may be calculated w.r.t. $\boldsymbol{\xi}$ and $\boldsymbol{\zeta}$, i.e., $\mathrm{E}[\eta] \equiv \mathrm{E}_{\boldsymbol{\xi}\boldsymbol{\zeta}}[\eta]$. For all $K_{\boldsymbol{\alpha}}$ and $f_{\boldsymbol{\xi}\boldsymbol{\zeta}}$ considered here, η is absolutely continuous and satisfies $\mathrm{E}[|\eta|] < \infty$ and $\mathrm{E}[\eta^2] < \infty$. Let $\{\eta^i,\ i : 1 \leq i \leq N\}$ be a collection of independent, identically distributed random variables with the same distribution as η. The estimate (113) may be written as a random variable in terms of the η^i as

$$\widehat{\boldsymbol{\partial}^{\boldsymbol{\alpha}} F_g}(\boldsymbol{x}) = \sum_{l=1}^{N} w^l \eta^l, \tag{116}$$

with expected value

$$\mathrm{E}\Big[\widehat{\boldsymbol{\partial}^{\boldsymbol{\alpha}} F_g}(\boldsymbol{x})\Big] = \sum_{l=1}^{N} w^l E\Big[\eta^l\Big] = \mathcal{M} E[\eta]. \tag{117}$$

Under the assumptions (114), the following strong law of large numbers holds ([LW07, Proposition 4.3]):

$$\Pr\Big(\lim_{N\to\infty} \widehat{\boldsymbol{\partial}^{\boldsymbol{\alpha}} F_g} = \mathcal{M}\,\mathrm{E}[\eta]\Big) = 1. \tag{118}$$

Accordingly, $\mathcal{M}\,\mathrm{E}[\eta] = \boldsymbol{\partial}^{\boldsymbol{\alpha}} F_g$ is a necessary condition for $\widehat{\boldsymbol{\partial}^{\boldsymbol{\alpha}} F_g}$ to consistently estimate $\boldsymbol{\partial}^{\boldsymbol{\alpha}} F_g$. For suitable $K_{\boldsymbol{\alpha}}$, Equation (112) shows that this condition is met in the limit $\boldsymbol{h} \to \mathbf{0}^+$. Consequently, set $\boldsymbol{h} = \boldsymbol{h}(N)$, $h_i > 0$, $i = 1, \ldots, d_x$, while requiring

$$\lim_{N\to\infty} h_i = 0, \quad \text{as well as} \quad \lim_{N\to\infty} \frac{\sum_{l=1}^{N} \Big(w^l\Big)^2}{\boldsymbol{h}^{\boldsymbol{2\alpha+1}}} = 0. \tag{119}$$

The h_i will be referred to as kernel bandwidths in the following, and, for dimensional data $\boldsymbol{x}^l$, are assumed to have the same units as the coordinates $\boldsymbol{x}$, so that the kernel may be defined in a dimensionless reference frame. Under the assumptions (114) and (119), it follows that the estimate (113) converges almost surely for suitable $K_{\boldsymbol{\alpha}}$, i.e.,

$$\Pr\Big(\lim_{N\to\infty} \widehat{\boldsymbol{\partial}^{\boldsymbol{\alpha}} F_g} = \boldsymbol{\partial}^{\boldsymbol{\alpha}} F_g\Big) = 1. \tag{120}$$

19.3 Kernel Consistency Constraints

In the following, constraints on the kernel $K_{\boldsymbol{\alpha}}$ are derived in order to fulfill the condition $\mathcal{M} \lim_{\boldsymbol{h}\to\mathbf{0}^+} \mathrm{E}[\eta] = \boldsymbol{\partial}^{\boldsymbol{\alpha}} F_g$. For η as defined in Equation (115), one obtains

$$\begin{aligned}
\mathcal{M}\,\mathrm{E}[\eta] &= \iint_{\mathbb{R}^{d_x}\times\mathbb{R}^{d_c}} \frac{1}{\boldsymbol{h}^{\boldsymbol{\alpha+1}}} K_{\boldsymbol{\alpha}}\Big((\operatorname{diag}\boldsymbol{h})^{-1}(\boldsymbol{x}-\boldsymbol{x}')\Big) g(\boldsymbol{c}',\boldsymbol{x}') f(\boldsymbol{x}',\boldsymbol{c}') \mathrm{d}\boldsymbol{x}'^{\mathbf{1}} \mathrm{d}\boldsymbol{c}'^{\mathbf{1}} \\
&= \frac{1}{\boldsymbol{h}^{\boldsymbol{\alpha+1}}} \int_{\mathbb{R}^{d_x}} K_{\boldsymbol{\alpha}}\Big((\operatorname{diag}\boldsymbol{h})^{-1}(\boldsymbol{x}-\boldsymbol{x}')\Big) F_g(\boldsymbol{x}') \mathrm{d}\boldsymbol{x}'^{\mathbf{1}} \\
&= \frac{1}{\boldsymbol{h}^{\boldsymbol{\alpha}}} \int_{\mathbb{R}^{d_x}} K_{\boldsymbol{\alpha}}(\boldsymbol{y}') F_g(\boldsymbol{x} - (\operatorname{diag}\boldsymbol{h})\boldsymbol{y}') \mathrm{d}\boldsymbol{y}'^{\mathbf{1}},
\end{aligned} \tag{121}$$

where the substitution $x'_i := x_i - h_i y'_i$, $\mathrm{d}x'_i = -h_i \mathrm{d}y'_i$, and $\lim\limits_{x'_i \to \pm\infty} x'_i = \mp \lim\limits_{x'_i \to \pm\infty} y'_i$, $i = 1, \ldots, d_x$ was used; and $\mathrm{d}\boldsymbol{y'^{\mathbf{1}}}$ denotes $\mathrm{d}y'_1 \mathrm{d}y'_2 \cdots \mathrm{d}y'_{d_x}$. Replacing the expression $F_g(\boldsymbol{x} - (\operatorname{diag} \boldsymbol{h})\boldsymbol{y}')$ in Equation (121) by the corresponding multivariate Taylor series around the point $\boldsymbol{x}$ yields

$$\begin{aligned}
\mathcal{M}\,\mathrm{E}[\eta] &= \frac{1}{\boldsymbol{h}^{\boldsymbol{\alpha}}} \int_{\mathbb{R}^{d_x}} K_{\boldsymbol{\alpha}}(\boldsymbol{y}') \sum_{|\boldsymbol{\gamma}|=0}^{\infty} \frac{(-1)^{|\boldsymbol{\gamma}|}}{\boldsymbol{\gamma}!} \boldsymbol{h}^{\boldsymbol{\gamma}} \boldsymbol{y}'^{\boldsymbol{\gamma}}\, \boldsymbol{\partial}^{\boldsymbol{\gamma}} F_g(\boldsymbol{x}) \mathrm{d}\boldsymbol{y}'^{\mathbf{1}} \\
&= \frac{1}{\boldsymbol{h}^{\boldsymbol{\alpha}}} \sum_{|\boldsymbol{\gamma}|=0}^{\infty} \frac{(-1)^{|\boldsymbol{\gamma}|}}{\boldsymbol{\gamma}!} \boldsymbol{h}^{\boldsymbol{\gamma}} \mu_{\boldsymbol{\gamma}}(K_{\boldsymbol{\alpha}})\, \boldsymbol{\partial}^{\boldsymbol{\gamma}} F_g(\boldsymbol{x}) \\
&\left(= \frac{1}{\prod_{i=1}^{d_x} h_i^{\alpha_i}} \left\{ \mu_{\mathbf{0}}(K_{\boldsymbol{\alpha}}) F_g(\boldsymbol{x}) - \sum_{j_1=1}^{d_x} h_{j_1} \mu_{\boldsymbol{\delta}_{j_1}}(K_{\boldsymbol{\alpha}}) \partial_{j_1} F_g(\boldsymbol{x}) \right.\right. \\
&\qquad + \frac{1}{2} \sum_{j_1=1}^{d_x} \sum_{j_2=1}^{d_x} h_{j_1} h_{j_2} \mu_{\left(\boldsymbol{\delta}_{j_1} + \boldsymbol{\delta}_{j_2}\right)}(K_{\boldsymbol{\alpha}}) \partial_{j_1} \partial_{j_2} F_g(\boldsymbol{x}) \\
&\left.\left. - \frac{1}{6} \sum_{j_1=1}^{d_x} \sum_{j_2=1}^{d_x} \sum_{j_3=1}^{d_x} h_{j_1} h_{j_2} h_{j_3} \mu_{\left(\boldsymbol{\delta}_{j_1} + \boldsymbol{\delta}_{j_2} + \boldsymbol{\delta}_{j_3}\right)}(K_{\boldsymbol{\alpha}}) \partial_{j_1} \partial_{j_2} \partial_{j_3} F_g(\boldsymbol{x}) + \text{h.o.t.} \right\} \right),
\end{aligned} \tag{122}$$

where each term in the series is characterized by the multi index $\boldsymbol{\gamma}$, and

$$\mu_{\boldsymbol{\gamma}}(K_{\boldsymbol{\alpha}}) = \int_{\mathbb{R}^{d_x}} \boldsymbol{y}'^{\boldsymbol{\gamma}} K_{\boldsymbol{\alpha}}(\boldsymbol{y}') \mathrm{d}\boldsymbol{y}'^{\mathbf{1}} = \int_{\mathbb{R}^{d_x}} y'^{\gamma_1}_1 y'^{\gamma_2}_2 \cdots y'^{\gamma_{d_x}}_{d_x} K_{\boldsymbol{\alpha}}(\boldsymbol{y}') \mathrm{d}\boldsymbol{y}'^{\mathbf{1}} \tag{123}$$

are the moments of order $|\boldsymbol{\gamma}|$ of the kernel $K_{\boldsymbol{\alpha}}$. It follows that $K_{\boldsymbol{\alpha}}$ should satisfy the following moment conditions to ensure $\mathcal{M} \lim\limits_{\boldsymbol{h} \to \mathbf{0}^+} \mathrm{E}[\eta] = \boldsymbol{\partial}^{\boldsymbol{\alpha}} F_g$:

$$\mu_{\boldsymbol{\gamma}}(K_{\boldsymbol{\alpha}}) \stackrel{!}{=} \begin{cases} 0, & \boldsymbol{\gamma} \neq \boldsymbol{\alpha},\ |\boldsymbol{\gamma}| < k \\ (-1)^{|\boldsymbol{\gamma}|} \boldsymbol{\gamma}! & \boldsymbol{\gamma} = \boldsymbol{\alpha}, \end{cases} \tag{124}$$

where $k = \nu + 2i$ with $\nu = |\boldsymbol{\alpha}|$, $i \in \mathbb{N}$, is the kernel order (Gasser et al. [GMM85]), defined as the lowest order of the kernel moments greater ν for which at least one $\mu_{\boldsymbol{\gamma}:\,|\boldsymbol{\gamma}|=k}(K_{\boldsymbol{\alpha}})$ is non-zero. Note that the conditions (124) also imply

$$\mu_{\boldsymbol{\gamma}}(K_{\boldsymbol{\alpha}}) = 0\ \forall \boldsymbol{\gamma} : |\boldsymbol{\gamma}| < |\boldsymbol{\alpha}| \Rightarrow \mu_{\boldsymbol{\gamma}}(K_{\boldsymbol{\alpha}}) = 0\ \forall \boldsymbol{\gamma} : \boldsymbol{\gamma} \not\geq \boldsymbol{\alpha}, \tag{125}$$

since the integration in Equation (123) over the ith spatial dimension, corresponding to an index with $\gamma_i < \alpha_i$, will always yield zero, due to the conditions (124) implying $\mu_{\gamma_i \boldsymbol{\delta}_i} = 0$.

For $K_{\boldsymbol{\alpha}}$ satisfying the constraints (124), Equation (126) reads

$$\mathcal{M}\,\mathrm{E}[\eta] = \boldsymbol{\partial}^{\boldsymbol{\alpha}} F_g(\boldsymbol{x}) + \frac{(-1)^k}{\boldsymbol{h}^{\boldsymbol{\alpha}}} \sum_{\substack{|\boldsymbol{\gamma}|=k \\ \boldsymbol{\gamma} > \boldsymbol{\alpha}}} \frac{\boldsymbol{h}^{\boldsymbol{\gamma}}}{\boldsymbol{\gamma}!} \mu_{\boldsymbol{\gamma}}(K_{\boldsymbol{\alpha}})\, \boldsymbol{\partial}^{\boldsymbol{\gamma}} F(\boldsymbol{x}) + o\left\{ \sum_{\substack{|\boldsymbol{\gamma}|=k \\ \boldsymbol{\gamma} > \boldsymbol{\alpha}}} \boldsymbol{h}^{\boldsymbol{\gamma} - \boldsymbol{\alpha}} \right\}, \tag{126}$$

where the summation is over all multi indices $\boldsymbol{\gamma}$ of length $|\boldsymbol{\gamma}| = k$ satisfying $\boldsymbol{\gamma} > \boldsymbol{\alpha}$, and the "small oh" notation $o\{g(\boldsymbol{h})\}$ denotes all terms that approach zero faster than the argument g as $\boldsymbol{h} \to \mathbf{0}^+$.

20 Convergence of the Kernel Estimate

To study the convergence of the estimate (113) as a function of N as $N \to \infty$, the following expressions, defined point-wise, i.e., at every position $\boldsymbol{x}$, are studied:

$$\mathrm{B}\Big[\widehat{\boldsymbol{\partial}^{\boldsymbol{\alpha}} F_g}\Big] = \mathrm{E}\Big[\widehat{\boldsymbol{\partial}^{\boldsymbol{\alpha}} F_g}\Big] - \boldsymbol{\partial}^{\boldsymbol{\alpha}} F_g \qquad \text{(bias)}, \tag{127}$$

$$\mathrm{Var}\Big[\widehat{\boldsymbol{\partial}^{\boldsymbol{\alpha}} F_g}\Big] = \mathrm{E}\Big[\Big(\widehat{\boldsymbol{\partial}^{\boldsymbol{\alpha}} F_g} - \mathrm{E}\Big[\widehat{\boldsymbol{\partial}^{\boldsymbol{\alpha}} F_g}\Big]\Big)^2\Big] \qquad \text{(variance)}, \tag{128}$$

$$\text{and} \quad \mathrm{MSE}\Big[\widehat{\boldsymbol{\partial}^{\boldsymbol{\alpha}} F_g}\Big] = \mathrm{E}\Big[\Big(\widehat{\boldsymbol{\partial}^{\boldsymbol{\alpha}} F_g} - \boldsymbol{\partial}^{\boldsymbol{\alpha}} F_g\Big)^2\Big] \qquad \text{(mean squared error)}, \tag{129}$$

under the assumption that the kernel $K_{\boldsymbol{\alpha}}$ satisfies the constraints (124).

20.1 Bias

For a kernel of order k, satisfying Equations (124), using Equations (117) and (126), the point-wise bias of the estimate (113) reads

$$\mathrm{B}\Big[\widehat{\boldsymbol{\partial}^{\boldsymbol{\alpha}} F_g}(\boldsymbol{x})\Big] = \frac{(-1)^k}{\boldsymbol{h}^{\boldsymbol{\alpha}}} \sum_{\substack{|\boldsymbol{\gamma}|=k \\ \boldsymbol{\gamma}>\boldsymbol{\alpha}}} \frac{\boldsymbol{h}^{\boldsymbol{\gamma}}}{\boldsymbol{\gamma}!} \mu_{\boldsymbol{\gamma}}(K_{\boldsymbol{\alpha}})\, \boldsymbol{\partial}^{\boldsymbol{\gamma}} F(\boldsymbol{x}) + o\left\{ \sum_{\substack{|\boldsymbol{\gamma}|=k \\ \boldsymbol{\gamma}>\boldsymbol{\alpha}}} \boldsymbol{h}^{\boldsymbol{\gamma}-\boldsymbol{\alpha}} \right\}. \tag{130}$$

Equation (130) shows that the order of the kernel determines the leading order term in the expansion of the bias. The specific form of the leading bias term may be controlled by imposing additional conditions on the kth-order moments, e.g.,

$$\mu_{\boldsymbol{\gamma}:\,|\boldsymbol{\gamma}|=k}(K_{\boldsymbol{\alpha}}) \overset{!}{=} \begin{cases} \mu^{K_{\boldsymbol{\alpha}}} \boldsymbol{\gamma}!, & \boldsymbol{\gamma} = \boldsymbol{\alpha} + (k-\nu)\boldsymbol{\delta}_i,\ i \in \{1, 2, \ldots, d_x\} \\ 0 & \text{else}, \end{cases} \tag{131}$$

with scalar parameter $\mu^{K_{\boldsymbol{\alpha}}} \in \mathbb{R}$ (cf., for example, [Vas08] and references therein). This choice of $\mu_{\boldsymbol{\gamma}}$ ensures that the leading bias term only depends on the mixed partial derivatives $\boldsymbol{\partial}^{\boldsymbol{\gamma}} F_g$ with $\boldsymbol{\gamma} > \boldsymbol{\alpha}$, and equally weights the contribution of each coordinate direction. Other choices are of course possible: for example, the condition

$$\mu_{\boldsymbol{\gamma}:\,|\boldsymbol{\gamma}|=k}(K_{\boldsymbol{\alpha}}) \overset{!}{=} \begin{cases} \mu^{K_{\boldsymbol{\alpha}}} \boldsymbol{\gamma}! \frac{|\boldsymbol{\omega}|!}{\boldsymbol{\omega}!}, & \boldsymbol{\gamma} = \boldsymbol{\alpha} + 2\boldsymbol{\omega} \\ 0 & \text{else}, \end{cases} \tag{132}$$

with multi index $\boldsymbol{\omega}$, would ensure that the leading bias term is proportional to the total trace of the mixed derivative tensor of order $k-\nu$, $\frac{\partial^{k-\nu}}{\partial_{j_1}\partial_{j_2}\cdots\partial_{j_{(k-\nu)}}}$, $1 \le j_i \le d_x$, which, in contrast to the sum of elements with equal indices implied by the constraints (131), is a tensor invariant. Alternative choices to Equation (131) are not investigated in the following.

Under the additional assumption that the kernel $K_{\boldsymbol{\alpha}}$ satisfies the kth-order moment constraints (131), Equation (130) reads

$$\mathrm{B}\Big[\widehat{\boldsymbol{\partial}^{\boldsymbol{\alpha}} F_g}(\boldsymbol{x})\Big] = (-1)^k \mu^{K_{\boldsymbol{\alpha}}} \sum_{i=1}^{d_x} \boldsymbol{h}^{(k-\nu)\boldsymbol{\delta}_i}\, \boldsymbol{\partial}^{\boldsymbol{\alpha}+(\mathrm{k}-\nu)\boldsymbol{\delta}_\mathrm{i}} F_g(\boldsymbol{x}) + o\left\{ \sum_{\substack{|\boldsymbol{\gamma}|=k \\ \boldsymbol{\gamma}>\boldsymbol{\alpha}}} \boldsymbol{h}^{\boldsymbol{\gamma}-\boldsymbol{\alpha}} \right\}. \tag{133}$$

For example, inserting $\boldsymbol{\alpha} = \boldsymbol{\delta}_j$, $h_i \equiv h$, $d = 3$ and $k = 5$ into Equation (133) yields

$$\mathrm{B}\Big[\widehat{\partial_j F_g}(\boldsymbol{x})\Big] = -h^4 \mu^{K_{\boldsymbol{\alpha}}} \partial_j \Big(\partial_1^4 + \partial_2^4 + \partial_3^4\Big) F_g(\boldsymbol{x}) + o\Big\{h^4\Big\}. \tag{134}$$

20.2 Variance and Mean Squared Error

Applying the variance operator to Equation (116) and using Bienaymé's identity for a weighted sum of independent random variables yields

$$\mathrm{Var}\Big[\widehat{\boldsymbol{\partial}^{\boldsymbol{\alpha}} F_g}\Big] = \mathrm{Var}\left[\sum_{l=1}^{N} w^l \eta^l\right] = \Big(\mathrm{E}\Big[\eta^2\Big] - \mathrm{E}[\eta]^2\Big) \sum_{l=1}^{N} \Big(w^l\Big)^2. \tag{135}$$

Using the same change of variables as in Equation (121), one finds the intermediate result

$$\begin{aligned} \mathcal{M}\,\mathrm{E}\Big[\eta^2\Big] &= \frac{1}{\boldsymbol{h}^{2(\boldsymbol{\alpha}+\mathbf{1})}} \iint_{\mathbb{R}^{d_x}\times\mathbb{R}^{d_c}} K_{\boldsymbol{\alpha}}\Big((\mathrm{diag}\,\boldsymbol{h})^{-1}(\boldsymbol{x}-\boldsymbol{x}')\Big)^2 g(\boldsymbol{c}',\boldsymbol{x}')^2 f(\boldsymbol{x}',\boldsymbol{c}')\mathrm{d}\boldsymbol{x}'^{\mathbf{1}}\mathrm{d}\boldsymbol{c}'^{\mathbf{1}} \\ &= \frac{1}{\boldsymbol{h}^{2(\boldsymbol{\alpha}+\mathbf{1})}} \int_{\mathbb{R}^{d_x}} K_{\boldsymbol{\alpha}}\Big((\mathrm{diag}\,\boldsymbol{h})^{-1}(\boldsymbol{x}-\boldsymbol{x}')\Big)^2 F_{g^2}(\boldsymbol{x}')\mathrm{d}\boldsymbol{x}'^{\mathbf{1}} \\ &= \frac{1}{\boldsymbol{h}^{2\boldsymbol{\alpha}+\mathbf{1}}} \int_{\mathbb{R}^{d_x}} K_{\boldsymbol{\alpha}}(\boldsymbol{y})^2 F_{g^2}(\boldsymbol{x}-(\mathrm{diag}\,\boldsymbol{h})\boldsymbol{y})\mathrm{d}\boldsymbol{y}^{\mathbf{1}} \\ &= \frac{R(K_{\boldsymbol{\alpha}})F_{g^2}(\boldsymbol{x})}{\boldsymbol{h}^{2\boldsymbol{\alpha}+\mathbf{1}}} + o\{1\}, \end{aligned} \tag{136}$$

where

$$R(g) = \int_{\mathbb{R}^{d_x}} g(\boldsymbol{y})^2 \mathrm{d}\boldsymbol{y}^{\mathbf{1}} \tag{137}$$

is defined for any square integrable function $g\colon \mathbb{R}^{d_x} \to \mathbb{R}$. Inserting Equation (136) into Equation (135), the point-wise variance of the estimate (113) follows as

$$\begin{aligned} \mathrm{Var}\Big[\widehat{\boldsymbol{\partial}^{\boldsymbol{\alpha}} F_g}(\boldsymbol{x})\Big] &= \sum_{l=1}^{N} \Big(w^l\Big)^2 \Big(\mathrm{E}\Big[\eta^2\Big] - \mathrm{E}[\eta]^2\Big) \\ &= \frac{1}{\mathcal{M}} \sum_{l=1}^{N} \Big(w^l\Big)^2 \left(\frac{R(K_{\boldsymbol{\alpha}})F_{g^2}(\boldsymbol{x})}{\boldsymbol{h}^{2\boldsymbol{\alpha}+\mathbf{1}}} + o\{1\} - \frac{1}{\mathcal{M}} \big(\boldsymbol{\partial}^{\boldsymbol{\alpha}} F_g(\boldsymbol{x}) + o\{1\}\big)^2 \right) \\ &= \frac{\sum_{l=1}^{N} \Big(w^l\Big)^2}{\mathcal{M}\boldsymbol{h}^{2\boldsymbol{\alpha}+\mathbf{1}}} R(K_{\boldsymbol{\alpha}})F_{g^2}(\boldsymbol{x}) + o\left\{ \frac{1}{\boldsymbol{h}^{2\boldsymbol{\alpha}+\mathbf{1}}} \sum_{l=1}^{N} \Big(w^l\Big)^2 \right\}. \end{aligned} \tag{138}$$

By the assumptions (119) on $\boldsymbol{h}$, $\mathrm{Var}\Big[\widehat{\boldsymbol{\partial^{\alpha}} F_g}(\boldsymbol{x})\Big]$ approaches zero as $N \to \infty$. The fastest rate of convergence of $\frac{1}{\mathcal{M}} \sum_{l=1}^{N} \Big(w^l\Big)^2 \to 0$ is obtained in the case of equally weighted samples, i.e., with $w^l = w = \frac{\mathcal{M}}{N}$, $l = 1, \ldots, N$, which, in turn, implies $\frac{1}{\mathcal{M}} \sum_{l=1}^{N} w^2 = \frac{\mathcal{M}}{N}$.

The point-wise mean squared error (MSE) of the estimate (113) follows as

$$\begin{aligned}\mathrm{MSE}\Big[\widehat{\boldsymbol{\partial^{\alpha}} F_g}\Big] &= \mathrm{Var}\Big[\widehat{\boldsymbol{\partial^{\alpha}} F_g}\Big] + \mathrm{B}\Big[\widehat{\boldsymbol{\partial^{\alpha}} F_g}\Big]^2 \\ &= \frac{\sum_{l=1}^{N}\Big(w^l\Big)^2}{\mathcal{M}\boldsymbol{h}^{2\boldsymbol{\alpha}+\mathbf{1}}} R(K_{\boldsymbol{\alpha}}) F_{g^2}(\boldsymbol{x}) + \frac{1}{\boldsymbol{h}^{2\boldsymbol{\alpha}}} \left(\sum_{\substack{|\boldsymbol{\gamma}|=k \\ \boldsymbol{\gamma}>\boldsymbol{\alpha}}} \frac{\boldsymbol{h}^{\boldsymbol{\gamma}}}{\boldsymbol{\gamma}!} \mu_{\boldsymbol{\gamma}}(K_{\boldsymbol{\alpha}})\, \boldsymbol{\partial}^{\boldsymbol{\gamma}} F_g(\boldsymbol{x}) \right)^2 \\ &\qquad + o\left\{ \frac{1}{\boldsymbol{h}^{2\boldsymbol{\alpha}+\mathbf{1}}} \sum_{l=1}^{N} \Big(w^l\Big)^2 \right\}. \end{aligned} \tag{139}$$

21 Asymptotically Optimal Bandwidth

Under the additional assumptions that $h_i = h$, $i = 1, \ldots, d_x$, and $w^l = w = \frac{\mathcal{M}}{N}$, $l = 1, \ldots, N$, assuming $K_{\boldsymbol{\alpha}}$ satisfies conditions (131), the asymptotic values for the point-wise bias, variance and mean squared error, i.e., the leading terms in expressions (130), (138) and (139), read

$$\mathrm{AB}\Big[\widehat{\boldsymbol{\partial^{\alpha}} F_g}(\boldsymbol{x})\Big] = (-1)^k h^{(k-\nu)} \mu^{K_{\boldsymbol{\alpha}}} \sum_{i=1}^{d_x} \boldsymbol{\partial}^{\boldsymbol{\alpha}+(\mathrm{k}-\nu)\boldsymbol{\delta}_{\mathrm{i}}} F_g(\boldsymbol{x}), \tag{140}$$

$$\mathrm{AVar}\Big[\widehat{\boldsymbol{\partial^{\alpha}} F_g}(\boldsymbol{x})\Big] = \frac{1}{N h^{d_x+2\nu}} R(K_{\boldsymbol{\alpha}}) \mathcal{M} F_{g^2}(\boldsymbol{x}), \tag{141}$$

$$\begin{aligned}\text{and} \quad \mathrm{AMSE}\Big[\widehat{\boldsymbol{\partial^{\alpha}} F_g}(\boldsymbol{x})\Big] &= \frac{1}{N h^{d_x+2\nu}} R(K_{\boldsymbol{\alpha}}) \mathcal{M} F_{g^2}(\boldsymbol{x}) \\ &\qquad + h^{2(k-\nu)} \Big(\mu^{K_{\boldsymbol{\alpha}}}\Big)^2 \left(\sum_{i=1}^{d_x} \boldsymbol{\partial}^{\boldsymbol{\alpha}+(\mathrm{k}-\nu)\boldsymbol{\delta}_{\mathrm{i}}} F_g(\boldsymbol{x}) \right)^2, \end{aligned} \tag{142}$$

respectively. Since for any kernel $K_{\boldsymbol{\alpha}}$ satisfying conditions (124) and (131), the scaled kernel $K^{\delta}_{\boldsymbol{\alpha}} := \delta^{-(d_x+\nu)} K_{\boldsymbol{\alpha}}(\delta^{-1}\boldsymbol{x})$, with $\delta \in \mathbb{R} > 0$, also satisfies Equation (124), as well as Equation (131) with $\mu^{K^{\delta}_{\boldsymbol{\alpha}}} = \delta^{k-\nu} \mu^{K_{\boldsymbol{\alpha}}}$, one may choose a scaling $\delta = \delta_c$ such that $\Big(\mu^{K^{\delta_c}_{\boldsymbol{\alpha}}}\Big)^2 = R\Big(K^{\delta_c}_{\boldsymbol{\alpha}}\Big) =: T(K_{\boldsymbol{\alpha}})$, with R as defined in Equation (137), and such a kernel is referred to as canonical [WJ94]. Specifically, $R\Big(K^{\delta}_{\boldsymbol{\alpha}}\Big) = \delta^{-(2\nu+d_x)} R(K_{\boldsymbol{\alpha}})$, and hence $\delta_c = \left(\frac{R(K_{\boldsymbol{\alpha}})}{(\mu^{K_{\boldsymbol{\alpha}}})^2}\right)^{\frac{1}{2k+d_x}}$. It follows that

$$T(K_{\boldsymbol{\alpha}}) = \left[R(K_{\boldsymbol{\alpha}})^{k-\nu} \Big| \mu^{K_{\boldsymbol{\alpha}}} \Big|^{2\nu+d_x} \right]^{\frac{2}{2k+d_x}}. \tag{143}$$

Note that $T(K_{\boldsymbol{\alpha}})$ is invariant to any re-scaling of $K_{\boldsymbol{\alpha}}$, i.e., $T\Big(K_{\boldsymbol{\alpha}}^{\delta}\Big) \equiv T(K_{\boldsymbol{\alpha}})$. Equation (142) may be written in terms of $T(K_{\boldsymbol{\alpha}})$ as

$$\begin{aligned}\mathrm{AMSE}\Big[\widehat{\boldsymbol{\partial}^{\boldsymbol{\alpha}} F_g}(\boldsymbol{x})\Big] = T(K_{\boldsymbol{\alpha}})\Bigg(&\frac{1}{N h^{d_x+2\nu}} \mathcal{M} F_{g^2}(\boldsymbol{x}) \\ &+ h^{2(k-\nu)} \left(\sum_{i=1}^{d_x} \boldsymbol{\partial}^{\boldsymbol{\alpha}+(\mathrm{k}-\nu)\boldsymbol{\delta}_{\mathrm{i}}} F_g(\boldsymbol{x})\right)^2\Bigg).\end{aligned} \tag{144}$$

Assuming $\mathrm{AB}\Big[\widehat{\boldsymbol{\partial}^{\boldsymbol{\alpha}} F_g}(\boldsymbol{x})\Big] \neq 0$, differentiating Equation (144) w.r.t. h and equating to zero formally yields the point-wise AMSE-optimal bandwidth h_{AMSE}, i.e.,

$$h_{\mathrm{AMSE}}(\boldsymbol{x}, N; d_x, k, \nu, F_g, F_{g^2}) = \left[\frac{1}{N}\frac{(d_x+2\nu)}{2(k-\nu)}\frac{\mathcal{M} F_{g^2}(\boldsymbol{x})}{\Big(\sum_{i=1}^{d_x} \boldsymbol{\partial}^{\boldsymbol{\alpha}+(\mathrm{k}-\nu)\boldsymbol{\delta}_{\mathrm{i}}} F_g(\boldsymbol{x})\Big)^2}\right]^{\frac{1}{2k+d_x}}. \tag{145}$$

22 Kernel Estimate of Unweighted Generalized Moments

In case the density ρ is known at the sample locations $\boldsymbol{x}^l$, one may set $\tilde{g}(\boldsymbol{c}^l, \boldsymbol{x}^l) := g(\boldsymbol{c}^l, \boldsymbol{x}^l)\big/\rho(\boldsymbol{x}^l)$, in which case $\widehat{\boldsymbol{\partial}^{\boldsymbol{\alpha}} F_{\tilde{g}}}(\boldsymbol{x})$ (Equation (113)) yields a consistent estimate for the unweighted, generalized moment $\boldsymbol{\partial}^{\boldsymbol{\alpha}} \mathrm{E}_{\boldsymbol{\zeta}}[g(\boldsymbol{\zeta}; \boldsymbol{x})]$ directly. In the general case, an estimate of $\boldsymbol{\partial}^{\boldsymbol{\alpha}} \mathrm{E}_{\boldsymbol{\zeta}}[g(\boldsymbol{\zeta}; \boldsymbol{x})]$ may be obtained by applying the generalized Leibniz rule to expand the mixed derivative $\boldsymbol{\partial}^{\boldsymbol{\alpha}} F_g$, i.e.,

$$\boldsymbol{\partial}^{\boldsymbol{\alpha}} F_g(\boldsymbol{x}) = \boldsymbol{\partial}^{\boldsymbol{\alpha}}\, \mathrm{E}_{\boldsymbol{\zeta}}[\rho(\boldsymbol{x}) g(\boldsymbol{\zeta}; \boldsymbol{x})] = \boldsymbol{\alpha}! \sum_{\boldsymbol{\eta}+\boldsymbol{\omega}=\boldsymbol{\alpha}} \frac{\boldsymbol{\partial}^{\boldsymbol{\eta}} \rho(\boldsymbol{x})}{\boldsymbol{\eta}!} \frac{\boldsymbol{\partial}^{\boldsymbol{\omega}} \mathrm{E}_{\boldsymbol{\zeta}}[g(\boldsymbol{\zeta}; \boldsymbol{x})]}{\boldsymbol{\omega}!}, \tag{146}$$

where the summation is over all multi indices $\boldsymbol{\eta}, \boldsymbol{\omega}$ for which $\boldsymbol{\eta} + \boldsymbol{\omega} = \boldsymbol{\alpha}$. Rearranging Equation (146) yields

$$\boldsymbol{\partial}^{\boldsymbol{\alpha}} \mathrm{E}_{\boldsymbol{\zeta}}[g(\boldsymbol{\zeta}; \boldsymbol{x})] = \frac{1}{\rho(\boldsymbol{x})}\left(\boldsymbol{\partial}^{\boldsymbol{\alpha}} F_g(\boldsymbol{x}) - \boldsymbol{\alpha}! \sum_{\substack{\boldsymbol{\eta}+\boldsymbol{\omega}=\boldsymbol{\alpha} \\ \boldsymbol{\omega}\neq\boldsymbol{\alpha}}} \frac{\boldsymbol{\partial}^{\boldsymbol{\eta}} \rho(\boldsymbol{x})}{\boldsymbol{\eta}!} \frac{\boldsymbol{\partial}^{\boldsymbol{\omega}} \mathrm{E}_{\boldsymbol{\zeta}}[g(\boldsymbol{\zeta}; \boldsymbol{x})]}{\boldsymbol{\omega}!}\right), \tag{147}$$

where the unknown partial derivatives $\boldsymbol{\partial}^{\boldsymbol{\eta}} \rho$ on the right hand side may be estimated by setting $g \equiv 1$ in Equation (113) (since $F_1(\boldsymbol{x}) \equiv \rho(\boldsymbol{x})$), and the unknown partial derivatives $\boldsymbol{\partial}^{\boldsymbol{\omega}} E_{\zeta}[g(\boldsymbol{\zeta}; \boldsymbol{x})]$ with $\boldsymbol{\omega} \neq \boldsymbol{\alpha}$, all of which of order smaller $|\boldsymbol{\alpha}|$, may be estimated by applying the procedure recursively. Obviously, the case $\boldsymbol{\alpha} = \mathbf{0}$ leads to $\mathrm{E}_{\zeta}[g(\boldsymbol{\zeta}; \boldsymbol{x})] = F_g(\boldsymbol{x})/F_1(\boldsymbol{x})$.

23 Kernel Interpolation

It is worth noting that kernel-based interpolation of a (deterministic but potentially unknown) function $u\colon \boldsymbol{x} \in \mathbb{R}^{d_x} \to \mathbb{R}$ is directly related to the estimate $\widehat{\boldsymbol{\partial}^{\boldsymbol{\alpha}} F_g}(\boldsymbol{x})$ with $g(\boldsymbol{c}, \boldsymbol{x}) \equiv g(\boldsymbol{x})$, since

$$\boldsymbol{\partial}^{\boldsymbol{\alpha}} F_g(\boldsymbol{x}) = \boldsymbol{\partial}^{\boldsymbol{\alpha}} \operatorname{E}_{\boldsymbol{\zeta}}[\rho(\boldsymbol{x}) g(\boldsymbol{x})] \equiv \boldsymbol{\partial}^{\boldsymbol{\alpha}} (\rho(\boldsymbol{x}) g(\boldsymbol{x})). \tag{148}$$

Given as data N values $u^l := u(\boldsymbol{x}^l)$ at (random) locations $\boldsymbol{x}^l$, $l = 1, \ldots, N$, one may directly use the data u^l in Equation (113) by setting $g(\boldsymbol{x}^l) := u^l$. By the results obtained in Section 19, $\widehat{\boldsymbol{\partial}^{\boldsymbol{\alpha}} F_g}(\boldsymbol{x})$ then yields a consistent estimate for $\boldsymbol{\partial}^{\boldsymbol{\alpha}}(\rho(\boldsymbol{x}) u(\boldsymbol{x}))$. The quantity $\boldsymbol{\partial}^{\boldsymbol{\alpha}} u(\boldsymbol{x})$ may, in turn, be estimated via the procedure outlined in Section 22.

24 Explicit Kernel Functions

To derive explicit expressions for $K_{\boldsymbol{\alpha}}$, again, only the case $h_i = h$, $i = 1, \ldots, d_x$ is considered. Following Gasser et al. [GMM85], two classes of kernels are investigated: "minimum variance kernels" $K^v_{\boldsymbol{\alpha}}$, that minimize the asymptotic variance (Equation (141)), and "optimal kernels" $K^o_{\boldsymbol{\alpha}}$, minimizing the asymptotic MSE (Equation (142)). In the following, only continuous kernels with compact support on the d_x-dimensional interval $[-1, 1]^{d_x}$ will be considered.

24.1 Minimum Variance Kernels

For a given configuration space dimension d_x, mixed partial derivative operator $\boldsymbol{\partial}^{\boldsymbol{\alpha}}$ of order $|\boldsymbol{\alpha}| = \nu$, and kernel order $k > \nu$, the goal is to find the kernel $K^{v,k}_{\boldsymbol{\alpha}}$ that minimizes the functional $R\big(K^k_{\boldsymbol{\alpha}}\big) = \int K^k_{\boldsymbol{\alpha}}(\boldsymbol{x})^2 \mathrm{d}\boldsymbol{x}^{\mathbf{1}}$ (cf. Equation (137)) on the d_x-dimensional interval $[-1, 1]^{d_x}$, subject to the moment conditions (124), and no condition on the values of the kth-order moments. Using the method of Lagrangian multipliers, this may be stated as the following minimization problem over the square-integrable functions $u : \mathbb{R}^{d_x} \to \mathbb{R}$:

$$\begin{aligned} K^{v,k}_{\boldsymbol{\alpha}}(\boldsymbol{x}) &= \operatorname*{argmin}_u \int_{-1}^{1} \left(u(\boldsymbol{x})^2 + u(\boldsymbol{x}) \sum_{|\boldsymbol{\omega}|<k} \lambda_{\boldsymbol{\omega}} \boldsymbol{x}^{\boldsymbol{\omega}} \right) \mathrm{d}\boldsymbol{x}^{\mathbf{1}} \\ &= \operatorname*{argmin}_u \int_{-1}^{1} L(\boldsymbol{x}, u(\boldsymbol{x})) \mathrm{d}\boldsymbol{x}^{\mathbf{1}}, \end{aligned} \tag{149}$$

where the summation is over all multi indices $\boldsymbol{\omega}$ with length $|\boldsymbol{\omega}| < k$, and the Lagrange multipliers $\lambda_{\boldsymbol{\omega}} \in \mathbb{R}$ are indexed by the respective multi index $\boldsymbol{\omega}$. The solution of the associated (trivial) Euler-Lagrange equation $\frac{\partial L}{\partial u} = 0$ is

$$K^{v,k}_{\boldsymbol{\alpha}}(\boldsymbol{x}) = \sum_{|\boldsymbol{\omega}|<k} \lambda^\star_{\boldsymbol{\omega}} \boldsymbol{x}^{\boldsymbol{\omega}}, \tag{150}$$

where the coefficients $\lambda_{\boldsymbol{\omega}}^{\star} \in \mathbb{R}$ are determined by the solution of the linear system (124).

Explicit expressions for minimum variance kernels of orders $k = \{\nu + 2, \nu + 4\}$ in $d_x = \{1, 2, 3\}$ for the estimation of mixed partial derivatives of orders $\nu = \{0, 1, 2, 3, 4\}$ are given in Appendix D.1.

24.2 Optimal Kernels

For a given configuration space dimension d_x, mixed partial derivative operator $\boldsymbol{\partial}^{\boldsymbol{\alpha}}$ of order $|\boldsymbol{\alpha}| = \nu$, and kernel order $k > \nu$, the goal is to find the kernel $K_{\boldsymbol{\alpha}}^{v,\,k}$ that minimizes the functional $T\Big(K_{\boldsymbol{\alpha}}^{k}\Big)$ (cf. Equation (143)) on the d_x-dimensional interval $[-1, 1]^{d_x}$, subject to the moment conditions (124) and (131). The corresponding variational problem is equivalent to the following minimization problem over square-integrable functions $u : \mathbb{R}^{d_x} \to \mathbb{R}$:

$$\begin{aligned} K_{\boldsymbol{\alpha}}^{o,\,k} = \underset{u}{\operatorname{argmin}} \min_{I_1, I_2} \max_{\lambda_1, \lambda_2} \Bigg\{ & I_1^a I_2^b + \lambda_1 \Bigg(I_1 - \int_{-1}^{1} u(\boldsymbol{x})^2 \mathrm{d}\boldsymbol{x}^{\mathbf{1}} \Bigg) \\ & + \lambda_2 \Bigg(I_2 - \int_{-1}^{1} u(\boldsymbol{x}) \sum_{|\boldsymbol{\omega}|=k} \boldsymbol{x}^{\boldsymbol{\omega}} \mathrm{d}\boldsymbol{x}^{\mathbf{1}} \Bigg) \Bigg\}, \end{aligned} \tag{151}$$

subject to conditions (124) and (131), where the constants a and b are shorthand for $k - \nu$ and $2\nu - d$, respectively, and $I_1, I_2, \lambda_1, \lambda_2 \in \mathbb{R}$. Carrying out the minimization w.r.t. I_1, I_2 to determine λ_1 and λ_2, and using Lagrange multipliers $\lambda_{\boldsymbol{\omega}} \in \mathbb{R}$, indexed by the multi index $\boldsymbol{\omega}$, to incorporate the constraints (124) and (131), problem (151) reduces to

$$\begin{aligned} K_{\boldsymbol{\alpha}}^{o,\,k} &= \underset{u}{\operatorname{argmin}} \int_{-1}^{1} \Bigg(a I_1^{a-1} I_2^b u(\boldsymbol{x})^2 + b I_1^a I_2^{b-1} u(\boldsymbol{x}) \sum_{|\boldsymbol{\omega}|=k} \boldsymbol{x}^{\boldsymbol{\omega}} \\ &\qquad + u(\boldsymbol{x}) \sum_{|\boldsymbol{\omega}|\leq k} \lambda_{\boldsymbol{\omega}} \boldsymbol{x}^{\boldsymbol{\omega}} \Bigg) \mathrm{d}\boldsymbol{x}^{\mathbf{1}} \\ &= \underset{u}{\operatorname{argmin}} \int_{-1}^{1} \Bigg(u(\boldsymbol{x})^2 + u(\boldsymbol{x}) \sum_{|\boldsymbol{\omega}|\leq k} \lambda_{\boldsymbol{\omega}}^{\star} \boldsymbol{x}^{\boldsymbol{\omega}} \Bigg) \mathrm{d}\boldsymbol{x}^{\mathbf{1}}, \end{aligned} \tag{152}$$

where the coefficients $\lambda_{\boldsymbol{\omega}}^{\star} \in \mathbb{R}$ are given by

$$\lambda_{\boldsymbol{\omega}}^{\star} = (a I_1^{a-1} I_2^b)^{-1} \begin{cases} \lambda_{\boldsymbol{\omega}}, & |\boldsymbol{\omega}| < k \\ \Big(\lambda_{\boldsymbol{\omega}} - b I_1^a I_2^{b-1} \Big) & |\boldsymbol{\omega}| = k, \end{cases} \tag{153}$$

and I_1, I_2 are fixed. The solution of the associated Euler-Lagrange equation is

$$K_{\boldsymbol{\alpha}}^{o,\,k} = \sum_{|\boldsymbol{\omega}|\leq k} \lambda_{\boldsymbol{\omega}}^{\star\star} \boldsymbol{x}^{\boldsymbol{\omega}}, \tag{154}$$

where the coefficients $\lambda_{\boldsymbol{\omega}}^{\star\star} \in \mathbb{R}$ are determined by the solution of the linear system implied by conditions (124) and (131). However, $\mu^{K_{\boldsymbol{\alpha}}}$ remains as a free parameter, which means that problem (151) is actually degenerate, since the functional T may be made arbitrarily small by choosing $\mu^{K_{\boldsymbol{\alpha}}}$ arbitrarily close to zero. In order to regularize problem (151), two choices of additional constraints are considered: let optimal kernels be designated as Type I, if $\mu^{K_{\boldsymbol{\alpha}}}$ is chosen according to

$$\mu^{K_{\boldsymbol{\alpha}}^I} = \underset{\mu^{K_{\boldsymbol{\alpha}}}}{\operatorname{argmin}} \int_B \Big(K_{\boldsymbol{\alpha}}^{o,k}\Big(\boldsymbol{x}; \mu^{K_{\boldsymbol{\alpha}}}\Big)\Big)^2 \mathrm{d}\boldsymbol{x}^{\mathbf{1}}, \tag{155}$$

and as Type II, if $\mu^{K_{\boldsymbol{\alpha}}}$ is chosen according to

$$\mu^{K_{\boldsymbol{\alpha}}^{II}} = \underset{\mu^{K_{\boldsymbol{\alpha}}}}{\operatorname{argmin}} \int_{\partial B} \Big(K_{\boldsymbol{\alpha}}^{o,k}\Big(\boldsymbol{x}; \mu^{K_{\boldsymbol{\alpha}}}\Big)\Big)^2 \mathrm{d}\boldsymbol{x}^{\mathbf{1}}, \tag{156}$$

where B denotes the region of support of K, i.e., $[-1,1]^{d_x}$, and ∂B its boundary. In $d_x = 1$, kernels of Type I are equivalent to minimum variance kernels, and kernels of Type II coincide with the "minimal kernels" defined and derived by Gasser et al. [GMM85].

Explicit expressions for kernels of Type I and II of orders $k = \{\nu+2, \nu+4\}$ in $d_x = \{1,2,3\}$ for the estimation of mixed partial derivatives of orders $\nu = \{0,1,2,3,4\}$ are given in Appendices D.2 and D.3, respectively.

24.3 Explicit Coefficient Expressions

Both the minimum variance and the "optimal" kernels are polynomials, with coefficients determined by fixing their moments as defined by Equation (123). To derive explicit expressions for the coefficients, an expression for a general polynomial $K\colon \mathbb{R}^{d_x} \to \mathbb{R}$ of degree l in terms of its moments is required (similar to the procedure in Gasser et al. [GMM85] for $d_x = 1$):

Let $P_n\colon [-1,1] \to \mathbb{R}$, $n = 0,1,2,\ldots$, be a polynomial sequence satisfying the orthogonality property

$$\int_{-1}^{1} P_m(x) P_n(x) \mathrm{d}x = p_m \delta_{mn}. \tag{157}$$

Define $P_{\boldsymbol{\omega}}(\boldsymbol{x}) = P_{\omega_1}(x_1) P_{\omega_2}(x_2) \cdots P_{\omega_{d_x}}(x_{d_x}) : [-1,1]^{d_x} \to \mathbb{R}$, and, correspondingly, $p_{\boldsymbol{\omega}} = p_{\omega_1} p_{\omega_2} \cdots p_{\omega_{d_x}}$. If P_n has coefficients c_{kn}, that is, $P_n(x) = \sum_{k=0}^{n} c_{kn} x^k$, then

$$P_{\boldsymbol{\omega}}(\boldsymbol{x}) = \sum_{|\boldsymbol{\gamma}| \le |\boldsymbol{\omega}|} c_{\boldsymbol{\gamma}\boldsymbol{\omega}} \boldsymbol{x}^{\boldsymbol{\gamma}}, \tag{158}$$

where $c_{\boldsymbol{\gamma}\boldsymbol{\omega}} = c_{\gamma_1\omega_1} c_{\gamma_2\omega_2} \cdots c_{\gamma_{d_x}\omega_{d_x}}$, and the summation is over all multi indices $\boldsymbol{\gamma}$ of length $|\boldsymbol{\gamma}| \le |\boldsymbol{\omega}|$. The multivariate extension of Equation (157) retains its simple form in the multi-index notation, viz.

$$\int_{-1}^{1} P_{\boldsymbol{\omega}}(\boldsymbol{x}) P_{\boldsymbol{\eta}}(\boldsymbol{x}) \mathrm{d}\boldsymbol{x}^{\mathbf{1}} = p_{\boldsymbol{\omega}} \delta_{\boldsymbol{\omega}\boldsymbol{\eta}}, \tag{159}$$

where $\delta_{\boldsymbol{\omega\eta}} = \delta_{\omega_1\eta_1}\delta_{\omega_2\eta_2}\cdots\delta_{\omega_{d_x}\eta_{d_x}}$. A natural choice for the P_n are the Legendre polynomials, in which case

$$\begin{aligned} c_{kn,\mathrm{L}} &= 2^n \binom{n}{k}\binom{\frac{n+k-1}{2}}{n}, \qquad & p_{n,\mathrm{L}} &= \frac{2}{2n+1} \\ \text{and} \quad c_{\boldsymbol{\gamma\omega},\mathrm{L}} &= 2^{|\boldsymbol{\omega}|}\binom{\boldsymbol{\omega}}{\boldsymbol{\gamma}}\binom{\frac{\boldsymbol{\omega+\gamma-1}}{2}}{\boldsymbol{\omega}}, \qquad & p_{\boldsymbol{\omega},\mathrm{L}} &= 2^{d_x}\frac{(2\boldsymbol{\omega})!}{(2\boldsymbol{\omega}+1)!}, \end{aligned} \tag{160}$$

respectively, where

$$\begin{aligned} \binom{\boldsymbol{\omega}}{\boldsymbol{\gamma}} &= \binom{\omega_1}{\gamma_1}\binom{\omega_2}{\gamma_2}\cdots\binom{\omega_{d_x}}{\gamma_{d_x}}, \\ \binom{\frac{\boldsymbol{\omega+\gamma-1}}{2}}{\boldsymbol{\omega}} &= \binom{\frac{\omega_1+\gamma_1-1}{2}}{\omega_1}\binom{\frac{\omega_2+\gamma_2-1}{2}}{\omega_2}\cdots\binom{\frac{\omega_{d_x}+\gamma_{d_x}-1}{2}}{\omega_{d_x}} \\ \text{and} \quad \binom{a}{k} &= \frac{a(a-1)(a-2)\cdots(a-k+1)}{k!}, \quad a\in\mathbb{R},\ k\in\mathbb{N}. \end{aligned} \tag{161}$$

The Legendre polynomials further satisfy

$$P_n(-1) = (-1)^n \quad \text{and} \quad P_n(1) = 1. \tag{162}$$

Since the polynomials $P_{\boldsymbol{\omega}}$ form a basis in the space of polynomials of degree $|\boldsymbol{\omega}|$ with support $[-1,1]^{d_x}$, any polynomial $K:[-1,1]^{d_x}\to\mathbb{R}$ of degree l with coefficients $\lambda_{\boldsymbol{\eta}}$ may be expressed in terms of $P_{\boldsymbol{\omega}}$ as

$$K(\boldsymbol{x}) = \sum_{|\boldsymbol{\eta}|\le l}\lambda_{\boldsymbol{\eta}}\boldsymbol{x}^{\boldsymbol{\eta}} = \sum_{|\boldsymbol{\eta}|\le l}\lambda^*_{\boldsymbol{\eta}}P_{\boldsymbol{\eta}}(\boldsymbol{x}), \tag{163}$$

with new coefficients $\lambda^*_\eta\in\mathbb{R}$. Recall Equation (123), the definition of the moments $\mu_\gamma: K\to\int_{-1}^1\boldsymbol{x}^\gamma K(\boldsymbol{x})\mathrm{d}\boldsymbol{x}^{\mathbf{1}}$. With $P_{\boldsymbol{\omega}}$ as in Equation (158), one finds

$$\int_{-1}^1 P_{\boldsymbol{\omega}}(\boldsymbol{x})K(\boldsymbol{x})\mathrm{d}\boldsymbol{x}^{\mathbf{1}} = \int_{-1}^1\sum_{|\boldsymbol{\gamma}|\le|\boldsymbol{\omega}|}c_{\boldsymbol{\gamma\omega}}\boldsymbol{x}^{\boldsymbol{\gamma}}K(\boldsymbol{x})\mathrm{d}\boldsymbol{x}^{\mathbf{1}} = \sum_{|\boldsymbol{\gamma}|\le|\boldsymbol{\omega}|}c_{\boldsymbol{\gamma\omega}}\mu_{\boldsymbol{\gamma}}(K), \tag{164}$$

and for K given by Equation (163), using Equation (159), one obtains

$$\int_{-1}^1 P_{\boldsymbol{\omega}}(\boldsymbol{x})K(\boldsymbol{x})\mathrm{d}\boldsymbol{x}^{\mathbf{1}} = \int_{-1}^1 P_{\boldsymbol{\omega}}(\boldsymbol{x})\sum_{|\boldsymbol{\eta}|\le l}\lambda^*_{\boldsymbol{\eta}}P_{\boldsymbol{\eta}}(\boldsymbol{x})\mathrm{d}\boldsymbol{x}^{\mathbf{1}} = \lambda^*_{\boldsymbol{\omega}}p_{\boldsymbol{\omega}}. \tag{165}$$

Comparing Equations (164) and (165), and inserting fixed values $\mu^\star_{\boldsymbol{\gamma}}$ for the moments $\mu_{\boldsymbol{\gamma}}(K)$, determines the unique set of coefficients $\lambda^*_{\boldsymbol{\omega}} = \lambda^\star_{\boldsymbol{\omega}}$ for which $\mu_{\boldsymbol{\gamma}}(K(\boldsymbol{x};\lambda^\star_{\boldsymbol{\omega}})) = \mu^\star_{\boldsymbol{\gamma}}$:

$$\lambda^\star_{\boldsymbol{\omega}} = \frac{1}{p_{\boldsymbol{\omega}}}\sum_{|\boldsymbol{\gamma}|\le|\boldsymbol{\omega}|}c_{\boldsymbol{\gamma\omega}}\mu^\star_{\boldsymbol{\gamma}}. \tag{166}$$

Combining Equations (163) and (166), the explicit expression for the polynomial $K(\boldsymbol{x})$ of degree l, in terms of the given moment conditions $\mu^\star_\gamma$ and the basis polynomial coefficients $c_{\gamma\omega}$, is

$$K(\boldsymbol{x}) = \sum_{|\omega|\leq l} \frac{1}{p_\omega} \left(\sum_{|\gamma|\leq|\omega|} c_{\gamma\omega}\mu^\star_\gamma \right) P_\omega(\boldsymbol{x}). \tag{167}$$

Recall Equation (137), the definition of the functional $R\colon K \to \int_{-1}^{1} K(\boldsymbol{x})^2 \mathrm{d}\boldsymbol{x}^{\mathbf{1}}$. Using Equation (159), one finds

$$R(K) = \sum_{|\omega|\leq l} p_\omega (\lambda^\star_\omega)^2 = \sum_{|\omega|\leq l} \frac{1}{p_\omega} \left(\sum_{|\gamma|\leq|\omega|} c_{\gamma\omega}\mu^\star_\gamma \right)^2. \tag{168}$$

24.3.1 Coefficients for Minimum Variance Kernels

For the minimum variance kernels of order k (Equation (150)), inserting the conditions (124) into Equations (167) and (168) yields

$$K^{v,\,k}_{\boldsymbol{\alpha}}(\boldsymbol{x}) = (-1)^\nu \boldsymbol{\alpha}! \sum_{\nu\leq|\omega|<k} \frac{c_{\alpha\omega}}{p_\omega} P_\omega(\boldsymbol{x}) \tag{169}$$

and

$$R\Big(K^{v,\,k}_{\boldsymbol{\alpha}}\Big) = (\boldsymbol{\alpha}!)^2 \sum_{\nu\leq|\omega|<k} \frac{c^2_{\alpha\omega}}{p_\omega}, \tag{170}$$

respectively.

24.3.2 Coefficients for Optimal Kernels

For notational convenience, define

$$b^k_{\alpha\omega} = \sum_{i=1}^{d_x} c_{(\alpha+(k-\nu)\boldsymbol{\delta}_i)\omega} (\boldsymbol{\alpha} + (k-\nu)\boldsymbol{\delta}_i)!. \tag{171}$$

For the "optimal" kernels of order k (cf. Equation (154)), subject to conditions (124) and (131), Equations (167) and (168) read

$$K^{o,\,k}_{\boldsymbol{\alpha}}(\boldsymbol{x}) = (-1)^\nu \boldsymbol{\alpha}! \sum_{\nu\leq|\omega|\leq k} \frac{c_{\alpha\omega}}{p_\omega} P_\omega(\boldsymbol{x}) + \mu^{K_\alpha} \sum_{|\omega|=k} \frac{b^k_{\alpha\omega}}{p_\omega} P_\omega(\boldsymbol{x}) \tag{172}$$

and

$$\begin{aligned} R\Big(K^{o,\,k}_{\boldsymbol{\alpha}}\Big) = (\boldsymbol{\alpha}!)^2 \sum_{\nu\leq|\omega|\leq k} \frac{c^2_{\alpha\omega}}{p_\omega} + 2(-1)^\nu \boldsymbol{\alpha}! \mu^{K_\alpha} \sum_{|\omega|=k} \frac{c_{\alpha\omega} b^k_{\alpha\omega}}{p_\omega} \\ + \Big(\mu^{K_\alpha}\Big)^2 \sum_{|\omega|=k} \frac{\Big(b^k_{\alpha\omega}\Big)^2}{p_\omega}, \end{aligned} \tag{173}$$

respectively. Differentiating Equation (173) w.r.t. $\mu^{K_{\boldsymbol{\alpha}}}$ and equating to zero yields the value $\mu^{K_{\boldsymbol{\alpha}}^I}$ for "optimal" kernels of Type I according to Equation (155), i.e.,

$$\mu^{K_{\boldsymbol{\alpha}}^I} = (-1)^{\nu+1}\boldsymbol{\alpha}!\left(\sum_{|\boldsymbol{\omega}|=k}\frac{\left(b_{\boldsymbol{\alpha}\boldsymbol{\omega}}^k\right)^2}{p_{\boldsymbol{\omega}}}\right)^{-1}\sum_{|\boldsymbol{\omega}|=k}\frac{c_{\boldsymbol{\alpha}\boldsymbol{\omega}}b_{\boldsymbol{\alpha}\boldsymbol{\omega}}^k}{p_{\boldsymbol{\omega}}}. \tag{174}$$

To derive the value $\mu^{K_{\boldsymbol{\alpha}}^{II}}$ for "optimal" kernels of Type II according to Equation (156), it will be assumed in the following that the basis polynomials satisfy Equation (162). Let ∂B be the surface of the d_x-dimensional cube with edge length 2, centered at the origin, i.e., the boundary of the region $[-1,1]^{d_x}$. It follows that

$$\int_{\partial B} P_{\boldsymbol{\omega}}(\boldsymbol{x})P_{\boldsymbol{\eta}}(\boldsymbol{x})\mathrm{d}\boldsymbol{x}^{\mathbf{1}} = \sum_{i=1}^{d_x}\frac{p_{\boldsymbol{\omega}}}{p_{\omega_i}}\Big(1+(-1)^{\omega_i+\eta_i}\Big)\delta_{(\boldsymbol{\omega}-\omega_i\boldsymbol{\delta}_i)(\boldsymbol{\eta}-\eta_i\boldsymbol{\delta}_i)}, \tag{175}$$

where $\delta_{(\boldsymbol{\omega}-\omega_i\boldsymbol{\delta}_i)(\boldsymbol{\eta}-\eta_i\boldsymbol{\delta}_i)} \equiv \delta_{\omega_1\eta_1}\delta_{\omega_2\eta_2}\cdots\delta_{\omega_{i-1}\eta_{i-1}}\delta_{\omega_{i+1}\eta_{i+1}}\cdots\delta_{\omega_{d_x}\eta_{d_x}}$. Accordingly, for K as in Equation (163),

$$\int_{\partial B} K(\boldsymbol{x})^2\mathrm{d}\boldsymbol{x}^{\mathbf{1}} = \sum_{i=1}^{d_x}\sum_{|\boldsymbol{\omega}|\le k}\left\{\lambda_{\boldsymbol{\omega}}^\star\frac{p_{\boldsymbol{\omega}}}{p_{\omega_i}}\sum_{j=0}^{k-|\boldsymbol{\omega}|+\omega_i}\lambda_{\boldsymbol{\omega}+(j-\omega_i)\delta_i}^\star\Big(1+(-1)^{\omega_i+j}\Big)\right\}. \tag{176}$$

Inserting Equation (166), subject to conditions (124) and (131), into Equation (176) and expanding terms, yields

$$\begin{aligned}
&\int_{\partial B} K(\boldsymbol{x})^2\mathrm{d}\boldsymbol{x}^{\mathbf{1}} = \\
&\sum_{i=1}^{d_x}\Bigg[(-1)^\nu\boldsymbol{\alpha}!\sum_{\nu\le|\boldsymbol{\omega}|\le k}\Bigg\{(-1)^\nu\boldsymbol{\alpha}!\sum_{\substack{j=\max(\\ \nu-|\boldsymbol{\omega}|+\omega_i,0)}}^{k-|\boldsymbol{\omega}|+\omega_i}\frac{c_{\boldsymbol{\alpha}\boldsymbol{\omega}}c_{\boldsymbol{\alpha}(\boldsymbol{\omega}+(j-\omega_i)\boldsymbol{\delta}_i)}}{p_{\boldsymbol{\omega}}p_j}\Big(1+(-1)^{\omega_i+j}\Big) \\
&\qquad + \mu^{K_{\boldsymbol{\alpha}}}\frac{c_{\boldsymbol{\alpha}\boldsymbol{\omega}}b_{\boldsymbol{\alpha}(\boldsymbol{\omega}+(k-|\boldsymbol{\omega}|)\boldsymbol{\delta}_i)}^k}{p_{\boldsymbol{\omega}}p_{k-|\boldsymbol{\omega}|+\omega_i}}\Big(1+(-1)^{2\omega_i+k-|\boldsymbol{\omega}|}\Big)\Bigg\} \\
&\quad + \sum_{|\boldsymbol{\omega}|=k}\Bigg\{(-1)^\nu\boldsymbol{\alpha}!\mu^{K_{\boldsymbol{\alpha}}}\sum_{\substack{j=\max(\\ \nu-k+\omega_i,0)}}^{\omega_i}\frac{b_{\boldsymbol{\alpha}\boldsymbol{\omega}}^k c_{\boldsymbol{\alpha}(\boldsymbol{\omega}+(j-\omega_i)\boldsymbol{\delta}_i)}}{p_{\boldsymbol{\omega}}p_j}\Big(1+(-1)^{\omega_i+j}\Big) \\
&\qquad\qquad +2\Big(\mu^{K_{\boldsymbol{\alpha}}}\Big)^2\frac{\left(b_{\boldsymbol{\alpha}\boldsymbol{\omega}}^k\right)^2}{p_{\boldsymbol{\omega}}p_{\omega_i}}\Bigg\}\Bigg].
\end{aligned} \tag{177}$$

Equation (177) is minimized by

$$\mu^{K_{\boldsymbol{\alpha}}^{II}} = \left(4 \sum_{i=1}^{d_x} \sum_{|\boldsymbol{\omega}|=k} \frac{\left(b_{\boldsymbol{\alpha}\boldsymbol{\omega}}^{k}\right)^2}{p_{\boldsymbol{\omega}} p_{\omega_i}} \right)^{-1} (-1)^{\nu+1} \boldsymbol{\alpha}! \sum_{i=1}^{d_x} \Bigg\{ \sum_{\nu \leq |\boldsymbol{\omega}| \leq k} \frac{c_{\boldsymbol{\alpha}\boldsymbol{\omega}} b_{\boldsymbol{\alpha}(\boldsymbol{\omega}+(k-|\boldsymbol{\omega}|)\boldsymbol{\delta}_i)}^{k}}{p_{\boldsymbol{\omega}} p_{k-|\boldsymbol{\omega}|+\omega_i}} \left(1 + (-1)^{2\omega_i + k - |\boldsymbol{\omega}|}\right) + \sum_{|\boldsymbol{\omega}|=k} \sum_{\substack{j=\max(\\ \nu-k+\omega_i,0)}}^{\omega_i} \frac{b_{\boldsymbol{\alpha}\boldsymbol{\omega}}^{k} c_{\boldsymbol{\alpha}(\boldsymbol{\omega}+(j-\omega_i)\boldsymbol{\delta}_i)}}{p_{\boldsymbol{\omega}} p_j} \left(1 + (-1)^{\omega_i + j}\right) \Bigg\}. \tag{178}$$

25 Numerical Study: Sample from Maxwellian with Quadratic Spatial Parameter Variation

Define the state space moments of $f(\boldsymbol{x}, \boldsymbol{c})$ as

$$F_{\boldsymbol{\beta}}^{s}(\boldsymbol{x}) = \int_{\mathbb{R}^{d_c}} c'^{2s} \boldsymbol{c}'^{\boldsymbol{\beta}} f(\boldsymbol{x}, \boldsymbol{c}') \mathrm{d}\boldsymbol{c}'^{\mathbf{1}}, \tag{179}$$

where $c^{2s} = \left(\sum_{k=1}^{d_c} c_k^2\right)^s = \|\boldsymbol{c}\|^{2s}$, $\boldsymbol{\beta}$ is a multi index, and $\mathrm{d}\boldsymbol{c}'^{\mathbf{1}}$ denotes the differential volume element of state space $\mathrm{d}c'_1 \mathrm{d}c'_2 \cdots \mathrm{d}c'_{d_c}$. Equation (179) corresponds to Equation (109), the definition of $F_g(\boldsymbol{x})$, with $g(\boldsymbol{c}) := c^{2s} \boldsymbol{c}^{\boldsymbol{\beta}}$.

To assess the expected quality of the moment derivative estimates obtainable by the presented method, M samples of size N are generated from a space-dependent Maxwellian phase density defined on the domain $B = [-L, L]^{d_x}$, $L > 0$, i.e., samples with position vector $\boldsymbol{x} \in B$ and state vector $\boldsymbol{c} \in \mathbb{R}^{d_c}$ are generated with probability

$$P(\boldsymbol{x}, \boldsymbol{c}) \sim \frac{\rho(\boldsymbol{x})}{\theta(\boldsymbol{x})^{\frac{d_c}{2}}} \exp\left(-\frac{\|\boldsymbol{c} - \boldsymbol{v}(\boldsymbol{x})\|^2}{2\theta(\boldsymbol{x})} \right). \tag{180}$$

For the spatial dependence of the quantities $\rho, \theta \colon \mathbb{R}^{d_x} \to \mathbb{R}^+$ and $\boldsymbol{v} \colon \mathbb{R}^{d_x} \to \mathbb{R}^{d_c}$, the following polynomial forms are chosen:

$$\begin{aligned} \rho(\boldsymbol{x}) &= \rho_0 \left(1 + \frac{1}{L} \sum_k \alpha_k^{\rho} x_k + \frac{1}{2L^2} \sum_k \sum_l \beta_{kl}^{\rho} x_k x_l \right), \\ \theta(\boldsymbol{x}) &= \theta_0 \left(1 + \frac{1}{L} \sum_k \alpha_k^{\theta} x_k + \frac{1}{2L^2} \sum_k \sum_l \beta_{kl}^{\theta} x_k x_l \right) \\ \text{and} \quad v_i(\boldsymbol{x}) &= \sqrt{\theta_0} \left(\frac{1}{L} \sum_k \alpha_k^{v_i} x_k + \frac{1}{2L^2} \sum_k \sum_l \beta_{kl}^{v_i} x_k x_l \right), \end{aligned} \tag{181}$$

with parameters $\rho_0, \theta_0 \in \mathbb{R} > 0$, $\alpha \in \mathbb{R}^{d_x}$, $\beta \in \mathbb{R}^{2d_x}$ and $\beta_{kl} \equiv \beta_{lk}$, $k, l \in \{1, \ldots, d_x\}$. To generate a sample $\{\boldsymbol{x}^l, \boldsymbol{c}^l\}$ according to Equation (180), first, a vector $\tilde{\boldsymbol{x}}$ is

chosen with uniform probability from the region B, and accepted as sample $\boldsymbol{x}^l$ with probability $\rho(\tilde{\boldsymbol{x}}) \Big/ \max_B \rho$. Next, a sample for the state vector $\boldsymbol{c}^l$ is obtained by setting $\boldsymbol{c}^l = \boldsymbol{v}\left(\boldsymbol{x}^l\right) + \sqrt{\theta(\boldsymbol{x}^l)}\boldsymbol{\psi}$, where $\boldsymbol{\psi} \in \mathbb{R}^{d_c}$ is a vector of samples from the standard normal distribution.

Subsequently, various kernels are used to estimate the normalized derivatives $\frac{L^\nu}{\rho_0}\boldsymbol{\partial}^{\boldsymbol{\alpha}} F^0_{\mathbf{0}}$, $\frac{L^\nu}{\rho_0\sqrt{\theta_0}}\boldsymbol{\partial}^{\boldsymbol{\alpha}} F^0_{\boldsymbol{\delta}_i}$ and $\frac{L^\nu}{\rho_0\theta_0}\partial^\alpha \frac{1}{2} F^1_{\mathbf{0}}$ at $\boldsymbol{x} = \mathbf{0}$, where the quantities $F^s_{\boldsymbol{\beta}}$ are defined according to Equation (179), with the multi index $\boldsymbol{\alpha}$ successively chosen from $\{\mathbf{0}, \boldsymbol{\delta}_{m_1}, \boldsymbol{\delta}_{m_1} + \boldsymbol{\delta}_{m_2}\}$, with $m_1, m_2 \in \{1, \ldots, d_x\}$. From the phase density implied by Equation (180), one finds:

$$\frac{F^0_{\mathbf{0}}}{\rho_0} = \frac{1}{\rho_0}\rho(\boldsymbol{x}), \quad \frac{F^0_{\boldsymbol{\delta}_i}}{\rho_0\sqrt{\theta_0}} = \frac{\rho(\boldsymbol{x})v_i(\boldsymbol{x})}{\rho_0\sqrt{\theta_0}} \quad \text{and} \quad \frac{\frac{1}{2}F^1_{\mathbf{0}}}{\rho_0\theta_0} = \frac{\rho(\boldsymbol{x})}{\rho_0\theta_0}\left(\frac{1}{2}\|\boldsymbol{v}(\boldsymbol{x})\|^2 + \frac{d_c}{2}\theta(\boldsymbol{x})\right). \tag{182}$$

The exact expressions for the estimated quantities are thus:

F	$F\Big\vert_{\boldsymbol{x}=\mathbf{0}}$	$\partial_{x_{m_1}} F\Big\vert_{\boldsymbol{x}=\mathbf{0}}$	$\partial_{x_{m_1}}\partial_{x_{m_2}} F\Big\vert_{\boldsymbol{x}=\mathbf{0}}$
$\frac{F^0_{\mathbf{0}}}{\rho_0}$	1	$\alpha^\rho_{m_1}$	$\beta^\rho_{m_1 m_2}$
$\frac{F^0_{\boldsymbol{\delta}_i}}{\rho_0\sqrt{\theta_0}}$	0	$\alpha^{v_i}_{m_1}$	$\beta^{v_i}_{m_1 m_2} + \alpha^\rho_{m_1}\alpha^{v_i}_{m_2} + \alpha^\rho_{m_2}\alpha^{v_i}_{m_1}$
$\frac{\frac{1}{2}F^1_{\mathbf{0}}}{\rho_0\theta_0}$	$\frac{d_c}{2}$	$\frac{d_c}{2}\left(\alpha^\theta_{m_1} + \alpha^\rho_{m_1}\right)$	$\frac{d_c}{2}\left(\beta^\theta_{m_1 m_2} + \beta^\rho_{m_1 m_2}\alpha^\rho_{m_1}\alpha^\theta_{m_2} + \alpha^\rho_{m_2}\alpha^\theta_{m_1}\right) + \sum_{k=1}^{d_c}\alpha^{v_k}_{m_1}\alpha^{v_k}_{m_2}$

(183)

Here, the case $d_x = d_c = 2$ is considered. Minimum variance and optimal Type I and Type II kernels of orders $\nu + 2$ and $\nu + 4$ are used to estimate the quantities in Equations (183) with sample sizes up to $N = 1 \times 10^7$ from $M = 500$ independent realizations. Four different sets of parameters α and β—listed in Table 2—are investigated. Since the reference parameters ρ_0, θ_0 and the domain size L do not appear on any right hand side of the Equations (183), they are set to unity.

The results in terms of the root mean squared error (RMSE) of the estimation of the quantities in Equations (183) are presented in Figures 45, 46, 47 and 48 in Appendix D.4. For comparison, also the analytical form of the RMSE is plotted. Two result series, concerning the estimation of $\frac{1}{2}F^1_{\mathbf{0}}$ with parameter set 2, as well as the estimation of $\frac{1}{2}\partial^2_{x_1}F^1_{\mathbf{0}}$ with parameter set 4, are presented enlarged in Figures 39 and 40, respectively. All plots confirm the expected trends: at first, RMSE is dominated by the variance contribution and decreases with $N^{-1/2}$, until the bias becomes dominant, see, e.g., Figure 39. For the cases where the bias is not zero by construction, its value is approached asymptotically. To achieve lower RMSE, the bandwidth would have to be adjusted as a function of the sample size, as per the consistency requirements (119). Since for this study, the variation of the estimated quantities is normalized over the kernel support, one may equivalently compare the parameter cases with large and small values of the coefficients α and β, see, for example, Figure 40. From the presented results, one may conclude that for small

Table 2: Coefficients for four different polynomial spatial variations of the specified macroscopic parameters for the kernel RMSE Monte Carlo study.

	1				2			
	ρ	θ	v_1	v_2	ρ	θ	v_1	v_2
α_1	0	0	.025	.025	.025	.025	.025	.025
α_2	0	0	.05	−.025	.05	−.025	.05	−.025
β_{11}	0	0	.0125	.05	.0125	.05	.0125	.05
β_{12}	0	0	.0125	−.025	.0125	−.025	.0125	−.025
β_{22}	0	0	.025	−.025	.025	−.025	.025	−.025
	3				4			
	ρ	θ	v_1	v_2	ρ	θ	v_1	v_2
α_1	0	0	0	0	−.5	−.5	−.5	−.5
α_2	0	0	0	0	.25	.25	.25	.25
β_{11}	.05	.0125	.05	.0125	.75	.75	.75	.75
β_{12}	−.025	.0125	−.025	.0125	−.5	−.5	−.5	−.5
β_{22}	−.025	.025	−.025	.025	.0125	.0125	.0125	.0125

sample sizes and limited spatial variation of the phase density, lower order kernels are the preferred choice. The differences between kernel classes (minimum variance vs. Type I vs. Type II) are less pronounced, but may become significant at large sample sizes and large spatial variation of the phase density, as illustrated by the results using parameter set 4 in Figure 48. For the estimates with moderate spatial variation (parameter sets 1, 2, and 3), using kernels of order $\nu + 2$, about 1×10^5 samples are required to achieve a RMSE on the order of 0.01.

26 Discussion and Conclusion

26.1 Other Kernels

Besides the kernels derived above, many other functions satisfy the necessary requirements (124) for moment derivative estimation. Some interesting choices, which have equal or worse variance compared to the optimal choices $K_{\boldsymbol{\alpha}}^{v,k}$, and do not, in general, fulfill the additional conditions (131), are mentioned below.

26.1.1 Derivatives of Density Kernels

Due to the linearity of Equations (109) and (113), any kernel defined by $K_{\boldsymbol{\alpha}}^{d}(\boldsymbol{x}) := a\,\boldsymbol{\partial}^{\boldsymbol{\alpha}} K(\boldsymbol{x})$, where K is an appropriately differentiable kernel satisfying the conditions (124) for $\boldsymbol{\alpha} = \mathbf{0}$, and $a \in \mathbb{R}$ a normalization factor, may be used in the estimate $\widehat{\boldsymbol{\partial}^{\boldsymbol{\alpha}} F_g}$. For example, Duong et al. [Duo+08] developed the appropriate formalism along with significance tests in the context of multivariate density derivative estimation.

Kernel interpolation (cf. Section 23) is at the heart of the Smoothed Particle

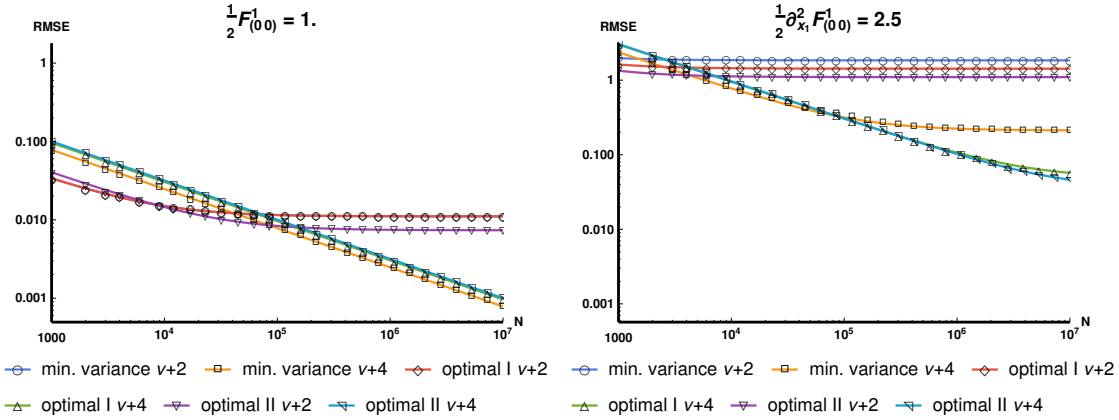


Figure 39: Empirical (symbols) and exact (lines) RMSE vs. sample size for the estimation of $\frac{1}{2}F^1_{\mathbf{0}}$ ($\boldsymbol{\alpha} = 0$) with parameter set 2 using various kernels. Note that the results using the minimum variance kernel and the optimal kernel Type I of order $\nu + 2$ coincide in this case.

Figure 40:]
Empirical (symbols) and exact (lines) RMSE vs. sample size for the estimation of $\frac{1}{2}\partial^2_{x_1}F^1_{\mathbf{0}}$ ($\boldsymbol{\alpha} = (2, 0)$) with parameter set 4 using various kernels.

Hydrodynamics (SPH) method [Mon92], and it might be worthwhile to investigate the consequences of using derivative kernels $K_{\boldsymbol{\alpha}}$ in place of derivatives of kernels $\boldsymbol{\partial}^{\boldsymbol{\alpha}} K$—the standard practice—in this method.

26.1.2 Product Kernels

The product kernel (see, e.g. [Sin76])

$$K^{\pi}_{\boldsymbol{\alpha}}(\boldsymbol{r}) := \prod_{i=1}^{d_x} K_{\alpha_i}(r_i), \tag{184}$$

fulfills conditions (124), if the kernels $K_{\alpha_i} : [-1, 1] \to \mathbb{R}$ fulfill the corresponding conditions in $d_x = 1$. Conditions (131) are only fulfilled in specific cases (e.g., if $\boldsymbol{\alpha} = a\mathbf{1}$ and $K_{\alpha_i} \equiv K\, \forall i$). The order of the kernel $K^{\pi}_{\boldsymbol{\alpha}}$ is given by the maximum order of the kernels K_{α_i}. The case $|\boldsymbol{\alpha}| = 1$ for density gradient estimation has received significant attention in literature, cf., e.g., Refs. [FH75; GMM85; CM02].

26.1.3 Uniform Kernel for $\nu = 0$

Kernels for the case $\boldsymbol{\alpha} = 0$, $\nu = |\boldsymbol{\alpha}| = 0$ are equivalent to density estimation kernels, which have been studied extensively, cf., e.g., Refs. [Epa69; WJ94; GMM85; Sco15]. Note that the simplest possible kernel, the uniform kernel, which is a constant over the support $[-1, 1]^{d_x}$, i.e.,

$$K^{u}(\boldsymbol{x}) := \frac{1}{2^{d_x}}, \tag{185}$$

is of particular relevance to particle methods such as Direct Simulation Monte Carlo (DSMC), since the corresponding moment estimates are simple arithmetic averages. It may be, for example, more important to estimate the total kinetic energy of particles in a given volume of space than to have a potentially superior estimate of that quantity at a given location. In DSMC, the state space moments defined in Equation (179) within a grid cell $B_{\text{cell}} \in \mathcal{B}\left(\mathbb{R}^{d_x}\right)$ are typically estimated by

$$\widehat{F^s_\beta} = \frac{1}{V_{\text{cell}}} \sum_{l:\, \boldsymbol{x}^l \in B_{\text{cell}}} w^l \left(c^l\right)^{2s} \left(\boldsymbol{c}^l\right)^{\beta}, \tag{186}$$

where $V_{\text{cell}} = |B_{\text{cell}}|$ is the volume of the cell. If the cell is a rectangular cuboid, this is equivalent to Equation (113) using $K = K^u$ and setting $\boldsymbol{h}$ equal to the vector of cell half-widths.

26.1.4 Discontinuous Kernel for $\nu = 1$

The analogue to the uniform kernel (cf. Equation (185)) for the case $\boldsymbol{\alpha} = \boldsymbol{\delta}_m$, $m \in \{1, \ldots, d_x\}$, $\nu = |\boldsymbol{\alpha}| = 1$, is the function

$$K^s_{\boldsymbol{\delta}_m}(\boldsymbol{x}) := -\frac{1}{2^{d_x-1}} \text{Sgn}(x_m) = -\frac{1}{2^{d_x-1}} \times \begin{cases} -1, & x_m < 0 \\ 0, & x_m = 0 \\ 1 & else, \end{cases} \tag{187}$$

also with support $[-1, 1]^{d_x}$, which yields the finite difference estimate of the mth component of the moment gradient. The estimate is equivalent to first computing separate estimates $\widehat{F_g^-}$ and $\widehat{F_g^+}$ using a uniform kernel in each of the intervals $[-1, 0]_m \times [-1, 1]^{d_x-1}$ and $[0, 1]_m \times [-1, 1]^{d_x-1}$, respectively, and subsequently setting $\widehat{\partial_m F_g} := \left(\widehat{F_g^+} - \widehat{F_g^-}\right)$.

26.2 Time Averaging of Estimates

One may additionally use time averaging to increase the quality of the estimates $\widehat{\boldsymbol{\partial}^{\boldsymbol{\alpha}} F_g}$, according to the procedure discussed in [JTH10]: let $\widetilde{\widehat{F_g}}^{n-1}$ be the time-averaged estimate of quantity F_g up to time step number $n-1$, and $\widehat{F_g}^n$ the estimate of F_g based on the sample currently available at time step n. The update of $\widetilde{\widehat{F_g}}^{n-1}$ to $\widetilde{\widehat{F_g}}^{n}$ then reads

$$\widetilde{\widehat{F_g}}^{n} = (1-\mu)\widehat{F_g}^n + \mu \widetilde{\widehat{F_g}}^{n-1}, \tag{188}$$

where $\mu \in \mathbb{R}$, $0 \leq \mu \leq 1$, is a weighting factor. Setting $\mu := {}^{n-1}\!/_n$ results in a simple arithmetic average of all samples, and this factor should be chosen for statistically stationary sample processes.

26.3 Conclusions

This part of the thesis presents the extension of established non-parametric density estimation techniques to the estimation of mixed spatial derivatives of statistics of the phase density in arbitrary dimensions. Specifically, it is demonstrated that kernel estimation may be used to estimate mixed spatial derivatives of generalized state space moments of the phase density, from which kernel interpolation of mixed spatial derivatives of deterministic functions of the position follows as a special case. Consistency requirements for kernels, together with asymptotic bias and variance expressions, as well as the asymptotically optimal bandwidth, are derived. Three classes of kernels are investigated: "minimum variance", and two types of "optimal" kernels, minimizing asymptotic variance and asymptotic mean squared error, under additional regularization conditions, respectively. In addition to closed-form expressions for the kernels in d dimensions for the estimation of the νth-order mixed spatial derivatives, several explicit kernel functions for dimensions $1, 2$ and 3 for the estimation of mixed derivatives up to order 4 are provided. A Monte Carlo study illustrates the root mean squared error dependence on kernel type and sample size for the estimation of state space moments from samples from a two-space-two-state-dimensional Maxwellian phase density with parameters varying quadratically in space.

The presented estimation idea may prove useful for any computational task where spatial derivatives of scattered data are sought: in the context of computations where physical space is discretized into axis aligned grid cells, one would choose the kernel bandwidth vector equal to the cell half-width vector. The estimation of any desired spatial derivative of the data at each cell center may then proceed in an "embarrassingly" parallel fashion, only requiring the sample data within each cell. This inherent locality of the kernel approach makes it particularly simple to implement in the context of locally refined computational grids and parallel algorithms based on domain decomposition. In fact, the asymptotically optimal bandwidth may be used to inform automatic mesh refinement procedures. Further, all obtained values may be time-averaged to reduce noise. Potential applications that may benefit from these properties include molecular dynamics, smoothed particle hydrodynamics, particle-in-cell, direct simulation Monte Carlo, and n-body simulations, as well as spatial econometrics, computer vision and machine intelligence.

For this contribution, the explicit kernels were derived under the assumption that they should be continuous functions with compact support on $[-1, 1]^{d_x}$. It would be worthwhile to investigate and compare the performance of other kernels that do not share these restrictions. For applications in which data are investigated in a nearest neighbor fashion, kernels with support on the unit sphere may be useful. With regards to non-continuous kernels, the approach presented by Boykin [Boy03] could be used to generalize the finite difference kernel.

Part V

Gradient-Based Automatic Mesh Refinement

In Parts II & III of this thesis, it was demonstrated that the FP-DSMC algorithm offers numerous advantages over "pure" DSMC in terms of computational resources needed to achieve a desired level of accuracy. As mentioned in the introduction (Part I, Section 3.5), if the greatest accuracy given certain computational resources is desired, the FP collision operator can serve as an automatic fall-back to prevent excessive increase of execution time and load imbalance which would occur if cells in the domain fail to fulfill the DSMC resolution requirements, necessitating the computation of a large number of binary collisions compared to the number of particles. Indeed, in Part II it was demonstrated that using FP where DSMC would be under-resolved also improves accuracy. However, the full potential of FP-DSMC has not yet been leveraged. In order to further improve the performance, one would like to apply the FP operator in even larger regions of the simulated physical domain. To this end, the computational mesh needs to be as coarse as possible, but still fine enough to satisfy the FP resolution requirements. Now, DSMC serves as the fall-back, in cells where there are not enough collisions to justify the drift-diffusion ansatz, and where DSMC is more efficient anyway.

The following sections describe the necessary extensions of the implementation presented so far in order to maximize performance and accuracy for the FP-DSMC procedure. While the implementation of the gradient-based automatic mesh refinement and the local variable time stepping are completed, and the necessary changes to the code to incorporate the improved position integration scheme outlined below have been made, the necessary numerical validation studies are still "work in progress" at the time of writing.

27 Computing the Mesh Refinement Criterion

The Fokker-Planck algorithm requires the resolution of the spatial gradients of the macroscopic quantities. This ensures that a given particle is subject to constant coefficients along its trajectory during one time step. While the necessary developments to efficiently adapt the mesh according to any desired local criteria have been developed and presented in Part III, the next step is to make use of the theory developed in Part IV to base the automatic mesh refinement on the length scales implied by the gradients of the macroscopic fields.

One would like all cells of the mesh to satisfy the following gradient resolution

requirement with threshold parameter κ:

$$\sum_{i=1}^{d_x} \Delta_i^2 \max_j L_{ij}^2 < \kappa^2, \tag{189}$$

where the matrix L contains the normalized gradients of selected macroscopic fields Q_j, i.e.,

$$L_{ij} = \frac{1}{Q_{(j)}^{\text{ref}}} \left(\partial_{x_i} Q_{(j)} \right), \tag{190}$$

where no summation over the index j is implied, and $Q^{\text{ref}} \neq 0$ is a suitable reference value for the quantity Q.

27.1 Conversion of Derivatives of Full Moments to Their Centered Counterparts

Of the relevant quantities Q for gas dynamics, i.e., the macroscopic fields, only density and momentum are directly defined in terms of the "full" moments

$$F_{\boldsymbol{\beta}}^{a,b,s} = \langle \left(\sum_{i=a}^{b} c_i^2 \right)^s \boldsymbol{c}^{\boldsymbol{\beta}} \rangle, \tag{191}$$

cf. Equation (179), where $\boldsymbol{\beta}$ is a multi index, and, if omitted, $a = 1$ and $b = d_c$ is assumed. To compute other macroscopic fields, also the centered moments $\rho_{\boldsymbol{\beta}}^{a,b,s}$ are required, defined as

$$\rho_{\beta}^{a,b,s} = \langle \left(\sum_{i=a}^{b} C_i^2 \right)^s \boldsymbol{C}^{\beta} \rangle, \tag{192}$$

where $\boldsymbol{C}$ is the state vector with the velocity components replaced by their thermal counterparts, i.e., $\boldsymbol{C} = \boldsymbol{c} - \boldsymbol{U}_0$, with $\boldsymbol{U}_0 := (\boldsymbol{U}, \boldsymbol{0})$, where $U_i \equiv F_{\boldsymbol{\delta}_i}^0 \big/ F_{\boldsymbol{0}}^0$, $i = 1, 2, 3$, and, as before, $\boldsymbol{\delta}_i = (\delta_{1i}, \delta_{2i}, \ldots, \delta_{ni})$ (cf. Sec. 18). Since only derivatives of the full moments may be directly computed via the kernel approach, an explicit expression to compute the derivatives of the centered moments from their full counterparts is required, which will be derived in the following.

Equation (192) may be re-cast using the multinomial theorem to yield

$$\rho_{\boldsymbol{\beta}}^{a,b,s} = \sum_{\substack{|\boldsymbol{\eta}|=s \\ \text{supp}(\boldsymbol{\eta})=\{a,\ldots,b\}}} \binom{s}{\boldsymbol{\eta}} \langle (\boldsymbol{c} - \boldsymbol{U}_0)^{2s\boldsymbol{\eta}+\boldsymbol{\beta}} \rangle, \tag{193}$$

where $\text{supp}(\boldsymbol{\eta})$ is the support of the multi index $\boldsymbol{\eta}$, defined as the set of indices $\{i : \eta_i > 0\}$. Equation (193), in turn, allows the straightforward application of the multi-binomial theorem to write the centered moments in terms of the full moments $F_{\boldsymbol{\beta}}^0$, viz.

$$\rho_{\boldsymbol{\beta}}^{a,b,s} = \sum_{\substack{|\boldsymbol{\eta}|=s \\ \text{supp}(\boldsymbol{\eta})=\{a,\ldots,b\}}} \binom{s}{\boldsymbol{\eta}} (2s\boldsymbol{\eta} + \boldsymbol{\beta})! \sum_{\boldsymbol{\gamma}+\boldsymbol{\nu}=(2s\boldsymbol{\eta}+\boldsymbol{\beta})} (-1)^{|\boldsymbol{\nu}|} \frac{F_{\boldsymbol{\gamma}}^0}{\boldsymbol{\gamma}!} \frac{\boldsymbol{U}_0^{\boldsymbol{\nu}}}{\boldsymbol{\nu}!}. \tag{194}$$

The νth-order mixed partial spatial derivatives of the moments $\rho_{\beta}^{a,b,s}$ follow from the generalized Leibniz rule:

$$\boldsymbol{\partial}^{\boldsymbol{\alpha}}\rho_{\boldsymbol{\beta}}^{a,b,s} = \sum_{\substack{|\boldsymbol{\eta}|=s \\ \operatorname{supp}(\boldsymbol{\eta})=\{a,\dots,b\}}} \binom{s}{\boldsymbol{\eta}} (2s\boldsymbol{\eta}+\boldsymbol{\beta})! \sum_{\boldsymbol{\gamma}+\boldsymbol{\nu}=(2s\boldsymbol{\eta}+\boldsymbol{\beta})} (-1)^{|\boldsymbol{\nu}|} \boldsymbol{\alpha}! \sum_{\boldsymbol{\mu}+\boldsymbol{\omega}=\boldsymbol{\alpha}} \frac{\boldsymbol{\partial}^{\boldsymbol{\mu}} F_{\boldsymbol{\gamma}}^{0}}{\boldsymbol{\mu}!\boldsymbol{\gamma}!} \frac{\boldsymbol{\partial}^{\boldsymbol{\omega}} \boldsymbol{U}_{0}^{\boldsymbol{\nu}}}{\boldsymbol{\omega}!\boldsymbol{\nu}!}. \tag{195}$$

Explicitly evaluating the term $\boldsymbol{\partial}^{\boldsymbol{\omega}}\boldsymbol{U}_0^{\boldsymbol{\nu}}$ requires the multivariate extension of the chain rule, here given for a general function $z\colon \boldsymbol{x} \in \mathbb{R}^{d_x} \to \boldsymbol{y} \in \mathbb{R}^{d_c} \to \mathbb{R}$ [Ma09, Equation (6)]:

$$\boldsymbol{\partial}^{\boldsymbol{\alpha}} z = \boldsymbol{\alpha}! \sum_{(l,\boldsymbol{p},\boldsymbol{m})\in\mathcal{D}} z^{(\boldsymbol{m})} \prod_{k=1}^{l} \frac{1}{\boldsymbol{m}_k!} \left[\frac{1}{\boldsymbol{p}_k!}\boldsymbol{\partial}^{\boldsymbol{p}_k}\boldsymbol{y}\right]^{\boldsymbol{m}_k}. \tag{196}$$

In Equation (196), $\mathcal{D}$ is the set of all possible decompositions of the multi index $\boldsymbol{\alpha}$ into l parts $\boldsymbol{p}_1, \dots, \boldsymbol{p}_l$, $|\boldsymbol{p}_i| > 0$, with multiplicities $\boldsymbol{m}_1, \dots, \boldsymbol{m}_l$, $|\boldsymbol{m}_i| > 0$, for which $\boldsymbol{\alpha} = |\boldsymbol{m}_1|\boldsymbol{p}_1 + \dots + |\boldsymbol{m}_l|\boldsymbol{p}_l$, such that $\boldsymbol{p}_i \neq \boldsymbol{p}_j\ \forall i \neq j$, with total multiplicity $\boldsymbol{m} = \boldsymbol{m}_1 + \dots + \boldsymbol{m}_l$. Note that the parts $\boldsymbol{p}_i$ and multiplicities $\boldsymbol{m}_i$ are multi indices of orders d_x and d_c, respectively. Further, $z^{(\boldsymbol{m})} \equiv \frac{\partial^{|\boldsymbol{m}|}}{\partial_{y_1}^{m_1}\cdots\partial_{y_s}^{m_s}} z(\boldsymbol{y})$, and for $z = \boldsymbol{y}^{\boldsymbol{\nu}}$,

$$z^{(\boldsymbol{m})} = \begin{cases} \frac{\boldsymbol{\nu}!}{(\boldsymbol{\nu}-\boldsymbol{m})!}\boldsymbol{y}^{\boldsymbol{\nu}-\boldsymbol{m}} & \text{if } \boldsymbol{m} \leq \boldsymbol{\nu}, \\ 0 & \text{otherwise.} \end{cases} \tag{197}$$

As before, $\boldsymbol{0}^{\boldsymbol{0}} \equiv 1$. Also, note that

$$\left[\frac{1}{\boldsymbol{p}_k!}\boldsymbol{\partial}^{\boldsymbol{p}_k}\boldsymbol{y}\right]^{\boldsymbol{m}_k} \equiv \left[\frac{1}{\boldsymbol{p}_k!}\boldsymbol{\partial}^{\boldsymbol{p}_k} y_1\right]^{m_{k1}} \times \dots \times \left[\frac{1}{\boldsymbol{p}_k!}\boldsymbol{\partial}^{\boldsymbol{p}_k} y_{d_c}\right]^{m_{kd_c}}. \tag{198}$$

For the application presented in this part of the thesis only gradients are of interest, i.e., $\boldsymbol{\alpha} = \boldsymbol{\delta}_i$, $i = 1, \dots, d_x$. In this case, Equation (195) simplifies to

$$\boldsymbol{\partial}^{\boldsymbol{\delta}_i}\rho_{\boldsymbol{\beta}}^{a,b,s} = \sum_{\substack{|\boldsymbol{\eta}|=s \\ \operatorname{supp}(\boldsymbol{\eta})=\{a,\dots,b\}}} (2s\boldsymbol{\eta}+\boldsymbol{\beta})! \sum_{\boldsymbol{\gamma}+\boldsymbol{\nu}=(2s\boldsymbol{\eta}+\boldsymbol{\beta})} (-1)^{|\boldsymbol{\nu}|} \frac{\boldsymbol{U}_0^{\boldsymbol{\nu}}\, \boldsymbol{\partial}^{\boldsymbol{\delta}_i} F_{\boldsymbol{\gamma}}^{0} + F_{\boldsymbol{\gamma}}^{0}\, \boldsymbol{\partial}^{\boldsymbol{\delta}_i} \boldsymbol{U}_0^{\boldsymbol{\nu}}}{\boldsymbol{\gamma}!\boldsymbol{\nu}!}. \tag{199}$$

Inserting $\boldsymbol{\alpha} = \boldsymbol{\delta}_i$ and $z = \boldsymbol{U}_0^{\boldsymbol{\nu}}$ into Equation (196), it is obvious that in this case each decomposition will have only one part, i.e, $l = 1$, with $\boldsymbol{p}_1 \equiv \boldsymbol{\delta}_i$. There exist d_c such decompositions, each with $|\boldsymbol{m}_1| = 1$. Due to Equation (197), with $\boldsymbol{y} = \boldsymbol{U}_0$ and the definition of $\boldsymbol{U}_0$, only the d_x decompositions with $\boldsymbol{m}_1 = \boldsymbol{\delta}_j$, $j = 1, \dots, d_x$, contribute to $\boldsymbol{\partial}^{\boldsymbol{\delta}_i}\boldsymbol{U}_0^{\boldsymbol{\nu}}$. It follows that

$$\begin{aligned} \boldsymbol{\partial}^{\boldsymbol{\delta}_i}\boldsymbol{U}_0^{\boldsymbol{\nu}} &= \sum_{j=1}^{d_x} \frac{\boldsymbol{\nu}!}{(\boldsymbol{\nu}-\boldsymbol{\delta}_j)!} \boldsymbol{U}_0^{\boldsymbol{\nu}-\boldsymbol{\delta}_j} \partial_{x_i} U_j \\ &= \sum_{j=1}^{d_x} \nu_j \boldsymbol{U}_0^{\boldsymbol{\nu}-\boldsymbol{\delta}_j} \partial_{x_i} U_j. \end{aligned} \tag{200}$$

27.2 Gradients of Selected Macroscopic Quantities for Automatic Mesh Refinement

The following macroscopic fields are chosen as components of the vector $\boldsymbol{Q}$:

$$\rho = F^0_{\mathbf{0}}, \qquad \text{(density)} \tag{201}$$

$$U_i = F^0_{\boldsymbol{\delta}_i} \big/ \rho, \quad i = 1,2,3, \qquad \text{(mean velocity)} \tag{202}$$

$$\begin{aligned} \text{and} \quad \theta &= \rho^{1,3,1}_{\mathbf{0}} \big/ 3\rho \\ &= 2\sum_{i=1}^{3} \Big(\frac{1}{2} F^0_{2\boldsymbol{\delta}_i} - F_{\boldsymbol{\delta}_i} \boldsymbol{U}_0^{\boldsymbol{\delta}_i} + \frac{1}{2}\rho \boldsymbol{U}_0^{2\boldsymbol{\delta}_i} \Big) \Big/ 3\rho \qquad \text{(temperature in energy units)} \\ &= \Big(F^{1,3,1}_{\mathbf{0}} - \rho \|\boldsymbol{U}\|^2 \Big) \big/ 3\rho . \end{aligned} \tag{203}$$

The respective gradients, in terms of the gradients of the full moments F, read

$$\partial_{x_i} \rho = \partial_{x_i} F^0_{\mathbf{0}}, \tag{204}$$

$$\partial_{x_i} U_j = \Big(\rho\, \partial_{x_i} F^0_{\boldsymbol{\delta}_j} - F^0_{\boldsymbol{\delta}_j} \partial_{x_i} \rho \Big) \big/ \rho^2 , \tag{205}$$

$$\begin{aligned} \text{and} \quad \partial_{x_i} \theta &= \Big(\rho \partial_{x_i} \rho^{1,3,1}_{\mathbf{0}} - \rho^{1,3,1}_{\mathbf{0}} \partial_{x_i} \rho \Big) \big/ 3\rho^2 \\ &= \left(\partial_{x_i} F^{1,3,1}_{\mathbf{0}} - \|\boldsymbol{U}\|^2 \partial_{x_i} \rho - 2\rho \sum_{j=1}^{3} U_j \partial_{x_i} U_j \right) \Big/ 3\rho \\ &\qquad - \Big(F^{1,3,1}_{\mathbf{0}} - \rho \|\boldsymbol{U}\|^2 \Big) \partial_{x_i} \rho \big/ 3\rho^2 \\ &= \partial_{x_i} F^{1,3,1}_{\mathbf{0}} \big/ 3\rho - \frac{2}{3} \sum_{j=1}^{3} U_j \partial_{x_i} U_j - F^{1,3,1}_{\mathbf{0}} \partial_{x_i} \rho \big/ 3\rho^2 . \end{aligned} \tag{206}$$

For the estimation of the gradients of the full moments, that is, the quantities $\partial_{x_i} F^{a,b,s}_{\boldsymbol{\beta}}$, the third-order minimum variance kernel $K(\boldsymbol{x}) = -\frac{3}{8}\boldsymbol{x}$ is used (cf. 24.1). Since the gradient estimates suffer from greater noise than the non-derivated moments, for practical purposes, exponentially weighted time averaging is used, and further, the grid adaption is limited to cells where the averages contain a certain minimal amount of samples, on the order of 10×10^5. Similarly, a suitable compromise between numerical accuracy and stability of the refinement process must be found when choosing the refinement threshold κ (see Equation (189)). Typical values for κ will be on the order of 5–10%, see also [PG16].

28 Local Variable Time Steps

Both the DSMC operator and the Fokker-Planck operator place restrictions on the time step size. For the Fokker-Planck operator, a CFL criterion should be satisfied. DSMC additionally requires that the time step size be smaller than the mean collision time. When using a locally refined mesh, the CFL limitation can become

severe. In particular, the octree grid used in the work presented here introduces significant differences in the cell sizes. Having to choose the time step according to the smallest cells on the grid may thus cause significant (unnecessary) overhead. Furthermore, regardless of the choice of the collision operator, if the grid is not refined to ensure a uniform number of particles in each cell, small cells may suffer from too few particles being present per time step, and hence slow convergence of the averages in the best case, and wrong results in the worst.

A common solution to address both issues is to introduce a cell local time step size [KB00]. This means that a single iteration of the particle algorithm no longer represents the same amount of physical time in each cell. Clearly, such a scheme may only be used for stationary flows. In order to reduce the particle-count imbalance throughout the domain of a mean free path adjusted grid, Kannenberg and Boyd [KB00] suggest choosing a time step size inversely proportional to density. If the grid spacing is proportional to the mean free path, this may be realized by scaling the time step according to the cell volume V. Here, the motivation is to satisfy the CFL criterion on the time step size. Accordingly, the time step should be proportional to V^{1/d_x}. For the octree grid layout presented in Part III, the cell volume is $V = V_{\text{ref}} 2^{d_x \times l}$, where l is the grid level.

To maintain some degree of flexibility, the local time step in cell i is computed according to

$$\Delta t_i = \Delta t_{\text{ref}} (f)^{l_i}, \tag{207}$$

where f is an input parameter. Local time stepping is disabled by setting $f = 1$. $f = 2$ maintains the CFL criterion, and $f = 2^{d_x}$ produces cell-volume proportional time steps. In general, it may be desirable to choose the time step according to local flow properties. This would be difficult to realize within the current code architecture, cf. the paragraph concerning implementation below.

28.1 Particle Weight Adjustment

The inter-cell differences in time step size are compensated by adjusting the particles' statistical weights [Kan98; WTW04]: let the reference time step be Δt_{ref}, and the time step in a cell with index i be adjusted to $\sigma_i \Delta t_{\text{ref}}$. Let N_{ij} be the number of particles, each with index l and weight $\varsigma_i w^l$, crossing from cell i to cell j during a given time step, where ς_i is the weight factor applied to particles in cell i. Cell i will "emit" a numerical flux of a macroscopic quantity $Q = \langle g \rangle$ to cell j equal to

$$\Phi_{ij} = \sum_{l=1}^{N_{ij}} \varsigma_i w_i^l g^l \Big/ A_{ij} \sigma_i \Delta t_{\text{ref}}, \tag{208}$$

where A_{ij} is the area of the interface shared between cells i and j. During the same iteration, if the time size step in cell j is $\sigma_j \Delta t_{\text{ref}}$, the same particle flux will result in cell j receiving a corresponding numerical flux

$$\Phi_{ji} = \sum_{l=1}^{N_{ji}} \varsigma_j w_i^l g^l \Big/ A_{ji} \sigma_j \Delta t_{\text{ref}}. \tag{209}$$

Clearly, $\Phi_{ij} \stackrel{!}{=} \Phi_{ji}$. Since $N_{ij} \equiv N_{ji}$ and $A_{ij} \equiv A_{ji}$, this condition necessitates

$$\frac{\sigma_i}{\varsigma_i} \stackrel{!}{=} \frac{\sigma_j}{\varsigma_j}. \tag{210}$$

Without loss of generality, this requirement is enforced by setting $\sigma_i \equiv \varsigma_i \, \forall i$ according to Equation (207).

28.2 Implementation

A defining characteristic of the code developed and presented in this thesis is that the particle movement at each time step can be computed on each subdomain (with dedicated MPI process) without communication. In other words, the final particle position should not depend on the grid and/or solution, but only on the boundaries. To implement local time steps, the particles should be traced through the mesh, with their remaining flight time updated precisely at the point where they cross an interface between cells with different weights. Clearly, this would violate the design assumption above, necessitating techniques such as "ghost cells" (halo cells) on every subdomain to avoid per-particle communication events. To avoid this difficulty, the time step in each cell is computed according to Equation (207). To complete the particle movement on each MPI process, after each dynamic grid adaptation step, only the manifold defined by the union of all interfaces between cells of different levels needs to be combined and duplicated on all MPI processes. A flow-dependent criterion would necessitate a ghost cell value exchange at every time step. The manifold is stored as a collection of quadrilaterals and treated the same as all other boundaries in the simulation, that is, the ray tracing library computes the particle trajectory intersection.

After a particle is determined to hit the "time zone" boundary, it is moved to the intersection location, and its remaining time of flight scaled by the factor f, if the crossing occurs from a smaller to a larger cell, and by $1/f$ otherwise. Note that the actual levels of the cells in each region with homogeneous cell level (time zone) does not matter. This procedure relies on the fact that levels of neighboring cells are ensured to differ by at most one.

Note that the interface manifold for a 4-level grid for the planetary probe simulations presented in Part III requires on the order of 10 MB of memory, while the entire grid requires on the order of 4 GB, and, for a simulation with 400×10^6 particles, the particle storage requires more than 75 GB.

28.2.1 Boundary Sampling

The code presented here uses immersed boundaries. In the context of local variable time steps and particle weights, particles of different weights and time steps might interact with the same boundary element. This would complicate the computation of the various average surface fluxes. A far greater difficulty arises at inflows, since it is not clear a priori which weight and time step the particles should receive when

they are sampled. Both difficulties are avoided by statically refining cells intersected by geometry to a pre-defined level and preventing these cells from being refined or coarsened during the simulation. This ensures that the time zone boundary does not intersect any geometry, and, consequently, all particles interacting with geometry will have equal weights. At inflow boundaries, the permanent cell level is used to compute the initial particle weights and time steps.

29 Fokker-Planck Algorithm with Higher-Order Position Integration

As the cell size and the time step are pushed to the gradient and CFL limits, respectively, the time step may become large with respect to the mean collision time. The Fokker-Planck algorithm due to Gorji et al. [GJ14] described in Part I naturally conserves momentum and enforces energy conservation due to the correction factor. However, as time steps increase, the error introduced by the first-order integration of the position, i.e., the free-flight assumption, is expected to become significant. In order to improve the evolution of the position variance and position velocity covariance, an improved position integration scheme may be used. In the following, the original derivation of the scheme presented in [GTJ11] is slightly extended to include consistent treatment of the non-linear drift. As before, it is assumed that all coefficients remain constant in integrals over a time step.

29.1 Conservation of Energy during Velocity Update

In order to conserve energy, instead of using a correction factor as in Equation (91), a modified diffusion factor $\widetilde{D}_{\mathrm{tr}}^2$ is employed. The modified system of SDEs (74) for the velocities and positions reads

$$\mathrm{d}M_i = [-\nu M_i + N_i]\mathrm{d}t + \widetilde{D}_{\mathrm{tr}}\mathrm{d}W_i, \tag{211a}$$

$$\mathrm{d}\xi_i = \mu_i \mathrm{d}t, \tag{211b}$$

and the numerical velocity update Equation (85) becomes

$$M_i^{n+1} = e^{-\nu\Delta t} M_i^n + \frac{1}{\nu} e^{-\nu\Delta t}\left(1 - e^{-\nu\Delta t}\right) N_i^n + \sqrt{\frac{\widetilde{D}_{\mathrm{tr}}^2}{2\nu}(1 - e^{-2\nu\Delta t})}\,\psi_{\mathrm{tr},i}, \tag{212}$$

where, as before, $\psi_{\mathrm{tr},i}$ is a random number drawn from the standard normal distribution, i.e., $\psi_{\mathrm{tr},i} \sim \mathcal{N}(0,1)$.

To determine $\widetilde{D}_{\mathrm{tr}}^2$, the exact and numerical integration of translational thermal energy over a time step Δt are compared. Based on Equation (42c), the exact evolution equation for the trace of the stress tensor reads

$$\begin{aligned} \mathrm{d}\langle V_s V_s\rangle &= \Pi^{\mathrm{FP}}_{V_s V_s}\mathrm{d}t \\ &= 2\langle A_{\mathrm{tr}\,s} V_s\rangle \mathrm{d}t + 3\rho D_{\mathrm{tr}}^2 \mathrm{d}t \\ &= -2\nu\langle V_s V_s\rangle \mathrm{d}t + 3\rho D_{\mathrm{tr}}^2 \mathrm{d}t. \end{aligned} \tag{213}$$

Using the integrating factor $e^{2\nu t}$, the above may be integrated from time t at time step n to time $t+\Delta t$ at time step $n+1$, viz.

$$\langle V_s V_s\rangle(t+\Delta t) = e^{-2\nu\Delta t}\langle V_s V_s\rangle(t) + \frac{3\rho}{2\nu}D_{\mathrm{tr}}^2\Big(1-e^{-2\nu\Delta t}\Big). \tag{214}$$

From the velocity update due to the numerical scheme Equation (212), on the other hand, one obtains

$$\begin{aligned}\overline{M_s^{n+1}M_s^{n+1}} = e^{-2\nu\Delta t}\overline{M_s^n M_s^n} + \frac{2}{\nu}e^{-2\nu\Delta t}\Big(1-e^{-\nu\Delta t}\Big)\overline{M_s^n N_s^n} \\ + \frac{1}{\nu^2}e^{-2\nu\Delta t}\Big(1-e^{-\nu\Delta t}\Big)^2\overline{N_s^n N_s^n} + \frac{3\rho}{2\nu}\widetilde{D}_{\mathrm{tr}}^2\Big(1-e^{-2\nu\Delta t}\Big),\end{aligned} \tag{215}$$

where over-lined terms denote their weighted ensemble average, i.e.,

$$\overline{g(\zeta)} = \frac{1}{|\Omega|}\sum_{l=1}^{N} w^l g\big(c^l\big), \tag{216}$$

where $|\Omega|$ is the volume of the grid cell under consideration. Note that Equation (215) assumes that the vectors $\boldsymbol{\psi}^l$ satisfy $\overline{\psi_i\psi_j} = \rho\delta_{ij}$. Comparing coefficients in Eqns. (214) and (215), one finds

$$\begin{aligned}\frac{\widetilde{D}_{\mathrm{tr}}^2}{2\nu}\Big(1-e^{-2\nu\Delta t}\Big) = \frac{D_{\mathrm{tr}}^2}{2\nu}\Big(1-e^{-2\nu\Delta t}\Big) - \frac{2}{\nu}e^{-2\nu\Delta t}\Big(1-e^{-\nu\Delta t}\Big)\overline{M_s^n N_s^n}\Big/3\rho \\ - \frac{1}{\nu^2}e^{-2\nu\Delta t}\Big(1-e^{-\nu\Delta t}\Big)^2\overline{N_s^n N_s^n}\Big/3\rho.\end{aligned} \tag{217}$$

29.2 Position Update as Time Integral of the Velocity Process

In the following, the position integration scheme with increased accuracy is derived, i.e., for the position process $\mathrm{d}\boldsymbol{\xi} = \boldsymbol{\mu}\mathrm{d}t$, instead of the approximation $\boldsymbol{\mu}(t+s) \approx \boldsymbol{\mu}(t), 0 \le s \le \Delta t$, which leads to $\xi_i^{n+1} = \xi_i^n + \mu_i^n\Delta t$, the integration is carried out explicitly. Formally,

$$\xi_i(t+\Delta t) = \xi_i(t) + U_i\Delta t + \int_t^{t+\Delta t} M_i(s)\mathrm{d}s. \tag{218}$$

Recall the formal solution for $\boldsymbol{M}(t+\Delta t)$ (Equation (81)), repeated here with the modified diffusion:

$$M_i(t+\Delta t) = e^{-\nu\Delta t}M_i(t) + \int_t^{t+\Delta t} e^{\nu(s-t-\Delta t)}N_i(s)\mathrm{d}s + \int_t^{t+\Delta t} e^{\nu(s-t-\Delta t)}\widetilde{D}_{\mathrm{tr}}\mathrm{d}W_i(s). \tag{219}$$

The integral in Equation (218) follows as

$$\begin{aligned}\int_t^{t+\Delta t} M_i(s)\mathrm{d}s &= \int_0^{\Delta t} M_i(t+s')\mathrm{d}s' \\ &= \int_0^{\Delta t} e^{-\nu s'} M_i(t)\mathrm{d}s' + \overbrace{\int_0^{\Delta t}\int_t^{t+s'} e^{\nu(s-t-s')} N_i(s)\mathrm{d}s\mathrm{d}s'}^{I} \\ &\qquad + \underbrace{\int_0^{\Delta t} e^{-\nu s'}\int_t^{t+s'} e^{\nu(s-t)}\widetilde{D}_{\mathrm{tr}}\mathrm{d}W_i(s)\mathrm{d}s'}_{II} \\ &= \frac{1}{\nu}\Big(1-e^{\nu\Delta t}\Big)M_i(t) + I + II.\end{aligned} \tag{220}$$

As before, the term I is approximated by assuming $N_i(t+s) \approx N_i(t)$, viz.

$$I \approx \frac{1}{\nu^2}\Big(\nu\Delta t + e^{-\nu\Delta t} - 1\Big)N_i(t). \tag{221}$$

Following Equations (164)–(171) in [Cha43], term II is simplified by integrating by parts:

$$\begin{aligned}II &= \left[-\frac{1}{\nu}e^{-\nu s'}\int_t^{t+s'} e^{\nu(s-t)}\widetilde{D}_{\mathrm{tr}}\mathrm{d}W_i(s)\right]_{s'=0}^{\Delta t} \\ &\qquad + \int_0^{\Delta t}\frac{1}{\nu}e^{-\nu s'}\frac{\partial}{\partial s'}\left(\int_t^{t+s'} e^{\nu(s-t)}\widetilde{D}_{\mathrm{tr}}\mathrm{d}W_i(s)\right)\mathrm{d}s' \\ &= -\frac{1}{\nu}e^{-\nu\Delta t}\int_t^{t+\Delta t} e^{\nu(s-t)}\widetilde{D}_{\mathrm{tr}}\mathrm{d}W_i(s) + \int_0^{\Delta t}\frac{1}{\nu}e^{-\nu s'}e^{\nu(t+s'-t)}\widetilde{D}_{\mathrm{tr}}\mathrm{d}W_i(s') \\ &= \int_t^{t+\Delta t}\frac{1}{\nu}\Big(1-e^{\nu(s-t-\Delta t)}\Big)\widetilde{D}_{\mathrm{tr}}\mathrm{d}W_i(s),\end{aligned} \tag{222}$$

where the last step follows from $\int_0^{\Delta t}\mathrm{d}W_i(s')$ having the same distribution as $\int_t^{t+\Delta t}\mathrm{d}W_i(s)$. According to Equation (83),

$$II \sim \mathcal{N}\Big(0, B^2\Big), \quad \text{with} \quad B^2 = \frac{1}{2\nu^3}\Big(2\nu\Delta t + 4e^{-\nu\Delta t} - e^{-2\nu\Delta t} - 3\Big)\widetilde{D}_{\mathrm{tr}}^2. \tag{223}$$

Using Equation (223), the formal position evolution Equation (218) may be written as

$$\begin{aligned}\xi_i(t+\Delta t) &= \xi_i(t) + U_i\Delta t - \frac{1}{\nu}\Big(1-e^{-\nu\Delta t}\Big)M_i(t) \\ &\quad + \int_0^{\Delta t}\int_t^{t+s'} e^{\nu(s-t-s')}N_i(s)\mathrm{d}s\mathrm{d}s' + \int_t^{t+\Delta t}\frac{1}{\nu}\Big(1-e^{\nu(s-t-\Delta t)}\Big)\widetilde{D}_{\mathrm{tr}}\mathrm{d}W_i(s),\end{aligned} \tag{224}$$

which, together with Equations (221) and (223), leads us to propose the numerical position update

$$\begin{aligned}\xi_i^{n+1} &= \xi_i^n + U_i\Delta t + \frac{1}{\nu}\Big(1-e^{-\nu\Delta t}\Big)M_i^n \\ &\qquad + \frac{1}{\nu^2}e^{-\nu\Delta t}\Big(\nu\Delta t + e^{-\nu\Delta t} - 1\Big)N_i^n + B\,\psi_{\mathrm{x},i},\end{aligned} \tag{225}$$

with B as in Equation (223), and where $\psi_{\mathrm{x},i} \sim \mathcal{N}(0,1)$. As in Equation (85), the additional factor $e^{-\nu\Delta t}$ in the non-linear term ensures consistency with the fact that $\lim_{\Delta t\to\infty} N = 0$, while maintaining the exact evolution as $\Delta t \to 0$, i.e.,

$$\text{as } \Delta t \to 0: \quad \frac{1}{\nu^2} e^{-\nu\Delta t}\Big(\nu\Delta t + e^{-\nu\Delta t} - 1\Big) N_i \to \frac{\Delta t^2}{2} N_i. \tag{226}$$

The numerical integration schemes given by Equations (212) and (225) are statistically exact to the level of approximation of the non-linear term $\boldsymbol{N}$ with regards to the marginal distributions of velocity and displacement, since both the analytical displacement and the analytical velocity change follow Gaussian distributions, and the numerical schemes reproduce the desired means and variances.

29.3 The Correct Joint Statistics of Position and Velocity

To recover the full joint distribution of displacement and velocity change, the stochastic terms in Equations (212) and (225) need to be appropriately correlated. To this end, Equation (212) is modified to read as follows:

$$M_i^{n+1} = e^{-\nu\Delta t} M_i^n + \frac{1}{\nu} e^{-\nu\Delta t}\Big(1 - e^{-\nu\Delta t}\Big) N_i^n + \sqrt{A^2 - \frac{C^2}{B^2}}\, \psi_{\mathrm{tr},i} + \sqrt{\frac{C^2}{B^2}}\, \psi_{\mathrm{x},i}, \tag{227}$$

where $A^2 = \frac{1}{2\nu}\Big(1 - e^{-2\nu\Delta t}\Big)\widetilde{D}_{\mathrm{tr}}^2$, and $\boldsymbol{\psi}_{\mathrm{x}}$ is the same vector of samples of independent standard normal random numbers as used in the position update. Note that the statistical evolution of $\boldsymbol{M}$ implied by Equation (227) and the original scheme (212) are identical. To determine the new constant C, the exact update of the weighted position-velocity covariance is derived: Itō's Lemma leads to

$$\begin{aligned} \mathrm{d}(\xi_i M_j) &= M_i \mathrm{d}\xi + \xi_i \mathrm{d}M_j \\ &= (M_i U_j + M_i M_j)\mathrm{d}t + \xi_i(-\nu M_j + N_j)\mathrm{d}t + \xi_i \widetilde{D}_{\mathrm{tr}} \mathrm{d}W_j, \end{aligned} \tag{228}$$

where $\mathrm{d}\boldsymbol{\xi}$ and $\mathrm{d}\boldsymbol{M}$ are given by Equation (211). Applying the expectation operator to both sides, and assuming that position and the nonlinear acceleration are instantaneously uncorrelated, i.e., $\mathrm{E}[\xi_i N_j] \equiv 0$, yields

$$\mathrm{dE}[\xi_i M_j] = -\nu\, \mathrm{E}[\xi_i M_j]\mathrm{d}t + \mathrm{E}[M_i M_j]\mathrm{d}t. \tag{229}$$

The formal solution of the above is obtained using the integrating factor $e^{\nu\Delta t}$:

$$\mathrm{E}[\xi_i M_j](t+\Delta t) = e^{-\nu\Delta t}\, \mathrm{E}[\xi_i M_j](t) + e^{-\nu(t+\Delta t)} \int_t^{t+\Delta t} e^{\nu s}\, \mathrm{E}[M_i M_j](s)\mathrm{d}s. \tag{230}$$

To derive the evolution of $M_i M_j$, Itō's Lemma is invoked yet again, viz.

$$\mathrm{d}(M_i M_j) = M_i \mathrm{d}M_j + M_j \mathrm{d}M_i + \mathrm{d}M_i \mathrm{d}M_j, \tag{231}$$

with $\mathrm{d}\boldsymbol{M}$ as in Equation (211). Expanding terms and applying the expectation operator, and recalling that $\mathrm{d}W_i \mathrm{d}W_j = \delta_{ij}\mathrm{d}t$, one finds

$$\mathrm{dE}[M_i M_j] = -2\nu\,\mathrm{E}[M_i M_j]\mathrm{d}t + \delta_{ij}\widetilde{D}_{\mathrm{tr}}^2\mathrm{d}t, \tag{232}$$

which is readily integrated using the integrating factor $e^{2\nu t}$:

$$\mathrm{E}[M_i M_j](t+\Delta t) = e^{-2\nu\Delta t}\,\mathrm{E}[M_i M_j](t) + \delta_{ij}\frac{1}{2\nu}\Big(1 - e^{-2\nu\Delta t}\Big)\widetilde{D}_{\mathrm{tr}}^2. \tag{233}$$

Inserting the above into Equation (230), one obtains

$$\begin{aligned}\mathrm{E}[\xi_i M_j](t+\Delta t) &= e^{-\nu\Delta t}\,\mathrm{E}[\xi_i M_j](t) + e^{-\nu(t+\Delta t)}\int_0^{\Delta t} e^{\nu(t-s)}\,\mathrm{E}[M_i M_j](t)\mathrm{d}s \\ &\quad + e^{-\nu(t+\Delta t)}\int_0^{\Delta t}\delta_{ij}\frac{1}{2\nu}\Big(e^{\nu(t+s)} - e^{\nu(t-s)}\Big)\widetilde{D}_{\mathrm{tr}}^2\mathrm{d}s \\ &= e^{-\nu\Delta t}\,\mathrm{E}[\xi_i M_j](t) + \frac{1}{\nu}\Big(e^{-\nu\Delta t} - e^{-2\nu\Delta t}\Big)\,\mathrm{E}[M_i M_j](t) \\ &\quad + \delta_{ij}\frac{1}{2\nu^2}\Big(1 - e^{-\nu\Delta t}\Big)^2\widetilde{D}_{\mathrm{tr}}^2.\end{aligned} \tag{234}$$

The corresponding expression due to the proposed numerical integration scheme is obtained by multiplication of Equation (227) with Equation (225) and subsequent (weighted) averaging, viz.

$$\begin{aligned}\overline{\xi_i^{n+1} M_j^{n+1}} &= e^{-\nu\Delta t}\,\overline{\xi_i^n M_j^n} - \frac{1}{\nu}\Big(e^{-\nu\Delta t} - e^{-2\nu\Delta t}\Big)\overline{M_i^n M_j^n} \\ &\quad + \underbrace{\frac{1}{\nu^3}e^{-2\nu\Delta t}\Big(1 - e^{-\nu\Delta t}\Big)\Big(\nu\Delta t + e^{-\nu\Delta t} - 1\Big)}_{\mathcal{O}(\Delta t^3)\ \text{as}\ \Delta t\to 0}\,\overline{N_i^n N_j^n} + \delta_{ij}\rho\sqrt{C^2}.\end{aligned} \tag{235}$$

By comparing Equations (234) and (235), while neglecting the third-order contribution by the non-linear terms in the latter, one obtains

$$C^2 = \left(\frac{1}{2\nu^2}\Big(1 - e^{-\nu\Delta t}\Big)^2\widetilde{D}_{\mathrm{tr}}^2\right)^2. \tag{236}$$

In summary, the final result, the statistically exact (to the order of approximation of the non-linear term) numerical integration scheme for the positions $\boldsymbol{\xi}$ and velocities $\boldsymbol{M}$, reads as follows:

$$\begin{aligned}M_i^{n+1} &= e^{-\nu\Delta t}M_i^n + \frac{1}{\nu}e^{-\nu\Delta t}\Big(1 - e^{-\nu\Delta t}\Big)N_i^n \\ &\quad + \sqrt{A^2 - \frac{C^2}{B^2}}\,\psi_{\mathrm{tr},i} + \sqrt{\frac{C^2}{B^2}}\,\psi_{\mathrm{x},i}\end{aligned} \tag{237a}$$

and

$$\begin{aligned}\xi_i^{n+1} &= \xi_i^n + U_i\Delta t + \frac{1}{\nu}\Big(1 - e^{-\nu\Delta t}\Big)M_i^n \\ &\quad + \frac{1}{\nu^2}e^{-\nu\Delta t}\Big(\nu\Delta t + e^{-\nu\Delta t} - 1\Big)N_i^n + \sqrt{B^2}\,\psi_{\mathrm{x},i},\end{aligned} \tag{237b}$$

where

$$\widetilde{D}_{\mathrm{tr}}^2 = D_{\mathrm{tr}}^2 - \frac{1}{\nu^2} e^{-2\nu\Delta t}\left(1 - e^{-\nu\Delta t}\right)^2 \overline{N_s^n N_s^n} \Big/ \frac{3\rho}{2\nu}\left(1 - e^{-2\nu\Delta t}\right), \tag{237c}$$

$$A^2 = \frac{1}{2\nu}\left(1 - e^{-2\nu\Delta t}\right)\widetilde{D}_{\mathrm{tr}}^2, \tag{237d}$$

$$B^2 = \frac{1}{2\nu^3}\left(2\nu\Delta t + 4e^{-\nu\Delta t} - e^{-2\nu\Delta t} - 3\right)\widetilde{D}_{\mathrm{tr}}^2 \tag{237e}$$

and

$$C^2 = \left(\frac{1}{2\nu^2}\left(1 - e^{-\nu\Delta t}\right)^2 \widetilde{D}_{\mathrm{tr}}^2\right)^2. \tag{237f}$$

29.4 Implementation

Some observations regarding the implementation of the numerical scheme (237) are given in the following: First, note that per particle, three additional samples from the standard normal distribution are required. The implementation presented here uses the highly efficient counter-based random number generation library Random123 [Sal+11] to generate random integers, which serve as input to a custom implementation of the Ziggurat algorithm due to Marsaglia and Tsang [MT00] to sample the standard normal distribution. This procedure avoids the costly evaluation of trigonometric functions, and is efficient enough to render the impact of the additionally required normal variates on overall performance negligible.

A further technical point concerns the modified diffusion coefficient $\widetilde{D}_{\mathrm{tr}}^2$. From the definition of $\widetilde{D}_{\mathrm{tr}}^2$, Equation (237c), it is not guaranteed a priori that $\widetilde{D}_{\mathrm{tr}}^2$ is positive. Should a negative value for $\widetilde{D}_{\mathrm{tr}}^2$ occur during a simulation, the time step in the respective cell may be subdivided into equal parts while maintaining the values of the coefficients, until a positive value is obtained. Since for the negative term in the definition of $\widetilde{D}_{\mathrm{tr}}^2$,

$$\text{as } \Delta t \to 0: \quad \frac{1}{\nu^2} e^{-2\nu\Delta t}\left(1 - e^{-\nu\Delta t}\right)^2 \overline{N_s^n N_s^n} \Big/ \frac{3\rho}{2\nu}\left(1 - e^{-2\nu\Delta t}\right) \to \Delta t \frac{\overline{N_s^n N_s^n}}{3\rho}, \tag{238}$$

this procedure is guaranteed to find a positive value for $\widetilde{D}_{\mathrm{tr}}^2$ after a finite number of subdivisions.

The most severe implication of the accurate position integration scheme for the implementation, however, is that particle time-step trajectories are no longer straight lines. This is fundamentally at odds with the ray-tracing procedure outlined in Part II, Section 6.3 to find intersections of particle trajectories with the boundaries. The only way to circumvent this problem is to augment the numerical particle state by an "effective" velocity vector

$$\boldsymbol{\mu}_{\mathrm{eff}}^{n+1} = \frac{\boldsymbol{\xi}^{n+1} - \boldsymbol{\xi}^n}{\Delta t}, \tag{239}$$

and to base the particle movement algorithm on $\boldsymbol{\mu}_{\mathrm{eff}}$ instead of $\boldsymbol{\mu}$. As before, if a particle trajectory is found to intersect a boundary, the particle is moved to

the intersection point, and, after its state is modified according to the boundary condition, the particle travels along a straight line according to its new velocity for the remainder of the time step. In essence, this means that the higher-order position integration scheme outlined above becomes again first-order near boundaries. In the context of a locally refined mesh with local variable time stepping, however, the boundaries are usually resolved to a relatively high degree. This means that the local time step in the vicinity of boundaries is usually low, which should mitigate the error introduced by the linearized trajectories.

Part VI
Conclusion

The most prevalent numerical tool to study the dynamics of rarefied gases is the Direct Simulation Monte Carlo (DSMC) method. However, since it is based on the explicit calculation of binary collisions, its computational cost becomes intractable in the near-continuum regime. The Fokker-Planck (FP) method, on the other hand, relies on the time integration of continuous stochastic processes, and is both accurate and efficient in the near continuum. Since both methods share the same fundamental structure, i.e., they are both stochastic particle methods, coupling the two is straightforward. The resulting FP-DSMC algorithm is capable of simulating flows of dilute gas ranging from the near-continuum to the fully rarefied regime. This thesis is concerned with the analysis and parallel implementation of the Fokker-Planck-DSMC algorithm.

30 Summary and Conclusions

After giving some background on rarefied gas dynamics and the Fokker-Planck method in particular, the general purpose, parallel implementation of the coupled FP-DSMC algorithm is presented. The implementation exploits both coarse-grained and fine-grained parallelism via distributed- and shared-memory multiprocessor programming, and is thus optimally suited for state-of-the-art high-performance computer cluster technologies. The external ray-tracing library Embree is used to deal efficiently with complex geometries. Several improvements to algorithms and data structures for particle-based simulations are introduced, which are relevant beyond the scope of the FP-DSMC method itself. The new algorithm is applied to various three-dimensional test cases. It is demonstrated that accurate, efficient simulations of gas flows in and around complex geometries, covering the entire range of Knudsen numbers, are possible using FP-DSMC.

In a next step, it is shown how ordering the grid and particle data structures according to locally computable space-filling curves (SFCs) allows to elegantly solve many challenges with regards to performance of particle simulations, especially in the context of hybrid shared-distributed memory computer architectures: the presented algorithms ensure cache-friendly computations, efficient inter-process communication, and parallel load balancing both at the fine- and coarse-grained parallelism level. The SFC-ordered data structures allow for efficient automatic mesh refinement, which is of particular relevance for the (FP-)DSMC algorithm. Additionally, a detailed explanation, as well as source code, for the fast computation of the relevant SFC indices, based on bit-wise operations and a look-up table approach, are provided. The modular simulation architecture allows to study the performance impact of using different SFCs for relevant rarefied gas dynamics test cases.

Motivated by the goal to perform automatic mesh refinement based on the gradients of macroscopic quantities, the theoretical foundation for a general approach to compute estimates of spatial derivatives of statistics—such as generalized moments—of scattered data was developed. The kernel-based method allows for the direct computation of these quantities as weighted sums of given data, without having to compute finite differences of the statistics. This "local" nature of the approach makes it simple to implement, as well as ideally suited for parallel algorithms based on domain decomposition and locally refined computational grids. Explicit expressions for kernel functions to estimate mixed spatial derivatives of arbitrary order in arbitrary dimensions are derived, and it is shown that the kernel approach trivially extends to kernel interpolation of mixed spatial derivatives of deterministic functions of position. The suggested approach may prove useful for any computational task where estimates of spatial derivatives of scattered data could be of interest: from molecular dynamics, smoothed particle hydrodynamics, particle-in-cell, DSMC, and other particle-based physics simulation algorithms, to spatial econometrics, computer vision, and machine intelligence.

The final steps towards maximally efficient parallel Fokker-Planck-DSMC simulations on gradient-capturing meshes are the implementation of the kernel method, the use of local variable time steps to cope with the large scale-variations and hence particle resolution imbalance, as well as a position integration scheme with improved accuracy for large time steps. To this end, the necessary theory and implementation of the above are discussed.

31 Outlook

"Now, here, you see, it takes all the running you can do, to keep in the same place. If you want to get somewhere else, you must run at least twice as fast as that!"

—Lewis Carroll, *Through the Looking-Glass*

Software for scientific numerical simulation is never "finished"; instead, it is the author's hope that the developments described in this thesis may lay the foundation for the pursuit of future research.

A worthwhile next area of investigation after the completion of the final implementation steps mentioned above would be to study different dynamic switching criteria for the FP and DSMC collision operators. The current approach is well suited for maximum performance, however, other groups have suggested using various equilibrium breakdown parameters, which may offer better guarantees on accuracy. Performing a comparison of different switching criteria would require minimal additions to the current implementation.

While there may still be room for algorithmic improvements, the presented simulation code was demonstrated to efficiently produce results for meaningful problem sizes. Incorporating next-generation computing hardware seems to be more promising: since the Fokker-Planck algorithm is inherently parallel on the

particle level, it would be worthwhile to extend the implementation to utilize general purpose graphics processing units (GPUs), which may offer large performance gains.

To expand the simulation capabilities, the focus of further development should be on the modeling side: after implementing and validating the recent Entropic Fokker-Planck model, its extension to mixtures of polyatomic gases may be pursued. Further, a consistent treatment of quantized internal degrees of freedom seems necessary. To perform realistic numerical simulations of reentry-type flows, also dissociation, recombination, ionization, and potentially radiation effects need to be considered.

The software presented in this thesis is meant to be released "open source" in the near future, in the hope that it may prove useful in accommodating such developments.

Part VII
Appendices

A Appendix to Part I

A.1 System of Constitutive Equations

The macroscopic coefficients $\tilde{c}_{i,j}$ and γ_i in the cubic Fokker-Planck algorithm are determined by requiring identical relaxation rates of phase density moments up to third order (heat fluxes) to those implied by the Boltzmann collision operator. The corresponding constitutive equations (63b) and (66) form a linear system for the coefficients, viz.

$$\begin{pmatrix} L_c & L_\gamma \\ M_c & M_\gamma \end{pmatrix} \begin{pmatrix} \tilde{\boldsymbol{c}} \\ \boldsymbol{\gamma} \end{pmatrix} = \begin{pmatrix} \boldsymbol{z} \\ \boldsymbol{r} \end{pmatrix}, \tag{240}$$

where $\tilde{\boldsymbol{c}} = \begin{pmatrix} \tilde{c}_{11} & \tilde{c}_{12} & \tilde{c}_{13} & \tilde{c}_{22} & \tilde{c}_{23} & \tilde{c}_{33} \end{pmatrix}^\top$ denotes the vector of unknowns in the symmetric coefficient matrix $\tilde{c}_{ij}$ and the matrices $L_c \in \mathbb{R}^{6\times 6}$, $L_\gamma \in \mathbb{R}^{6\times 3}$, $M_c \in \mathbb{R}^{3\times 6}$, and $M_\gamma \in \mathbb{R}^{3\times 3}$, as well as the vectors $\boldsymbol{z} \in \mathbb{R}^6$ and $\boldsymbol{r} \in \mathbb{R}^3$, are defined below.

Recall the definition of the weighted moments (Eq. (50)),

$$u^s_{i_1\dots i_k} = \int \|\boldsymbol{V}'\|^s V'_{i_1} \cdots V'_{i_k} f(\cdot, \boldsymbol{c}') \mathrm{d}\boldsymbol{c}'. \tag{241}$$

Note also that $u^2/\rho \equiv 3\theta$, and consequently, $\sigma_{ij} \equiv u^0_{ij} - \rho\theta\delta_{ij}$. The matrices L_c, L_γ, M_c, and M_γ follow as:

$$L_c = \begin{pmatrix} 2u^0_{11} & 2u^0_{12} & 2u^0_{13} & 0 & 0 & 0 \\ u^0_{12} & u^0_{11}+u^0_{22} & u^0_{23} & u^0_{12} & u^0_{13} & 0 \\ u^0_{13} & u^0_{23} & u^0_{11}+u^0_{33} & 0 & u^0_{12} & u^0_{13} \\ 0 & 2u^0_{12} & 0 & 2u^0_{22} & 2u^0_{23} & 0 \\ 0 & u^0_{13} & u^0_{12} & u^0_{23} & u^0_{22}+u^0_{33} & u^0_{23} \\ 0 & 0 & 2u^0_{13} & 0 & 2u^0_{23} & 2u^0_{33} \end{pmatrix}, \tag{242}$$

$$L_\gamma = \begin{pmatrix} 2u^2_1 & 0 & 0 \\ u^2_2 & u^2_1 & 0 \\ u^2_3 & 0 & u^2_1 \\ 0 & 2u^2_2 & 0 \\ 0 & u^2_3 & u^2_2 \\ 0 & 0 & 2u^2_3 \end{pmatrix}, \tag{243}$$

$$M_c = \begin{pmatrix} u^2_1 & u^2_2 & u^2_3 & 0 & 0 & 0 \\ 0 & u^2_1 & 0 & u^2_2 & u^2_3 & 0 \\ 0 & 0 & u^2_1 & 0 & u^2_2 & u^2_3 \end{pmatrix} + 2\begin{pmatrix} u^0_{111} & 2u^0_{112} & 2u^0_{113} & u^0_{122} & 2u^0_{123} & u^0_{133} \\ u^0_{112} & 2u^0_{122} & 2u^0_{123} & u^0_{222} & 2u^0_{223} & u^0_{233} \\ u^0_{113} & 2u^0_{123} & 2u^0_{133} & u^0_{223} & 2u^0_{233} & u^0_{333} \end{pmatrix}, \tag{244}$$

and

$$M_\gamma = \begin{pmatrix} u^4 - 3\theta u^2 & 0 & 0 \\ 0 & u^4 - 3\theta u^2 & 0 \\ 0 & 0 & u^4 - 3\theta u^2 \end{pmatrix} + 2\begin{pmatrix} u_{11}^2 - 3\theta u_{11}^0 & u_{12}^2 - 3\theta u_{12}^0 & u_{13}^2 - 3\theta u_{13}^0 \\ u_{12}^2 - 3\theta u_{12}^0 & u_{22}^2 - 3\theta u_{22}^0 & u_{23}^2 - 3\theta u_{23}^0 \\ u_{13}^2 - 3\theta u_{13}^0 & u_{23}^2 - 3\theta u_{23}^0 & u_{33}^2 - 3\theta u_{33}^0 \end{pmatrix}. \tag{245}$$

The right hand side vectors $\boldsymbol{z}$ and $\boldsymbol{r}$ are given by

$$\begin{aligned} \boldsymbol{z}^\top = \Big(\Pi^\star_{V_{\langle 1}V_{1\rangle}} \quad \Pi^\star_{V_{\langle 1}V_{2\rangle}} \quad \Pi^\star_{V_{\langle 1}V_{3\rangle}} \quad \Pi^\star_{V_{\langle 2}V_{2\rangle}} \quad \Pi^\star_{V_{\langle 2}V_{3\rangle}} \quad \Pi^\star_{V_{\langle 3}V_{3\rangle}}\Big) \\ + 2\nu\Big(u_{11}^0 - \rho\theta \quad u_{12}^0 \quad u_{13}^0 \quad u_{22}^0 - \rho\theta \quad u_{23}^0 \quad u_{33}^0 - \rho\theta\Big) \\ - 2\Lambda\Big(u_{11}^2 \quad u_{12}^2 \quad u_{13}^2 \quad u_{22}^2 \quad u_{23}^2 \quad u_{33}^2\Big) \end{aligned} \tag{246}$$

and

$$r_i = \Pi^*_{\|\boldsymbol{V}\|^2 V_i} + 3\nu u_i^2 - 3\Lambda\Big(u_i^4 - \theta u_i^2\Big) + \frac{2\Lambda}{\rho}\Big(u_{i1}^0 u_1^2 + u_{i2}^0 u_2^2 + u_{i3}^0 u_3^2\Big), \tag{247}$$

where $\nu = p/2\mu$, the coefficient Λ is defined in Eq. (49), and the Boltzmann production terms $\Pi^\star_{V_{\langle i}V_{j\rangle}}$ and $\Pi^*_{\|\boldsymbol{V}\|^2 V_i}$ are as in Eq. (41) and Eq. (67), respectively.

B Appendix to Part II

B.1 Tables of Simulation Parameters

Table 3: Simulation parameters for the supersonic corner flow simulations (Part II, Section 8.1).

geometry			
domain size (x, y, z)		[m]	$(0.3, 0.18, 0.18)$
corner profile	rectangle (x, y, z)	[m]	$(0.05 \dots 0.3, 0.0 \dots 0.18, 0.0)$
	rectangle (x, y, z)	[m]	$(0.05 \dots 0.3, 0.0, 0.0 \dots 0.18)$
bottom	rectangle (x, y, z)	[m]	$(0.0 \dots 0.05, 0.0 \dots 0.18, 0.0)$
front	rectangle (x, y, z)	[m]	$(0.0 \dots 0.05, 0.0, 0.0 \dots 0.18)$
top	rectangle (x, y, z)	[m]	$(0.0 \dots 0.3, 0.0 \dots 0.18, 0.18)$
back	rectangle (x, y, z)	[m]	$(0.0 \dots 0.3, 0.18, 0.0 \dots 0.18)$
left	rectangle (x, y, z)	[m]	$(0.0, 0.0 \dots 0.18, 0.0 \dots 0.18)$
right	rectangle (x, y, z)	[m]	$(0.3, 0.0 \dots 0.18, 0.0 \dots 0.18)$
discretization			
time step		[s]	$2.0 \times 10^{-6} \dots 5.0 \times 10^{-7}$
cell width		[m]	$0.01 \dots 0.0025$
simulation			
total number of time steps		[#]	15 000
number of sampling time steps		[#]	10 000
average number of particles per cell		[#]	≈ 100
gas model (Argon)			
potential			VHS
molecular mass		[kg]	6.63×10^{-26}
reference diameter		[m]	4.17×10^{-10}
reference temperature		[K]	273.0
viscosity exponent ω		[−]	0.81
ratio of specific heats γ		[−]	1.67
inflow			
number density		$[m^{-3}]$	1.0×10^{21}
temperature		[K]	300.0
Mach number		[−]	6.0
boundary conditions			
corner profile	type		fully diffusive
	temperature	[K]	1000.0
bottom and front	type		specular
other	type		stream

Table 4: Simulation parameters for the simulations of supersonic flow around a sphere (Part II, Section 8.2).

geometry			
domain size (x, y, z)		[m]	$(1.0, 1.0, 1.0)$
sphere	radius	[m]	0.1
	center (x, y, z)	[m]	$(0.0, 0.0, 0.0)$
bottom	rectangle (x, y, z)	[m]	$(-0.5 \dots 0.5, -0.5 \dots 0.5, -0.5)$
top	rectangle (x, y, z)	[m]	$(-0.5 \dots 0.5, -0.5 \dots 0.5, 0.5)$
front	rectangle (x, y, z)	[m]	$(-0.5 \dots 0.5, -0.5, -0.5 \dots 0.5)$
back	rectangle (x, y, z)	[m]	$(-0.5 \dots 0.5, 0.5, -0.5 \dots 0.5)$
left	rectangle (x, y, z)	[m]	$(-0.5, -0.5 \dots 0.5, -0.5 \dots 0.5)$
right	rectangle (x, y, z)	[m]	$(0.5, -0.5 \dots 0.5, -0.5 \dots 0.5)$
discretization			
time step		[s]	$7.576 \times 10^{-6} \dots 3.788 \times 10^{-6}$
cell width		[m]	$0.03 \dots 0.015$
simulation			
total number of time steps		[#]	12 500
number of sampling time steps		[#]	10 000
average number of particles per cell		[#]	≈ 100
gas model (Argon)			
potential			VHS
molecular mass		[kg]	6.63×10^{-26}
reference diameter		[m]	4.17×10^{-10}
reference temperature		[K]	273.0
viscosity exponent ω		[−]	0.81
ratio of specific heats γ		[−]	1.67
inflow			
number density		$[m^{-3}]$	5.0×10^{20}
temperature		[K]	300.0
Mach number		[−]	6.0
boundary conditions			
sphere	type		fully diffusive
	temperature	[K]	1000.0
other	type		stream

Table 5: Simulation parameters for the simulations of micro-nozzle flow (Part II, Section 8.3).

geometry			
domain size (x, y, z)		[mm]	$(7.5, 2.0, 0.15)$
nozzle & outlet	CAD model		see Figure 21
inlet	rectangle (x, y, z)	[mm]	$(0.0, 0.0 \ldots 0.15, 0.15 \ldots 0.3)$
y-symmetry	rectangle (x, y, z)	[mm]	$(0.0 \ldots 7.5, 0.0, 0.15 \ldots 0.3)$
z-symmetry	rectangle (x, y, z)	[mm]	$(0.0 \ldots 7.5, 0.0 \ldots 2.0, 0.15)$
discretization			
time step		[s]	2.0×10^{-9}
grid dimensions (n_x, n_y, n_z)		[−]	$(450, 120, 9) \ldots (1050, 280, 21)$
cell width		[mm]	$1.67 \times 10^{-2} \ldots 7.14 \times 10^{-3}$
simulation			
total number of time steps		[#]	100 000
number of sampling time steps		[#]	20 000
number of particles in simulation		[#]	$\approx 1.0 \times 10^6 \ldots \approx 1.0 \times 10^8$
gas model (Nitrogen)			
potential			VSS
molecular mass		[kg]	4.65×10^{-26}
reference viscosity		[Pa s^{-1}]	1.656×10^{-5}
reference temperature		[K]	273.0
scattering angle exponent α		[−]	1.36
viscosity exponent ω		[−]	0.74
Z_{rot}^{∞}		[−]	18
$T_{\text{rot}}^{\text{ref}}$ $(= T^*)$		[K]	91.5
C_1			9.1
C_2			220.0
$T_{\text{vib}}^{\text{ref}}$ (defines ν_0)		[K]	3340.0
ratio of specific heats γ		[−]	1.4
inflow			
pressure		[Pa]	5282.8
temperature		[K]	250.0
Mach number		[−]	1.0
boundary conditions			
nozzle	type		fully diffusive
	temperature	[K]	300.0

C Appendix to Part III

C.1 C++ Code to Generate Morton and Hilbert Codes

```
// "Insert" two 0 bit after each of the 21 low bits of x
uint64_t part1By2(uint64_t x) {
   x &= 0x00000000001fffff;
   x = (x ^ (x << 32))  & 0x001f00000000ffff;
   x = (x ^ (x << 16))  & 0x001f0000ff0000ff;
   x = (x ^ (x <<  8))  & 0x100f00f00f00f00f;
   x = (x ^ (x <<  4))  & 0x10c30c30c30c30c3;
   x = (x ^ (x <<  2))  & 0x1249249249249249;
   return x; }

// given indices in N3 i,j,k in range 0...2^21-1 = 2'097'151, compute Morton index
uint64_t mortonFromN3(const uint64_t i, const uint64_t j, const uint64_t k) {
   return (part1By2(i) << 2) ^ (part1By2(j) << 1) ^ part1By2(k);  }

// given Morton index m, compute Hilbert index h
uint64_t morton2Hilbert(uint64_t m) {
  uint64_t h     = 0;
  uint8_t  state = start_state;
  for (uint8_t shift = 60; shift != uint8_t(-3); shift -= 3){ // for each three bit digit
    const uint8_t orthant = (m >> shift) & 7;           // get orthant (Morton digit)
    h    ^= uint64_t(keyLUT[state][orthant]) << shift;  // save Hilbert digit
    state = stateLUT[state][orthant]); }                // state transition
  return h; }
```

C.2 Look-Up Tables for Selected Hilbert Curves

C.2.1 The Curve "Butz"

The base order of the "Butz" curve corresponds to state 3, and repeats after three consecutive visits of the zeroth octant.

2	1	3	0	5	6	4	7
0	1	7	6	3	2	4	5
2	3	5	4	1	0	6	7
0	7	3	4	1	6	2	5
2	5	1	6	3	4	0	7
0	3	1	2	7	4	6	5
4	5	3	2	7	6	0	1
4	7	5	6	3	0	2	1
4	3	7	0	5	2	6	1
6	5	7	4	1	2	0	3
6	7	1	0	5	4	2	3
6	1	5	2	7	0	4	3

Figure 41: Table of function H (Equation (102)) to generate the Hilbert code on curve "Butz" from the Morton code. If the encoding is in state i and the next 3-bit Morton digit is j, the entry at row i, column j specifies the next 3-bit digit of the Hilbert code.

8	8	5	10	4	4	5	2
3	5	8	9	2	5	2	9
7	1	0	1	7	11	0	4
5	7	4	4	1	10	1	10
6	2	6	2	3	3	9	0
1	0	3	3	6	0	11	11
10	5	10	9	11	5	4	9
9	10	3	3	9	2	11	11
11	11	9	0	1	10	1	10
8	8	1	7	4	4	6	7
7	3	0	8	7	6	0	6
6	2	6	2	5	7	8	8

Figure 42: State transition table (function S, Eq. (102)) to generate the Hilbert code on curve "Butz" from the Morton code. If the encoding is in state i and the next 3-bit Morton digit is j, the table entry at row i, column j specifies the next state.

C.2.2 The Curve "Ca00.chI"

The base order of the curve "Ca00.chI" corresponds to state 4, and repeats after three consecutive visits of the zeroth octant.

2	3	5	4	1	0	6	7
2	1	3	0	5	6	4	7
2	5	1	6	3	4	0	7
0	1	7	6	3	2	4	5
0	7	3	4	1	6	2	5
0	3	1	2	7	4	6	5
4	3	7	0	5	2	6	1
4	5	3	2	7	6	0	1
4	7	5	6	3	0	2	1
6	5	7	4	1	2	0	3
6	7	1	0	5	4	2	3
6	1	5	2	7	0	4	3

Figure 43: Table of function H (Equation (102)) to generate the Hilbert code on curve "Ca00.chI" from the Morton code. If the encoding is in state i and the next three-bit Morton digit is j, the entry at row i and column j specifies the next 3-bit digit of the Hilbert code.

0	9	0	5	8	11	1	2
1	6	11	10	1	2	4	0
2	2	7	0	10	3	9	1
4	5	6	9	1	3	8	3
5	8	0	7	3	10	4	4
3	2	4	5	7	6	11	5
0	7	9	1	6	6	3	10
1	7	8	7	11	5	2	9
2	10	8	4	6	0	8	11
6	9	3	11	2	9	7	4
8	4	1	6	10	9	10	5
7	0	11	11	5	8	10	3

Figure 44: State transition table (function S, Equation (102)) to generate the Hilbert code on curve "Ca00.chI" from the Morton code. If the encoding is in state i and the next three-bit Morton digit is j, the table entry at row i and column j specifies the next state.

C.3 Tables of Simulation Parameters

Table 6: Parameters for simulation of reentry flow over 70 degree blunted cone (Part III, Section 14).

geometry			
domain size (x, y, z)		[m]	$(0.14, 0.14, 0.14)$
simulation			
time step size		[s]	5.44×10^{-7}
total number of time steps			2500
number of sampling time steps			1500
average number of particles per cell			≈ 2000
mesh refinement			
initial number of grid cells $(\mathrm{n_x}, \mathrm{n_y}, \mathrm{n_z})$			$(32, 32, 32)$ (level 4)
pre-refinement			at geometry, to level 2
dynamic adaptation	at time steps no.		$50, 100, \ldots, 1000$
	cell split criterion		cellwidth $> 2.0 \times \lambda_0$
gas model (Nitrogen)			
potential			VHS
molecular mass		[kg]	4.65×10^{-26}
reference diameter		[m]	4.17×10^{-10}
reference temperature		[K]	273.0
viscosity exponent ω		[−]	0.75
$Z_{\mathrm{rot}}^{\infty}$		[−]	18.1
$T_{\mathrm{rot}}^{\mathrm{ref}}$ $(= T^*)$		[K]	91.5
C_1			9.1
C_2			220.0
$T_{\mathrm{vib}}^{\mathrm{ref}}$ (defines ν_0)		[K]	3340.0
ratio of specific heats γ		[−]	1.4
inflow			
number density		$[\mathrm{m}^{-3}]$	3.72×10^{20}
temperature		[K]	13.3
Mach number		[−]	20.2
boundary conditions			
cone geometry	type		fully diffusive
	temperature	[K]	300.0
other	type		stream

Table 7: Parameters for simulation of nozzle flow (Part III, Section 15).

geometry			
domain size (x, y, z)		[m]	$(0.05, 0.05, 0.17)$
simulation			
time step size		[s]	2.5×10^{-7}
total number of time steps			16 000
number of sampling time steps			4000
approx. number of particles in steady state			$\approx 208 \times 10^6$
approx. number of grid cells in steady state			$\approx 275 \times 10^3$
mesh refinement			
initial number of grid cells $(\mathrm{n_x}, \mathrm{n_y}, \mathrm{n_z})$			$(20, 20, 74)$ at level 4
pre-refinement			at geometry, to level 0
			in cylinder on z-axis ($r = 5\,\mathrm{mm}$), to level 1
gas model (Argon)			
potential			VHS
molecular mass		[kg]	6.63×10^{-26}
reference diameter		[m]	4.17×10^{-10}
reference temperature		[K]	273.0
viscosity exponent ω		[−]	0.81
ratio of specific heats γ		[−]	1.67
inflow			
pressure		[Pa]	2875.5
temperature		[K]	288.1
Mach number		[−]	0.226
boundary conditions			
walls	type		fully diffusive
	temperature	[K]	293.0
background	type		stream
	temperature	[K]	293.0
	pressure	[Pa]	40.0
	Mach number	[−]	0
outflow	type		stream
	temperature	[K]	293.0
	pressure	[Pa]	0.1
	Mach number	[−]	0

D Appendix to Part IV

D.1 Some Minimum Variance Kernels

Table 8: Minimum Variance Kernels in $d = 1$ with support on $[-1, 1]$ according to Eq. (150).

$\nu=\lvert\boldsymbol{\alpha}\rvert$	k	$\boldsymbol{\alpha}$	$\mathrm{K}_{\boldsymbol{\alpha}}(\boldsymbol{x})$	$\mathrm{AB}\left(\widehat{\partial^{\boldsymbol{\alpha}} \mathrm{f}}\right)$	$\mathrm{R}(\mathrm{K}_{\boldsymbol{\alpha}})$
0	2	(0)	$\frac{1}{2}$	$\frac{f^{*}}{6}$	0.5
	4	(0)	$-\frac{3}{8}\left(5\,x_1^2-3\right)$	$-\frac{f^{(4)}}{280}$	1.125
1	3	(1)	$-\frac{3\,x_1}{2}$	$\frac{f^{(3)}}{10}$	1.5
	5	(1)	$\frac{15}{8}\,x_1\left(7\,x_1^2-5\right)$	$-\frac{f^{(5)}}{504}$	9.375
2	4	(2)	$\frac{15}{4}\left(3\,x_1^2-1\right)$	$\frac{f^{(4)}}{14}$	22.5
	6	(2)	$-\frac{105}{32}\left(45\,x_1^4-42\,x_1^2+5\right)$	$-\frac{f^{(6)}}{792}$	275.625
3	5	(3)	$-\frac{105}{4}\,x_1\left(5\,x_1^2-3\right)$	$\frac{f^{(5)}}{18}$	787.5
	7	(3)	$\frac{945}{32}\,x_1\left(77\,x_1^4-90\,x_1^2+21\right)$	$-\frac{f^{(7)}}{1144}$	15946.9
4	6	(4)	$\frac{945}{16}\left(35\,x_1^4-30\,x_1^2+3\right)$	$\frac{f^{(6)}}{22}$	49612.5
	8	(4)	$-\frac{10\,395}{64}\left(273\,x_1^6-385\,x_1^4+135\,x_1^2-7\right)$	$-\frac{f^{(8)}}{1560}$	1500778.

Table 9: Minimum Variance Kernels in $d = 2$ with support on $[-1, 1]^2$ according to Eq. (150).

$\nu=\lvert\boldsymbol{\alpha}\rvert$	k	$\boldsymbol{\alpha}$	$\mathrm{K}_{\boldsymbol{\alpha}}(\boldsymbol{x})$	$\mathrm{AB}\left(\widehat{\partial^{\boldsymbol{\alpha}} \mathrm{f}}\right)$	$\mathrm{R}(\mathrm{K}_{\boldsymbol{\alpha}})$
0	2	(0 0)	$\frac{1}{4}$	$\frac{1}{6}\left(f^{(0,2)}+f^{(2,0)}\right)$	0.25
	4	(0 0)	$\frac{1}{16}\left(-15\,x_1^2-15\,x_2^2+14\right)$	$\frac{-9\,f^{(0,4)}-70\,f^{(2,2)}-9\,f^{(4,0)}}{2520}$	0.875
1	3	(1 0)	$-\frac{3\,x_1}{4}$	$\frac{1}{30}\left(5\,f^{(1,2)}+3\,f^{(3,0)}\right)$	0.75
	5	(1 0)	$\frac{15}{16}\,x_1\left(7\,x_1^2+3\,x_2^2-6\right)$	$\frac{-9\,f^{(1,4)}-42\,f^{(3,2)}-5\,f^{(5,0)}}{2520}$	5.625
2	4	(2 0)	$\frac{15}{8}\left(3\,x_1^2-1\right)$	$\frac{1}{42}\left(7\,f^{(2,2)}+3\,f^{(4,0)}\right)$	11.25
		(1 1)	$\frac{9\,x_1\,x_2}{4}$	$\frac{1}{10}\left(f^{(1,3)}+f^{(3,1)}\right)$	2.25
	6	(2 0)	$-\frac{45}{64}\left(105\,x_1^4+30\,x_2^2\,x_1^2-108\,x_1^2-10\,x_2^2+15\right)$	$\frac{-99\,f^{(2,4)}-330\,f^{(4,2)}-35\,f^{(6,0)}}{27\,720}$	151.875
		(1 1)	$-\frac{9}{16}\,x_1\,x_2\left(35\,x_1^2+35\,x_2^2-46\right)$	$\frac{-25\,f^{(1,5)}-126\,f^{(3,3)}-25\,f^{(5,1)}}{12\,600}$	25.875
3	5	(3 0)	$-\frac{105}{8}\,x_1\left(5\,x_1^2-3\right)$	$\frac{1}{18}\left(3\,f^{(3,2)}+f^{(5,0)}\right)$	393.75
		(2 1)	$-\frac{45}{8}\left(3\,x_1^2-1\right)x_2$	$\frac{1}{70}\left(7\,f^{(2,3)}+5\,f^{(4,1)}\right)$	33.75
	7	(3 0)	$\frac{105}{64}\,x_1\left(693\,x_1^4+150\,x_2^2\,x_1^2-860\,x_1^2-90\,x_2^2+219\right)$	$\frac{-3861\,f^{(3,4)}-10\,010\,f^{(5,2)}-945\,f^{(7,0)}}{1\,081\,080}$	8465.62
		(2 1)	$\frac{315}{64}\,x_2\left(45\,x_1^4+30\,x_2^2\,x_1^2-60\,x_1^2-10\,x_2^2+11\right)$	$\frac{-55\,f^{(2,5)}-198\,f^{(4,3)}-35\,f^{(6,1)}}{27\,720}$	590.625
4	6	(4 0)	$\frac{945}{32}\left(35\,x_1^4-30\,x_1^2+3\right)$	$\frac{1}{66}\left(11\,f^{(4,2)}+3\,f^{(6,0)}\right)$	24806.3
		(3 1)	$\frac{315}{8}\,x_1\left(5\,x_1^2-3\right)x_2$	$\frac{1}{90}\left(9\,f^{(3,3)}+5\,f^{(5,1)}\right)$	1181.25
		(2 2)	$\frac{225}{16}\left(3\,x_1^2-1\right)\left(3\,x_2^2-1\right)$	$\frac{1}{14}\left(f^{(2,4)}+f^{(4,2)}\right)$	506.25
	8	(4 0)	$-\frac{945}{128}\left(3003\,x_1^6+525\,x_2^2\,x_1^4-\right.$ $\left.4410\,x_1^4-450\,x_2^2\,x_1^2+1635\,x_1^2+45\,x_2^2-92\right)$	$\frac{-429\,f^{(4,4)}-910\,f^{(6,2)}-77\,f^{(8,0)}}{120\,120}$	781397.
		(3 1)	$-\frac{315}{64}\,x_1\,x_2\left(693\,x_1^4+350\,x_2^2\,x_1^2-1020\,x_1^2-210\,x_2^2+315\right)$	$\frac{-715\,f^{(3,5)}-2002\,f^{(5,3)}-315\,f^{(7,1)}}{360\,360}$	30121.9
		(2 2)	$-\frac{225}{128}\left(945\,x_2^2\,x_1^4-315\,x_1^4+945\,x_2^4\,x_1^2-\right.$ $\left.1692\,x_2^2\,x_1^2+375\,x_1^2-315\,x_2^4+375\,x_2^2-62\right)$	$\frac{-49\,f^{(2,6)}-198\,f^{(4,4)}-49\,f^{(6,2)}}{38\,808}$	11896.9

Table 10: Minimum Variance Kernels in $d = 3$ with support on $[-1, 1]^3$ according to Eq. (150).

$\nu=\lvert\boldsymbol{\alpha}\rvert$	k	$\boldsymbol{\alpha}$	$K_{\boldsymbol{\alpha}}(\boldsymbol{x})$	$AB\left(\widehat{\partial^{\boldsymbol{\alpha}} f}\right)$	$R(K_{\boldsymbol{\alpha}})$
0	2	(000)	$\frac{1}{8}$	$\frac{1}{6}\left(f^{(0,0,2)}+f^{(0,2,0)}+f^{(2,0,0)}\right)$	0.125
	4	(000)	$\frac{1}{32}\left(-15\,x_1^2-15\,x_2^2-15\,x_3^2+19\right)$	$\frac{1}{2520}\left(-9\,f^{(0,0,4)}-70\,f^{(0,2,2)}-9\,f^{(0,4,0)}-70\,f^{(2,0,2)}-70\,f^{(2,2,0)}-9\,f^{(4,0,0)}\right)$	0.59375
1	3	(100)	$-\frac{3\,x_1}{8}$	$\frac{1}{30}\left(5\,f^{(1,0,2)}+5\,f^{(1,2,0)}+3\,f^{(3,0,0)}\right)$	0.375
	5	(100)	$\frac{15}{32}\,x_1\left(7\,x_1^2+3\,x_2^2+3\,x_3^2-7\right)$	$\frac{1}{2520}\left(-9\,f^{(1,0,4)}-70\,f^{(1,2,2)}-9\,f^{(1,4,0)}-42\,f^{(3,0,2)}-42\,f^{(3,2,0)}-5\,f^{(5,0,0)}\right)$	3.28125
2	4	(200)	$\frac{15}{16}\left(3\,x_1^2-1\right)$	$\frac{1}{42}\left(7\,f^{(2,0,2)}+7\,f^{(2,2,0)}+3\,f^{(4,0,0)}\right)$	5.625
		(110)	$\frac{9\,x_1\,x_2}{8}$	$\frac{1}{30}\left(5\,f^{(1,1,2)}+3\,f^{(1,3,0)}+3\,f^{(3,1,0)}\right)$	1.125
	6	(200)	$-\frac{15}{128}\left(315\,x_1^4+90\,x_2^2\,x_1^2+90\,x_3^2\,x_1^2-354\,x_1^2-30\,x_2^2-30\,x_3^2+55\right)$	$\frac{1}{27\,720}\left(-99\,f^{(2,0,4)}-770\,f^{(2,2,2)}-99\,f^{(2,4,0)}-330\,f^{(4,0,2)}-330\,f^{(4,2,0)}-35\,f^{(6,0,0)}\right)$	82.9688
		(110)	$-\frac{9}{32}\,x_1\,x_2\left(35\,x_1^2+35\,x_2^2+15\,x_3^2-51\right)$	$\frac{1}{12\,600}\left(-45\,f^{(1,1,4)}-210\,f^{(1,3,2)}-25\,f^{(1,5,0)}-210\,f^{(3,1,2)}-126\,f^{(3,3,0)}-25\,f^{(5,1,0)}\right)$	14.3438
3	5	(300)	$-\frac{105}{16}\,x_1\left(5\,x_1^2-3\right)$	$\frac{1}{18}\left(3\,f^{(3,0,2)}+3\,f^{(3,2,0)}+f^{(5,0,0)}\right)$	196.875
		(210)	$-\frac{45}{16}\left(3\,x_1^2-1\right)x_2$	$\frac{1}{210}\left(35\,f^{(2,1,2)}+21\,f^{(2,3,0)}+15\,f^{(4,1,0)}\right)$	16.875
		(111)	$-\frac{27}{8}\,x_1\,x_2\,x_3$	$\frac{1}{10}\left(f^{(1,1,3)}+f^{(1,3,1)}+f^{(3,1,1)}\right)$	3.375
	7	(300)	$\frac{105}{128}\,x_1\left(693\,x_1^4+150\,x_2^2\,x_1^2+150\,x_3^2\,x_1^2-910\,x_1^2-90\,x_2^2-90\,x_3^2+249\right)$	$\frac{1}{1\,081\,080}\left(-3861\,f^{(3,0,4)}-30\,030\,f^{(3,2,2)}-3861\,f^{(3,4,0)}-10\,010\,f^{(5,0,2)}-10\,010\,f^{(5,2,0)}-945\,f^{(7,0,0)}\right)$	4478.91
		(210)	$\frac{45}{128}\,x_2\left(315\,x_1^4+210\,x_2^2\,x_1^2+90\,x_3^2\,x_1^2-450\,x_1^2-70\,x_2^2-30\,x_3^2+87\right)$	$\frac{1}{27\,720}\left(-99\,f^{(2,1,4)}-462\,f^{(2,3,2)}-55\,f^{(2,5,0)}-330\,f^{(4,1,2)}-198\,f^{(4,3,0)}-35\,f^{(6,1,0)}\right)$	316.406
		(111)	$\frac{27}{32}\,x_1\,x_2\,x_3\left(35\,x_1^2+35\,x_2^2+35\,x_3^2-67\right)$	$\frac{1}{12\,600}\left(-25\,f^{(1,1,5)}-126\,f^{(1,3,3)}-25\,f^{(1,5,1)}-126\,f^{(3,1,3)}-126\,f^{(3,3,1)}-25\,f^{(5,1,1)}\right)$	56.5313
4	6	(400)	$\frac{945}{64}\left(35\,x_1^4-30\,x_1^2+3\right)$	$\frac{1}{66}\left(11\,f^{(4,0,2)}+11\,f^{(4,2,0)}+3\,f^{(6,0,0)}\right)$	12403.1
		(310)	$\frac{315}{16}\,x_1\left(5\,x_1^2-3\right)x_2$	$\frac{1}{90}\left(15\,f^{(3,1,2)}+9\,f^{(3,3,0)}+5\,f^{(5,1,0)}\right)$	590.625
		(220)	$\frac{225}{32}\left(3\,x_1^2-1\right)\left(3\,x_2^2-1\right)$	$\frac{1}{42}\left(7\,f^{(2,2,2)}+3\,f^{(2,4,0)}+3\,f^{(4,2,0)}\right)$	253.125
		(211)	$\frac{135}{16}\left(3\,x_1^2-1\right)x_2\,x_3$	$\frac{1}{70}\left(7\,f^{(2,1,3)}+7\,f^{(2,3,1)}+5\,f^{(4,1,1)}\right)$	50.625
	8	(400)	$-\frac{945}{256}\left(3003\,x_1^6+525\,x_2^2\,x_1^4+525\,x_3^2\,x_1^4-4585\,x_1^4-450\,x_2^2\,x_1^2-450\,x_3^2\,x_1^2+1785\,x_1^2+45\,x_2^2+45\,x_3^2-107\right)$	$\frac{1}{360\,360}\left(-1287\,f^{(4,0,4)}-10\,010\,f^{(4,2,2)}-1287\,f^{(4,4,0)}-2730\,f^{(6,0,2)}-2730\,f^{(6,2,0)}-231\,f^{(8,0,0)}\right)$	406202.
		(310)	$-\frac{315}{128}\,x_1\,x_2\left(693\,x_1^4+350\,x_2^2\,x_1^2+150\,x_3^2\,x_1^2-1070\,x_1^2-210\,x_2^2-90\,x_3^2+345\right)$	$\frac{1}{1\,081\,080}\left(-3861\,f^{(3,1,4)}-18\,018\,f^{(3,3,2)}-2145\,f^{(3,5,0)}-10\,010\,f^{(5,1,2)}-6006\,f^{(5,3,0)}-945\,f^{(7,1,0)}\right)$	15799.2
		(220)	$-\frac{675}{256}\left(315\,x_2^2\,x_1^4-105\,x_1^4+315\,x_2^4\,x_1^2-594\,x_2^2\,x_1^2+90\,x_2^2\,x_3^2\,x_1^2-30\,x_3^2\,x_1^2+135\,x_1^2-105\,x_2^4+135\,x_2^2-30\,x_2^2\,x_3^2+10\,x_3^2-24\right)$	$\frac{1}{194\,040}\left(-693\,f^{(2,2,4)}-2310\,f^{(2,4,2)}-245\,f^{(2,6,0)}-2310\,f^{(4,2,2)}-990\,f^{(4,4,0)}-245\,f^{(6,2,0)}\right)$	6264.84
		(211)	$-\frac{945}{128}\,x_2\,x_3\left(45\,x_1^4+30\,x_2^2\,x_1^2+30\,x_3^2\,x_1^2-78\,x_1^2-10\,x_2^2-10\,x_3^2+17\right)$	$\frac{1}{138\,600}\left(-275\,f^{(2,1,5)}-1386\,f^{(2,3,3)}-275\,f^{(2,5,1)}-990\,f^{(4,1,3)}-990\,f^{(4,3,1)}-175\,f^{(6,1,1)}\right)$	1151.72

D.2 Some Optimal Kernels of Type I

Table 11: Optimal Kernels of Type I in $d = 1$ with support on $[-1, 1]$ according to Eqs. (154) and (155).

$\nu=\lvert\boldsymbol{\alpha}\rvert$	k	$\boldsymbol{\alpha}$	$\mathrm{K}_{\boldsymbol{\alpha}}(\boldsymbol{x})$	$\mathrm{AB}\left(\widehat{\partial^{\boldsymbol{\alpha}} \mathrm{f}}\right)$	$\mathrm{R}(\mathrm{K}_{\boldsymbol{\alpha}})$	$\mathrm{k}!\lvert\mu^{\mathrm{K}_{\boldsymbol{\alpha}}}\rvert$	$\mathrm{T}(\mathrm{K}_{\boldsymbol{\alpha}})\,(\mathrm{k}!)^{\frac{2\,\mathrm{d}+4\,\nu}{\mathrm{d}+2\,\mathrm{k}}}$
0	2	(0)	$\frac{1}{2}$	$\frac{f^{\prime\prime}}{6}$	0.5	0.333333	0.370107
	4	(0)	$-\frac{3}{8}\left(5\,x_1^2-3\right)$	$-\frac{f^{(4)}}{280}$	1.125	0.0857143	0.643235
1	3	(1)	$-\frac{3\,x_1}{2}$	$\frac{f^{(3)}}{10}$	1.5	0.6	0.813706
	5	(1)	$\frac{15}{8}\,x_1\left(7\,x_1^2-5\right)$	$-\frac{f^{(5)}}{504}$	9.375	0.238095	2.32774
2	4	(2)	$\frac{15}{4}\left(3\,x_1^2-1\right)$	$\frac{f^{(4)}}{14}$	22.5	1.71429	7.2621
	6	(2)	$-\frac{105}{32}\left(45\,x_1^4-42\,x_1^2+5\right)$	$-\frac{f^{(6)}}{792}$	275.625	0.909091	29.5049
3	5	(3)	$-\frac{105}{4}\,x_1\left(5\,x_1^2-3\right)$	$\frac{f^{(5)}}{18}$	787.5	6.66667	126.413
	7	(3)	$\frac{945}{32}\,x_1\left(77\,x_1^4-90\,x_1^2+21\right)$	$-\frac{f^{(7)}}{1144}$	15946.9	4.40559	695.822
4	6	(4)	$\frac{945}{16}\left(35\,x_1^4-30\,x_1^2+3\right)$	$\frac{f^{(6)}}{22}$	49612.5	32.7273	3486.3
	8	(4)	$-\frac{10\,395}{64}\left(273\,x_1^6-385\,x_1^4+135\,x_1^2-7\right)$	$-\frac{f^{(8)}}{1560}$	1500778.	25.8462	25233.6

Table 12: Optimal Kernels of Type I in $d = 2$ with support on $[-1, 1]^2$ according to Eqs. (154) and (155).

$\nu=\lvert\boldsymbol{\alpha}\rvert$	k	$\boldsymbol{\alpha}$	$\mathrm{K}_{\boldsymbol{\alpha}}(\boldsymbol{x})$	$\mathrm{AB}\left(\widehat{\partial^{\boldsymbol{\alpha}} \mathrm{f}}\right)$	$\mathrm{R}(\mathrm{K}_{\boldsymbol{\alpha}})$	$\mathrm{k}!\lvert\mu^{\mathrm{K}_{\boldsymbol{\alpha}}}\rvert$	$\mathrm{T}(\mathrm{K}_{\boldsymbol{\alpha}})\,(\mathrm{k}!)^{\frac{2\,\mathrm{d}+4\,\nu}{\mathrm{d}+2\,\mathrm{k}}}$
0	2	(0 0)	$\frac{1}{4}$	$\frac{1}{6}\left(f^{(0,2)}+f^{(2,0)}\right)$	0.25	0.333333	0.190786
	4	(0 0)	$\frac{9}{64}\left(5\,x_1^2-3\right)\left(5\,x_2^2-3\right)$	$\frac{1}{280}\left(-f^{(0,4)}-f^{(4,0)}\right)$	1.26563	0.0857143	0.451924
1	3	(1 0)	$-\frac{15}{304}\,x_1\left(7\,x_1^2-21\,x_2^2+18\right)$	$\frac{2}{19}\left(f^{(1,2)}+f^{(3,0)}\right)$	0.888158	0.631579	0.595213
	5	(1 0)	$\frac{3\,x_1\left(693\,x_1^4-17\,850\,x_2^2\,x_1^2+9940\,x_2^4-3465\,x_2^2+15\,720\,x_2^2-7782\right)}{2176}$	$-\frac{29\left(f^{(1,4)}+f^{(5,0)}\right)}{14\,280}$	10.7289	0.243697	1.89788
2	4	(2 0)	$\frac{21}{160}\left(15\,x_1^4-90\,x_2^2\,x_1^2+60\,x_1^2+30\,x_2^2-23\right)$	$\frac{11}{150}\left(f^{(2,2)}+f^{(4,0)}\right)$	15.75	1.76	5.93646
		(1 1)	$\frac{9\,x_1\,x_2}{4}$	$\frac{1}{10}\left(f^{(1,3)}+f^{(3,1)}\right)$	2.25	2.4	3.95482
	6	(2 0)	$-\frac{1}{67\,328}5\left(48\,048\,x_1^6-3\,728\,025\,x_2^2\,x_1^4+2\,171\,295\,x_1^4-720\,720\,x_2^4\,x_1^2+4\,097\,250\,x_2^2\,x_1^2-2\,127\,630\,x_1^2+240\,240\,x_2^4-620\,145\,x_2^2+268\,087\right)$	$-\frac{505\left(f^{(2,4)}+f^{(6,0)}\right)}{397\,656}$	316.01	0.914358	24.8354
		(1 1)	$\frac{225}{64}\,x_1\left(7\,x_1^2-5\right)x_2\left(7\,x_2^2-5\right)$	$\frac{1}{504}\left(-f^{(1,5)}-f^{(5,1)}\right)$	87.8906	1.42857	17.5226
3	5	(3 0)	$-\frac{315\,x_1\left(1155\,x_1^4-11\,550\,x_2^2\,x_1^2+7220\,x_2^4+6930\,x_2^2-4827\right)}{22\,336}$	$\frac{59\left(f^{(3,2)}+f^{(5,0)}\right)}{1047}$	610.933	6.76218	108.504
		(2 1)	$\frac{315}{832}\,x_2\left(-45\,x_1^4+90\,x_2^2\,x_1^2-60\,x_1^2-30\,x_2^2+29\right)$	$\frac{1}{13}\left(f^{(2,3)}+f^{(4,1)}\right)$	45.4327	9.23077	69.0939
	7	(3 0)	$\frac{1}{7\,856\,896}2835\,x_1\left(57\,915\,x_1^6-11\,816\,035\,x_2^2\,x_1^4+6\,996\,066\,x_1^4-2\,027\,025\,x_2^4\,x_1^2+15\,548\,400\,x_2^2\,x_1^2-8\,417\,790\,x_1^2+1\,216\,215\,x_2^4-4\,265\,025\,x_2^2+2\,033\,055\right)$	$-\frac{215\left(f^{(3,4)}+f^{(7,0)}\right)}{245\,528}$	18224.3	4.41335	595.791
		(2 1)	$-\frac{1}{20\,224}225\,x_2\left(-9009\,x_1^6+174\,195\,x_2^2\,x_1^4-112\,140\,x_1^4+27\,027\,x_2^4\,x_1^2-192\,612\,x_2^2\,x_1^2+118\,470\,x_1^2-9009\,x_2^4+29\,365\,x_2^2-15\,775\right)$	$-\frac{53\left(f^{(2,5)}+f^{(6,1)}\right)}{39\,816}$	2636.05	6.70886	344.45
4	6	(4 0)	$\frac{1}{8416}1155\left(1365\,x_1^6-20\,475\,x_2^2\,x_1^4+12\,495\,x_1^4+17\,550\,x_2^2\,x_1^2-11\,685\,x_1^2-1755\,x_2^2+1201\right)$	$\frac{217\left(f^{(4,2)}+f^{(6,0)}\right)}{4734}$	41155.1	33.0038	3074.57
		(3 1)	$\frac{675}{736}\,x_1\,x_2\left(231\,x_1^4-770\,x_2^2\,x_1^2+420\,x_1^2+462\,x_2^2-351\right)$	$\frac{19}{322}\left(f^{(3,3)}+f^{(5,1)}\right)$	2311.14	42.4845	1937.03
		(2 2)	$\frac{225}{16}\left(3\,x_1^2-1\right)\left(3\,x_2^2-1\right)$	$\frac{1}{14}\left(f^{(2,4)}+f^{(4,2)}\right)$	506.25	51.4286	1649.12
	8	(4 0)	$-\frac{1}{29\,668\,864}3465\left(1\,531\,530\,x_1^8-711\,878\,895\,x_2^2\,x_1^6+424\,268\,481\,x_1^6-107\,207\,100\,x_2^4\,x_1^4+1\,095\,823\,575\,x_2^2\,x_1^4-609\,898\,905\,x_1^4+91\,891\,800\,x_2^2\,x_1^2-430\,792\,425\,x_2^2\,x_1^2+218\,793\,375\,x_1^2-9\,189\,180\,x_2^2+26\,129\,745\,x_2^2-11\,731\,297\right)$	$-\frac{13\,381\left(f^{(4,4)}+f^{(8,0)}\right)}{20\,860\,920}$	1709509.	25.8628	21867.8
		(3 1)	$-\frac{1}{242\,432}1575\,x_1\,x_2\left(128\,700\,x_1^6-4\,593\,897\,x_2^2\,x_1^4+3\,073\,455\,x_1^4-900\,900\,x_2^4\,x_1^2+6\,370\,490\,x_2^2\,x_1^2-3\,955\,350\,x_1^2+540\,540\,x_2^4-1\,853\,481\,x_2^2+1\,013\,115\right)$	$-\frac{61\left(f^{(3,5)}+f^{(7,1)}\right)}{68\,184}$	154180.	36.0718	10863.7
		(2 2)	$\frac{11\,025\left(45\,x_1^4-42\,x_1^2+5\right)\left(45\,x_2^4-42\,x_2^2+5\right)}{1024}$	$\frac{1}{792}\left(-f^{(2,6)}-f^{(6,2)}\right)$	75969.1	50.9091	11630.8

Table 13: Optimal Kernels of Type I in $d = 3$ with support on $[-1, 1]^3$ according to Eqs. (154) and (155).

$\nu=\lvert\alpha\rvert$	k	α	$K_\alpha(x)$	$AB\left(\widehat{\partial^\alpha f}\right)$	$R(K_\alpha)$	$k!\lvert\mu^{K_\alpha}\rvert$	$T(K_\alpha)\,(k!)^{\frac{2d+4\nu}{d+2k}}$
0	2	(000)	$\frac{1}{8}$	$\frac{1}{6}\left(f^{(0,0,2)}+f^{(0,2,0)}+f^{(2,0,0)}\right)$	0.125	0.333333	0.118847
	4	(000)	$\frac{1}{128}\left(225\,x_2^2\,x_1^2+225\,x_3^2\,x_1^2-210\,x_1^2-210\,x_2^2+225\,x_2^2\,x_3^2-210\,x_3^2+151\right)$	$\frac{1}{280}\left(-f^{(0,0,4)}-f^{(0,4,0)}-f^{(4,0,0)}\right)$	1.17969	0.0857143	0.295273
1	3	(100)	$-\frac{15}{656}\,x_1\left(14\,x_1^2-21\,x_2^2-21\,x_3^2+22\right)$	$\frac{9}{82}\left(f^{(1,0,2)}+f^{(1,2,0)}+f^{(3,0,0)}\right)$	0.503049	0.658537	0.463238
	5	(100)	$\frac{1}{4480}3\,x_1\left(1386\,x_1^4-18375\,x_2^2\,x_1^2-18375\,x_3^2\,x_1^2+15610\,x_1^2-3465\,x_2^4-3465\,x_3^4+18720\,x_2^2-7875\,x_2^2\,x_3^2+18720\,x_3^2-13389\right)$	$-\frac{61\left(f^{(1,0,4)}+f^{(1,4,0)}+f^{(5,0,0)}\right)}{29400}$	8.96585	0.24898	1.32349
2	4	(200)	$\frac{35}{272}\left(15\,x_1^4-45\,x_2^2\,x_1^2-45\,x_3^2\,x_1^2+39\,x_1^2+15\,x_2^2+15\,x_3^2-16\right)$	$\frac{23}{306}\left(f^{(2,0,2)}+f^{(2,2,0)}+f^{(4,0,0)}\right)$	10.0368	1.80392	4.9013
		(110)	$\frac{9\,x_1\,x_2\left(35\,x_1^2+35\,x_2^2-210\,x_3^2+174\right)}{1168}$	$\frac{15}{146}\left(f^{(1,1,2)}+f^{(1,3,0)}+f^{(3,1,0)}\right)$	1.34075	2.46575	3.50876
	6	(200)	$-\frac{1}{810496}15\left(192192\,x_1^6-7479675\,x_2^2\,x_1^4-7479675\,x_3^2\,x_1^4+6718950\,x_1^4-1441440\,x_2^4\,x_1^2-1441440\,x_3^4\,x_1^2+8928900\,x_2^2\,x_1^2-2137050\,x_2^2\,x_3^2\,x_1^2+8928900\,x_3^2\,x_1^2-6912822\,x_1^2+480480\,x_2^4+480480\,x_3^4-1480365\,x_2^2+712350\,x_2^2\,x_3^2-1480365\,x_3^2+933028\right)$	$-\frac{1019\left(f^{(2,0,4)}+f^{(2,4,0)}+f^{(6,0,0)}\right)}{797832}$	255.874	0.919592	17.7952
		(110)	$-\frac{1}{8576}9\,x_1\,x_2\left(693\,x_1^4-82075\,x_2^2\,x_1^2-35175\,x_3^2\,x_1^2+69580\,x_1^2+693\,x_2^4-6930\,x_3^4+69580\,x_2^2-35175\,x_2^2\,x_3^2+52170\,x_3^2-57549\right)$	$-\frac{113\left(f^{(1,1,4)}+f^{(1,5,0)}+f^{(5,1,0)}\right)}{56280}$	60.3942	1.44563	12.5676
3	5	(300)	$-\frac{1}{22496}315\,x_1\left(1155\,x_1^4-5775\,x_2^2\,x_1^2-5775\,x_3^2\,x_1^2+4910\,x_1^2+3465\,x_2^2+3465\,x_3^2-3441\right)$	$\frac{241\left(f^{(3,0,2)}+f^{(3,2,0)}+f^{(5,0,0)}\right)}{4218}$	412.513	6.85633	91.7098
		(210)	$\frac{63\,x_2\left(-405\,x_1^4+600\,x_2^2\,x_1^2+630\,x_3^2\,x_1^2-540\,x_1^2-200\,x_2^2-210\,x_3^2+261\right)}{2368}$	$\frac{29}{370}\left(f^{(2,1,2)}+f^{(2,3,0)}+f^{(4,1,0)}\right)$	28.7331	9.40541	62.5874
		(111)	$-\frac{27}{8}\,x_1\,x_2\,x_3$	$\frac{1}{10}\left(f^{(1,1,3)}+f^{(1,3,1)}+f^{(3,1,1)}\right)$	3.375	12.	45.3729
	7	(300)	$\frac{1}{31445504}105\,x_1\left(6254820\,x_1^6-638429715\,x_2^2\,x_1^4-638429715\,x_3^2\,x_1^4+585763794\,x_1^4-109459350\,x_2^4\,x_1^2-109459350\,x_3^4\,x_1^2+886101600\,x_2^2\,x_1^2-138188250\,x_2^2\,x_3^2\,x_1^2+886101600\,x_3^2\,x_1^2-725994790\,x_1^2+65675610\,x_2^4+65675610\,x_3^4-258048225\,x_2^2+82912950\,x_2^2\,x_3^2-258048225\,x_3^2+182470308\right)$	$-\frac{431\left(f^{(3,0,4)}+f^{(3,4,0)}+f^{(7,0,0)}\right)}{491336}$	14545.1	4.42109	438.962
		(210)	$-\frac{1}{297472}15\,x_2\left(-1087086\,x_1^6+19216575\,x_2^2\,x_1^4+8235675\,x_3^2\,x_1^4-14988960\,x_1^4+2952180\,x_2^4\,x_1^2+1545390\,x_3^4\,x_1^2-23045820\,x_2^2\,x_1^2+5490450\,x_2^2\,x_3^2\,x_1^2-12305520\,x_3^2\,x_1^2+16812582\,x_1^2-984060\,x_2^2-515130\,x_3^4+3838625\,x_2^2-1830150\,x_2^2\,x_3^2+2454705\,x_3^2-2451104\right)$	$-\frac{1175\left(f^{(2,1,4)}+f^{(2,5,0)}+f^{(6,1,0)}\right)}{878472}$	1695.55	6.74125	249.554
		(111)	$-\frac{27}{128}\,x_1\,x_2\,x_3\left(1225\,x_2^2\,x_1^2+1225\,x_3^2\,x_1^2-1610\,x_1^2-1610\,x_2^2+1225\,x_2^2\,x_3^2-1610\,x_3^2+1591\right)$	$\frac{1}{504}\left(-f^{(1,1,5)}-f^{(1,5,1)}-f^{(5,1,1)}\right)$	335.602	10.	176.785

Table 13: Optimal Kernels of Type I in $d = 3$ with support on $[-1, 1]^3$ according to Eqs. (154) and (155) (contd.).

$\nu=\|\boldsymbol{\alpha}\|$	k	$\boldsymbol{\alpha}$	$K_\alpha(\boldsymbol{x})$	$AB\left(\widehat{\partial^\alpha f}\right)$	$R(K_\alpha)$	$k!\left\|\mu^{K_\alpha}\right\|$	$T(K_\alpha)\,(k!)^{\frac{2d+4\nu}{d+2k}}$
4	6	(400)	$\frac{1}{101\,312}3465\,(5460\,x_1^6 - 40\,950\,x_2^2\,x_1^4 - 40\,950\,x_3^2\,x_1^4 + 34\,965\,x_1^4 + 35\,100\,x_2^2\,x_1^2 + 35\,100\,x_3^2\,x_1^2 - 33\,870\,x_1^2 - 3510\,x_2^2 - 3510\,x_3^2 + 3517)$	$\frac{439\,(f^{(4,0,2)}+f^{(4,2,0)}+f^{(6,0,0)})}{9498}$	28700.3	33.2786	2638.02
		(310)	$\frac{1}{145\,216}1575\,x_1\,x_2\,(11\,781\,x_1^4 - 31\,990\,x_2^2\,x_1^2 - 21\,840\,x_3^2\,x_1^2 + 22\,460\,x_1^2 + 19\,194\,x_2^2 + 13\,104\,x_3^2 - 18\,525)$	$\frac{271\,(f^{(3,1,2)}+f^{(3,3,0)}+f^{(5,1,0)})}{4538}$	1461.6	42.9969	1736.45
		(220)	$\frac{25}{704}\,(315\,x_2^2\,x_1^4 - 105\,x_1^4 + 315\,x_2^4\,x_1^2 + 2502\,x_2^2\,x_1^2 - 3780\,x_2^2\,x_3^2\,x_1^2 + 1260\,x_3^2\,x_1^2 - 897\,x_1^2 - 105\,x_2^4 - 897\,x_2^2 + 1260\,x_2^2\,x_3^2 - 420\,x_3^2 + 320)$	$\frac{43}{594}\left(f^{(2,2,2)} + f^{(2,4,0)} + f^{(4,2,0)}\right)$	355.398	52.1212	1579.35
		(211)	$-\frac{945}{992}\,x_2\,x_3\,(-45\,x_1^4 + 45\,x_2^2\,x_1^2 + 45\,x_3^2\,x_1^2 - 42\,x_1^2 - 15\,x_2^2 - 15\,x_3^2 + 23)$	$\frac{5}{62}\left(f^{(2,1,3)} + f^{(2,3,1)} + f^{(4,1,1)}\right)$	80.0202	58.0645	1243.34
	8	(400)	$-\frac{1}{356\,076\,544}2835\,(22\,462\,440\,x_1^8 - 5\,221\,180\,965\,x_2^2\,x_1^6 - 5\,221\,180\,965\,x_3^2\,x_1^6 + 4\,831\,172\,346\,x_1^6 - 786\,185\,400\,x_2^4\,x_1^4 - 786\,185\,400\,x_3^4\,x_1^4 + 8\,341\,341\,750\,x_2^2\,x_1^4 - 912\,793\,875\,x_2^2\,x_3^2\,x_1^4 + 8\,341\,341\,750\,x_3^2\,x_1^4 - 7\,084\,329\,525\,x_1^4 + 673\,873\,200\,x_2^4\,x_1^2 + 673\,873\,200\,x_3^4\,x_1^2 - 3\,420\,306\,525\,x_2^2\,x_1^2 + 782\,394\,750\,x_2^2\,x_3^2\,x_1^2 - 3\,420\,306\,525\,x_3^2\,x_1^2 + 2\,607\,831\,460\,x_1^2 - 67\,387\,320\,x_2^4 - 67\,387\,320\,x_3^4 + 217\,716\,820\,x_2^2 - 78\,239\,475\,x_2^2\,x_3^2 + 217\,716\,820\,x_3^2 - 145\,074\,553)$	$-\frac{26\,783\,(f^{(4,0,4)}+f^{(4,4,0)}+f^{(8,0,0)})}{41\,727\,720}$	1353691.	25.8795	16510.
		(310)	$-\frac{1}{32\,018\,944}315\,x_1\,x_2\,(45\,598\,410\,x_1^6 - 1\,516\,834\,935\,x_2^2\,x_1^4 - 650\,072\,115\,x_3^2\,x_1^4 + 1\,226\,485\,260\,x_1^4 - 297\,054\,450\,x_2^4\,x_1^2 - 110\,672\,100\,x_3^4\,x_1^2 + 2\,212\,424\,200\,x_2^2\,x_1^2 - 328\,319\,250\,x_2^2\,x_3^2\,x_1^2 + 1\,051\,677\,900\,x_3^2\,x_1^2 - 1\,632\,045\,030\,x_1^2 + 178\,232\,670\,x_2^4 + 66\,403\,260\,x_3^4 - 677\,382\,405\,x_2^2 + 196\,991\,550\,x_2^2\,x_3^2 - 352\,404\,405\,x_3^2 + 438\,391\,008)$	$-\frac{1345\,(f^{(3,1,4)}+f^{(3,5,0)}+f^{(7,1,0)})}{1\,500\,888}$	96335.6	36.1322	7985.73
		(220)	$-\frac{1}{6\,453\,248}225\,(576\,576\,x_2^2\,x_1^6 - 192\,192\,x_1^6 - 312\,657\,975\,x_2^4\,x_1^4 + 320\,804\,820\,x_2^2\,x_1^4 - 89\,330\,850\,x_2^2\,x_3^2\,x_1^4 + 29\,776\,950\,x_3^2\,x_1^4 - 44\,403\,345\,x_1^4 + 576\,576\,x_2^6\,x_1^2 + 320\,804\,820\,x_2^4\,x_1^2 - 17\,297\,280\,x_2^2\,x_3^4\,x_1^2 + 5\,765\,760\,x_3^4\,x_1^2 - 326\,633\,220\,x_2^2\,x_1^2 - 89\,330\,850\,x_2^4\,x_3^2\,x_1^2 + 174\,771\,000\,x_2^2\,x_3^2\,x_1^2 - 40\,390\,830\,x_3^2\,x_1^2 + 44\,634\,408\,x_1^2 - 192\,192\,x_2^6 - 44\,403\,345\,x_2^4 + 5\,765\,760\,x_2^2\,x_3^4 - 1\,921\,920\,x_3^4 + 44\,634\,408\,x_2^2 + 29\,776\,950\,x_2^4\,x_3^2 - 40\,390\,830\,x_2^2\,x_3^2 + 7\,508\,220\,x_3^2 - 5\,970\,011)$	$-\frac{2011\,(f^{(2,2,4)}+f^{(2,6,0)}+f^{(6,2,0)})}{1\,588\,104}$	45553.8	51.0568	8694.21
		(211)	$\frac{1}{88\,576}135\,x_2\,x_3\,(-180\,180\,x_1^6 + 1\,907\,325\,x_2^2\,x_1^4 + 1\,907\,325\,x_3^2\,x_1^4 - 2\,261\,070\,x_1^4 + 270\,270\,x_2^4\,x_1^2 + 270\,270\,x_3^4\,x_1^2 - 2\,843\,400\,x_2^2\,x_1^2 + 1\,271\,550\,x_2^2\,x_3^2\,x_1^2 - 2\,843\,400\,x_3^2\,x_1^2 + 2\,844\,210\,x_1^2 - 90\,090\,x_2^4 - 90\,090\,x_3^4 + 566\,335\,x_2^2 - 423\,850\,x_2^2\,x_3^2 + 566\,335\,x_3^2 - 470\,116)$	$-\frac{121\,(f^{(2,1,5)}+f^{(2,5,1)}+f^{(6,1,1)})}{87\,192}$	8669.81	55.9538	4807.41

D.3 Some Optimal Kernels of Type II

Table 14: Optimal Kernels of Type II in $d = 1$ with support on $[-1, 1]$ according to Eqs. (154) and (156).

$\nu=\lvert\boldsymbol{\alpha}\rvert$	k	$\boldsymbol{\alpha}$	$\mathrm{K}_{\alpha}(\boldsymbol{x})$	$\mathrm{AB}\left(\widehat{\partial^{\alpha} \mathrm{f}}\right)$	$\mathrm{R}(\mathrm{K}_{\alpha})$	$\mathrm{k}!\lvert\mu^{\mathrm{K}_{\alpha}}\rvert$	$\mathrm{T}(\mathrm{K}_{\alpha})\,(\mathrm{k}!)^{\frac{2\,\mathrm{d}+4\,\nu}{\mathrm{d}+2\,\mathrm{k}}}$
0	2	(0)	$-\frac{3}{4}(x_1-1)(x_1+1)$	$\frac{f''}{10}$	0.6	0.2	0.349086
	4	(0)	$\frac{15}{32}(x_1-1)(x_1+1)\left(7\,x_1^2-3\right)$	$-\frac{f^{(4)}}{504}$	1.25	0.047619	0.619892
1	3	(1)	$\frac{15}{4}(x_1-1)\,x_1\,(x_1+1)$	$\frac{f^{(3)}}{14}$	2.14286	0.428571	0.747705
	5	(1)	$-\frac{105}{32}(x_1-1)\,x_1\,(x_1+1)\left(9\,x_1^2-5\right)$	$-\frac{f^{(5)}}{792}$	11.9318	0.151515	2.16788
2	4	(2)	$-\frac{105}{16}(x_1-1)(x_1+1)\left(5\,x_1^2-1\right)$	$\frac{f^{(4)}}{18}$	35.	1.33333	6.68457
	6	(2)	$\frac{315}{64}(x_1-1)(x_1+1)\left(77\,x_1^4-58\,x_1^2+5\right)$	$-\frac{f^{(6)}}{1144}$	381.635	0.629371	27.1656
3	5	(3)	$\frac{945}{16}(x_1-1)\,x_1\,(x_1+1)\left(7\,x_1^2-3\right)$	$\frac{f^{(5)}}{22}$	1288.64	5.45455	117.125
	7	(3)	$-\frac{10\,395}{64}(x_1-1)\,x_1\,(x_1+1)\left(39\,x_1^4-38\,x_1^2+7\right)$	$-\frac{f^{(7)}}{1560}$	23388.8	3.23077	638.983
4	6	(4)	$-\frac{10\,395}{32}(x_1-1)(x_1+1)\left(21\,x_1^4-14\,x_1^2+1\right)$	$\frac{f^{(6)}}{26}$	83959.6	27.6923	3252.45
	8	(4)	$\frac{135\,135}{512}(x_1-1)(x_1+1)\left(495\,x_1^6-597\,x_1^4+173\,x_1^2-7\right)$	$-\frac{f^{(8)}}{2040}$	2295308.	19.7647	23198.3

Table 15: Optimal Kernels of Type II in $d = 2$ with support on $[-1, 1]^2$ according to Eqs. (154) and (156).

$\nu=\lvert\boldsymbol{\alpha}\rvert$	k	$\boldsymbol{\alpha}$	$\mathrm{K}_{\boldsymbol{\alpha}}(\boldsymbol{x})$	$\mathrm{AB}\left(\widehat{\partial^{\boldsymbol{\alpha}} \mathrm{f}}\right)$	$\mathrm{R}(\mathrm{K}_{\boldsymbol{\alpha}})$	$\mathrm{k}!\lvert\mu^{\mathrm{K}_{\boldsymbol{\alpha}}}\rvert$	$\mathrm{T}(\mathrm{K}_{\boldsymbol{\alpha}})\,(\mathrm{k}!)^{\frac{2\mathrm{d}+4\nu}{\mathrm{d}+2\mathrm{k}}}$
0	2	(0 0)	$\frac{1}{48}\left(-15\,x_1^2 - 15\,x_2^2 + 22\right)$	$\frac{1}{9}\left(f^{(0,2)} + f^{(2,0)}\right)$	0.319444	0.222222	0.171444
	4	(0 0)	$\frac{9}{640}\left(105\,x_1^4 + 250\,x_2^2\,x_1^2 - 240\,x_1^2 + 105\,x_2^4 - 240\,x_2^2 + 108\right)$	$-\frac{3\left(f^{(0,4)}+f^{(4,0)}\right)}{1400}$	1.36688	0.0514286	0.3918
1	3	(1 0)	$\frac{15}{152}\,x_1\left(14\,x_1^2 + 15\,x_2^2 - 21\right)$	$\frac{3}{38}\left(f^{(1,2)} + f^{(3,0)}\right)$	1.18421	0.473684	0.51547
	5	(1 0)	$-\frac{9\,x_1\left(6237\,x_1^4+11\,900\,x_2^2\,x_1^2-14\,070\,x_1^2+3325\,x_2^4-11\,350\,x_2^2+6870\right)}{4352}$	$-\frac{29\left(f^{(1,4)}+f^{(5,0)}\right)}{21\,420}$	11.8883	0.162465	1.55091
2	4	(2 0)	$-\frac{3\left(7035\,x_1^4+7290\,x_2^2\,x_1^2-11\,460\,x_1^2-2430\,x_2^2+2413\right)}{1600}$	$\frac{22}{375}\left(f^{(2,2)} + f^{(4,0)}\right)$	21.195	1.408	5.11466
		(1 1)	$-\frac{9}{80}\,x_1\,x_2\left(35\,x_1^2 + 35\,x_2^2 - 62\right)$	$\frac{2}{25}\left(f^{(1,3)} + f^{(3,1)}\right)$	3.195	1.92	3.48137
	6	(2 0)	$\frac{1}{471\,296}5\left(16\,345\,329\,x_1^6 + 26\,096\,175\,x_2^2\,x_1^4 - 37\,946\,790\,x_1^4 + 5\,840\,415\,x_2^4\,x_1^2 - 29\,362\,500\,x_2^2\,x_1^2 + 22\,544\,160\,x_1^2 - 1\,946\,805\,x_2^4 + 4\,568\,265\,x_2^2 - 2\,260\,409\right)$	$-\frac{2525\left(f^{(2,4)}+f^{(6,0)}\right)}{2\,783\,592}$	362.391	0.653113	20.1282
		(1 1)	$\frac{45}{896}\,x_1\,x_2\left(693\,x_1^4 + 3430\,x_2^2\,x_1^2 - 3220\,x_1^2 + 693\,x_2^4 - 3220\,x_2^2 + 2080\right)$	$-\frac{5\left(f^{(1,5)}+f^{(5,1)}\right)}{3528}$	92.626	1.02041	13.5322
3	5	(3 0)	$\frac{35\,x_1\left(56\,133\,x_1^4+56\,400\,x_2^2\,x_1^2-102\,110\,x_1^2-33\,840\,x_2^2+37\,209\right)}{11\,168}$	$\frac{295\left(f^{(3,2)}+f^{(5,0)}\right)}{6282}$	829.119	5.63515	94.2062
		(2 1)	$\frac{105}{208}\,x_2\left(45\,x_1^4 + 105\,x_2^2\,x_1^2 - 135\,x_1^2 - 35\,x_2^2 + 36\right)$	$\frac{5}{78}\left(f^{(2,3)} + f^{(4,1)}\right)$	60.5769	7.69231	59.6364
	7	(3 0)	$-\frac{1}{62\,855\,168}2835\,x_1\left(65\,484\,705\,x_1^6 + 94\,528\,280\,x_2^2\,x_1^4 - 162\,499\,953\,x_1^4 + 17\,533\,075\,x_2^4\,x_1^2 - 125\,515\,950\,x_2^2\,x_1^2 + 115\,878\,570\,x_1^2 - 10\,519\,845\,x_2^4 + 34\,797\,450\,x_2^2 - 21\,712\,540\right)$	$-\frac{645\left(f^{(3,4)}+f^{(7,0)}\right)}{982\,112}$	21508.1	3.31001	485.434
		(2 1)	$-\frac{1}{161\,792}135\,x_2\left(463\,463\,x_1^6 + 2\,322\,600\,x_2^2\,x_1^4 - 2\,290\,995\,x_1^4 + 544\,005\,x_2^4\,x_1^2 - 2\,772\,210\,x_2^2\,x_1^2 + 1\,888\,590\,x_1^2 - 181\,335\,x_2^4 + 459\,550\,x_2^2 - 237\,540\right)$	$-\frac{53\left(f^{(2,5)}+f^{(6,1)}\right)}{53\,088}$	2764.96	5.03165	264.579
4	6	(4 0)	$-\frac{1}{33\,664}315\left(321\,321\,x_1^6 + 316\,575\,x_2^2\,x_1^4 - 654\,150\,x_1^4 - 271\,350\,x_2^4\,x_1^2 + 331\,185\,x_1^2 + 27\,135\,x_2^2 - 25\,468\right)$	$\frac{31}{789}\left(f^{(4,2)} + f^{(6,0)}\right)$	56262.1	28.289	2697.39
		(3 1)	$-\frac{45\,x_1\,x_2\left(69\,993\,x_1^4+194\,950\,x_2^2\,x_1^2-239\,820\,x_1^2-116\,970\,x_2^2+113\,895\right)}{10\,304}$	$\frac{57\left(f^{(3,3)}+f^{(5,1)}\right)}{1127}$	2878.71	36.4153	1654.81
		(2 2)	$-\frac{225}{896}\left(945\,x_2^2\,x_1^4 - 315\,x_1^4 + 945\,x_2^4\,x_1^2 - 2124\,x_2^2\,x_1^2 + 519\,x_1^2 - 315\,x_2^4 + 519\,x_2^2 - 110\right)$	$\frac{3}{49}\left(f^{(2,4)} + f^{(4,2)}\right)$	738.712	44.0816	1474.02
	8	(4 0)	$\frac{1}{118\,675\,456}35\left(208\,718\,986\,905\,x_1^8 + 281\,904\,042\,420\,x_2^2\,x_1^6 - 558\,751\,201\,008\,x_1^6 + 44\,519\,368\,950\,x_2^4\,x_1^4 - 435\,716\,442\,000\,x_2^2\,x_1^4 + 467\,124\,429\,240\,x_1^4 - 38\,159\,459\,100\,x_2^4\,x_1^2 + 172\,111\,205\,700\,x_2^2\,x_1^2 - 127\,780\,722\,900\,x_1^2 + 3\,815\,945\,910\,x_2^4 - 10\,499\,119\,560\,x_2^2 + 5\,799\,290\,051\right)$	$-\frac{93\,667\left(f^{(4,4)}+f^{(8,0)}\right)}{187\,748\,280}$	2064366.	20.1155	17986.2
		(3 1)	$\frac{1}{242\,432}175\,x_1\,x_2\left(10\,068\,201\,x_1^6 + 41\,345\,073\,x_2^2\,x_1^4 - 45\,796\,212\,x_1^4 + 9\,587\,655\,x_2^4\,x_1^2 - 58\,978\,360\,x_2^2\,x_1^2 + 44\,193\,660\,x_1^2 - 5\,752\,593\,x_2^4 + 17\,667\,699\,x_2^2 - 10\,245\,315\right)$	$-\frac{427\left(f^{(3,5)}+f^{(7,1)}\right)}{613\,656}$	162452.	28.0558	8409.95
		(2 2)	$\frac{1}{1024}175\left(18\,018\,x_2^2\,x_1^6 - 6006\,x_1^6 + 127\,575\,x_2^4\,x_1^4 - 143\,640\,x_2^2\,x_1^4 + 22\,365\,x_1^4 + 18\,018\,x_2^6\,x_1^2 - 143\,640\,x_2^4\,x_1^2 + 127\,512\,x_2^2\,x_1^2 - 16\,350\,x_1^2 - 6006\,x_2^6 + 22\,365\,x_2^4 - 16\,350\,x_2^2 + 1835\right)$	$-\frac{7\left(f^{(2,6)}+f^{(6,2)}\right)}{7128}$	78457.4	39.596	8924.02

Table 16: Optimal Kernels of Type II in $d = 3$ with support on $[-1, 1]^3$ according to Eqs. (154) and (156).

$\nu=\lvert\alpha\rvert$	k	α	$K_\alpha(x)$	$AB\left(\widehat{\partial^\alpha f}\right)$	$R(K_\alpha)$	$k!\lvert\mu^{K_\alpha}\rvert$	$T(K_\alpha)\,(k!)^{\frac{2d+4\nu}{d+2k}}$
0	2	(0 0 0)	$\frac{1}{112}\left(-15\,x_1^2-15\,x_2^2-15\,x_3^2+29\right)$	$\frac{5}{42}\left(f^{(0,0,2)}+f^{(0,2,0)}+f^{(2,0,0)}\right)$	0.163265	0.238095	0.103756
	4	(0 0 0)	$\frac{1}{1408}(945\,x_1^4+2475\,x_2^2\,x_1^2+2475\,x_3^2\,x_1^2-3120\,x_1^2+945\,x_2^4+$ $945\,x_3^4-3120\,x_2^2+2475\,x_2^2\,x_3^2-3120\,x_3^2+1904)$	$\frac{1}{440}\left(-f^{(0,0,4)}-f^{(0,4,0)}-f^{(4,0,0)}\right)$	1.24245	0.0545455	0.239621
1	3	(1 0 0)	$\frac{15}{656}\,x_1\left(21\,x_1^2+30\,x_2^2+30\,x_3^2-49\right)$	$\frac{7}{82}\left(f^{(1,0,2)}+f^{(1,2,0)}+f^{(3,0,0)}\right)$	0.640244	0.512195	0.390015
	5	(1 0 0)	$-\frac{1}{58\,240}3\,x_1\left(108\,801\,x_1^4+238\,875\,x_2^2\,x_1^2+238\,875\,x_3^2\,x_1^2-\right.$ $343\,840\,x_1^2+64\,260\,x_2^4+64\,260\,x_3^4-259\,830\,x_2^2+$ $\left.102\,375\,x_2^2\,x_3^2-259\,830\,x_3^2+207\,546\right)$	$-\frac{183\left(f^{(1,0,4)}+f^{(1,4,0)}+f^{(5,0,0)}\right)}{127\,400}$	9.49663	0.17237	1.03334
2	4	(2 0 0)	$-\frac{15\left(4095\,x_1^4+5310\,x_2^2\,x_1^2+5310\,x_3^2\,x_1^2-9294\,x_1^2-1770\,x_2^2-1770\,x_3^2+2279\right)}{11\,968}$	$\frac{23}{374}\left(f^{(2,0,2)}+f^{(2,2,0)}+f^{(4,0,0)}\right)$	12.4477	1.47594	4.10574
		(1 1 0)	$-\frac{9\,x_1\,x_2\left(2240\,x_1^2+2240\,x_2^2+2985\,x_3^2-5289\right)}{12\,848}$	$\frac{135\left(f^{(1,1,2)}+f^{(1,3,0)}+f^{(3,1,0)}\right)}{1606}$	1.7706	2.01743	3.00713
	6	(2 0 0)	$\frac{1}{810\,496}(64\,438\,374\,x_1^6+112\,195\,125\,x_2^2\,x_1^4+$ $112\,195\,125\,x_3^2\,x_1^4-192\,585\,960\,x_1^4+24\,831\,450\,x_2^4\,x_1^2+$ $24\,831\,450\,x_3^4\,x_1^2-136\,684\,800\,x_2^2\,x_1^2+$ $32\,055\,750\,x_2^2\,x_3^2\,x_1^2-136\,684\,800\,x_3^2\,x_1^2+134\,843\,160\,x_1^2-$ $8\,277\,150\,x_2^4-8\,277\,150\,x_3^4+23\,122\,575\,x_2^2-$ $10\,685\,250\,x_2^2\,x_3^2+23\,122\,575\,x_3^2-15\,636\,010)$	$-\frac{11\,209\left(f^{(2,0,4)}+f^{(2,4,0)}+f^{(6,0,0)}\right)}{11\,967\,480}$	276.372	0.674368	13.8814
		(1 1 0)	$\frac{1}{42\,880}9\,x_1\,x_2\left(74\,844\,x_1^4+410\,375\,x_2^2\,x_1^2+\right.$ $175\,875\,x_3^2\,x_1^2-434\,910\,x_1^2+74\,844\,x_2^4+46\,515\,x_3^4-$ $\left.434\,910\,x_2^2+175\,875\,x_2^2\,x_3^2-271\,020\,x_3^2+326\,052\right)$	$-\frac{1243\left(f^{(1,1,4)}+f^{(1,5,0)}+f^{(5,1,0)}\right)}{844\,200}$	62.5383	1.06013	9.58544
3	5	(3 0 0)	$\frac{1}{584\,896}105\,x_1\left(410\,949\,x_1^4+486\,600\,x_2^2\,x_1^2+486\,600\,x_3^2\,x_1^2-\right.$ $\left.963\,790\,x_1^2-291\,960\,x_2^2-291\,960\,x_3^2+402\,153\right)$	$\frac{2651\left(f^{(3,0,2)}+f^{(3,2,0)}+f^{(5,0,0)}\right)}{54\,834}$	508.759	5.80151	77.6218
		(2 1 0)	$\frac{1}{15\,392}27\,x_2\left(4515\,x_1^4+14\,175\,x_2^2\,x_1^2+10\,860\,x_3^2\,x_1^2-\right.$ $\left.20\,805\,x_1^2-4725\,x_2^2-3620\,x_3^2+6032\right)$	$\frac{319\left(f^{(2,1,2)}+f^{(2,3,0)}+f^{(4,1,0)}\right)}{4810}$	35.5419	7.95842	53.0212
		(1 1 1)	$\frac{27}{208}\,x_1\,x_2\,x_3\left(35\,x_1^2+35\,x_2^2+35\,x_3^2-89\right)$	$\frac{11}{130}\left(f^{(1,1,3)}+f^{(1,3,1)}+f^{(3,1,1)}\right)$	4.63314	10.1538	39.69
	7	(3 0 0)	$-\frac{1}{534\,573\,568}105\,x_1\left(7\,032\,618\,450\,x_1^6+10\,853\,305\,155\,x_2^2\,x_1^4+\right.$ $10\,853\,305\,155\,x_3^2\,x_1^4-21\,490\,135\,128\,x_1^4+$ $2\,003\,362\,200\,x_2^4\,x_1^2+2\,003\,362\,200\,x_3^4\,x_1^2-$ $15\,185\,915\,700\,x_2^2\,x_1^2+2\,349\,200\,250\,x_2^2\,x_3^2\,x_1^2-$ $15\,185\,915\,700\,x_3^2\,x_1^2+17\,608\,235\,780\,x_1^2-$ $1\,202\,017\,320\,x_2^4-1\,202\,017\,320\,x_3^4+$ $4\,460\,132\,925\,x_2^2-1\,409\,520\,150\,x_2^2\,x_3^2+$ $\left.4\,460\,132\,925\,x_3^2-3\,699\,089\,706\right)$	$-\frac{5603\left(f^{(3,0,4)}+f^{(3,4,0)}+f^{(7,0,0)}\right)}{8\,352\,712}$	16005.4	3.38083	345.638
		(2 1 0)	$-\frac{1}{5\,057\,024}15\,x_2$ $\left(59\,147\,088\,x_1^6+326\,681\,775\,x_2^2\,x_1^4+140\,006\,475\,x_3^2\,x_1^4-\right.$ $360\,668\,070\,x_1^4+74\,615\,310\,x_2^4\,x_1^2+29\,972\,880\,x_3^4\,x_1^2-$ $418\,921\,440\,x_2^2\,x_1^2+93\,337\,650\,x_2^2\,x_3^2\,x_1^2-$ $212\,366\,340\,x_3^2\,x_1^2+327\,232\,644\,x_1^2-$ $24\,871\,770\,x_2^2-9\,990\,960\,x_3^4+74\,304\,125\,x_2^2-$ $\left.31\,112\,550\,x_2^2\,x_3^2+42\,787\,485\,x_3^2-45\,393\,518\right)$	$-\frac{15\,275\left(f^{(2,1,4)}+f^{(2,5,0)}+f^{(6,1,0)}\right)}{14\,934\,024}$	1753.36	5.15507	190.835
		(1 1 1)	$-\frac{1}{2176}27\,x_1\,x_2\,x_3\left(3465\,x_1^4+20\,825\,x_2^2\,x_1^2+\right.$ $20\,825\,x_3^2\,x_1^2-31\,220\,x_1^2+3465\,x_2^4+3465\,x_3^4-$ $\left.31\,220\,x_2^2+20\,825\,x_2^2\,x_3^2-31\,220\,x_3^2+29\,522\right)$	$-\frac{13\left(f^{(1,1,5)}+f^{(1,5,1)}+f^{(5,1,1)}\right)}{8568}$	342.827	7.64706	134.413

Table 16: Optimal Kernels of Type II in $d = 3$ with support on $[-1, 1]^3$ according to Eqs. (154) and (156) (contd.).

$v=\|\alpha\|$	k	α	$K_\alpha(x)$	$AB(\widehat{\partial^\alpha f})$	$R(K_\alpha)$	$k!\|\mu^{K_\alpha}\|$	$T(K_\alpha)\,(k!)^{\frac{2d+4v}{d+2k}}$
4	6	(400)	$-\frac{1}{202\,624}$ $21\,(12\,699\,687\,x_1^6 + 14\,204\,925\,x_2^2\,x_1^4 + 14\,204\,925\,x_3^2\,x_1^4 - 31\,774\,155\,x_1^4 - 12\,175\,650\,x_2^2\,x_1^2 - 12\,175\,650\,x_3^2\,x_1^2 + 18\,163\,785\,x_1^2 + 1\,217\,565\,x_2^2 + 1\,217\,565\,x_3^2 - 1\,514\,005)$	$\frac{5707\,(f^{(4,0,2)}+f^{(4,2,0)}+f^{(6,0,0)})}{142\,470}$	35411.5	28.8414	2261.85
		(310)	$-\frac{1}{72\,608}21\,x_1\,x_2\,(403\,326\,x_1^4 + 1\,436\,750\,x_2^2\,x_1^2 + 879\,975\,x_3^2\,x_1^2 - 1\,943\,865\,x_1^2 - 862\,050\,x_2^2 - 527\,985\,x_3^2 + 993\,465)$	$\frac{3523\,(f^{(3,1,2)}+f^{(3,3,0)}+f^{(5,1,0)})}{68\,070}$	1716.49	37.264	1469.36
		(220)	$-\frac{1}{4224}$ $5\,(85\,365\,x_2^2\,x_1^4 - 28\,455\,x_1^4 + 85\,365\,x_2^4\,x_1^2 - 241\,470\,x_2^2\,x_1^2 + 125\,010\,x_2^2\,x_3^2\,x_1^2 - 41\,670\,x_2^2\,x_1^2 + 63\,417\,x_1^2 - 28\,455\,x_2^4 + 63\,417\,x_2^2 - 41\,670\,x_2^2\,x_3^2 + 13\,890\,x_3^2 - 15\,448)$	$\frac{559\,(f^{(2,2,2)}+f^{(2,4,0)}+f^{(4,2,0)})}{8910}$	460.455	45.1717	1371.89
		(211)	$-\frac{315\,x_2\,x_3\,(45\,x_1^4+420\,x_2^2\,x_1^2+420\,x_3^2\,x_1^2-702\,x_1^2-140\,x_2^2-140\,x_3^2+225)}{1984}$	$\frac{13}{186}\,(f^{(2,1,3)} + f^{(2,3,1)} + f^{(4,1,1)})$	99.0726	50.3226	1067.02
	8	(400)	$\frac{1}{13\,530\,908\,672}945\,(416\,418\,740\,595\,x_1^8 + 595\,214\,630\,010\,x_2^2\,x_1^6 + 595\,214\,630\,010\,x_3^2\,x_1^6 - 1\,332\,848\,637\,120\,x_1^6 + 93\,759\,091\,650\,x_2^4\,x_1^4 + 93\,759\,091\,650\,x_3^4\,x_1^4 - 954\,456\,350\,400\,x_2^2\,x_1^4 + 104\,058\,501\,750\,x_2^2\,x_3^2\,x_1^4 - 954\,456\,350\,400\,x_3^2\,x_1^4 + 1\,259\,530\,891\,920\,x_1^4 - 80\,364\,935\,700\,x_2^4\,x_1^2 - 80\,364\,935\,700\,x_3^4\,x_1^2 + 392\,952\,136\,050\,x_2^2\,x_1^2 - 89\,193\,001\,500\,x_2^2\,x_3^2\,x_1^2 + 392\,952\,136\,050\,x_3^2\,x_1^2 - 379\,938\,160\,860\,x_1^2 + 8\,036\,493\,570\,x_2^4 + 8\,036\,493\,570\,x_3^4 - 25\,123\,436\,700\,x_2^2 + 8\,919\,300\,150\,x_2^2\,x_3^2 - 25\,123\,436\,700\,x_3^2 + 18\,878\,074\,441)$	$-\frac{26\,783\,(f^{(4,0,4)}+f^{(4,4,0)}+f^{(8,0,0)})}{52\,855\,112}$	1513162.	20.4311	13159.5
		(310)	$\frac{1}{608\,359\,936}315\,x_1\,x_2$ $(6\,559\,684\,560\,x_1^6 + 28\,819\,863\,765\,x_2^2\,x_1^4 + 12\,351\,370\,185\,x_3^2\,x_1^4 - 35\,299\,153\,890\,x_1^4 + 6\,622\,723\,800\,x_2^4\,x_1^2 + 2\,251\,056\,150\,x_3^4\,x_1^2 - 43\,123\,492\,300\,x_2^2\,x_1^2 + 6\,238\,065\,750\,x_2^2\,x_3^2\,x_1^2 - 20\,108\,982\,600\,x_3^2\,x_1^2 + 36\,707\,284\,320\,x_1^2 - 3\,973\,634\,280\,x_2^4 - 1\,350\,633\,690\,x_3^4 + 13\,522\,725\,195\,x_2^2 - 3\,742\,839\,450\,x_2^2\,x_3^2 + 6\,771\,945\,195\,x_3^2 - 9\,082\,723\,302)$	$-\frac{6725\,(f^{(3,1,4)}+f^{(3,5,0)}+f^{(7,1,0)})}{9\,505\,624}$	100063.	28.5254	6171.4
		(220)	$\frac{1}{122\,611\,712}$ $225\,(786\,197\,412\,x_2^2\,x_1^6 - 262\,065\,804\,x_1^6 + 5\,940\,501\,525\,x_2^4\,x_1^4 - 7\,182\,317\,520\,x_2^2\,x_1^4 + 1\,697\,286\,150\,x_2^2\,x_3^2\,x_1^4 - 565\,762\,050\,x_3^2\,x_1^4 + 1\,206\,005\,535\,x_1^4 + 786\,197\,412\,x_2^6\,x_1^2 - 7\,182\,317\,520\,x_2^4\,x_1^2 + 366\,656\,220\,x_2^2\,x_3^4\,x_1^2 - 122\,218\,740\,x_3^4\,x_1^2 + 6\,933\,972\,960\,x_2^2\,x_1^2 + 1\,697\,286\,150\,x_2^4\,x_3^2\,x_1^2 - 3\,353\,227\,200\,x_2^2\,x_3^2\,x_1^2 + 778\,285\,170\,x_3^2\,x_1^2 - 987\,174\,732\,x_1^2 - 262\,065\,804\,x_2^6 + 1\,206\,005\,535\,x_2^4 - 122\,218\,740\,x_2^2\,x_3^4 + 40\,739\,580\,x_3^4 - 987\,174\,732\,x_2^2 - 565\,762\,050\,x_3^4\,x_3^2 + 778\,285\,170\,x_2^2\,x_3^2 - 146\,275\,980\,x_3^2 + 125\,295\,109)$	$-\frac{10\,055\,(f^{(2,2,4)}+f^{(2,6,0)}+f^{(6,2,0)})}{10\,057\,992}$	46678.7	40.308	6680.65
		(211)	$\frac{1}{1\,682\,944}$ $135\,x_2\,x_3\,(4\,570\,566\,x_1^6 + 36\,239\,175\,x_2^2\,x_1^4 + 36\,239\,175\,x_3^2\,x_1^4 - 53\,861\,220\,x_1^4 + 7\,650\,720\,x_2^4\,x_1^2 + 7\,650\,720\,x_3^4\,x_1^2 - 56\,819\,700\,x_2^2\,x_1^2 + 24\,159\,450\,x_2^2\,x_3^2\,x_1^2 - 56\,819\,700\,x_3^2\,x_1^2 + 58\,871\,520\,x_1^2 - 2\,550\,240\,x_2^4 - 2\,550\,240\,x_3^4 + 11\,692\,065\,x_2^2 - 8\,053\,150\,x_2^2\,x_3^2 + 11\,692\,065\,x_3^2 - 9\,504\,534)$	$-\frac{605\,(f^{(2,1,5)}+f^{(2,5,1)}+f^{(6,1,1)})}{552\,216}$	8832.99	44.174	3685.1

D.4 Results of the Monte Carlo Study

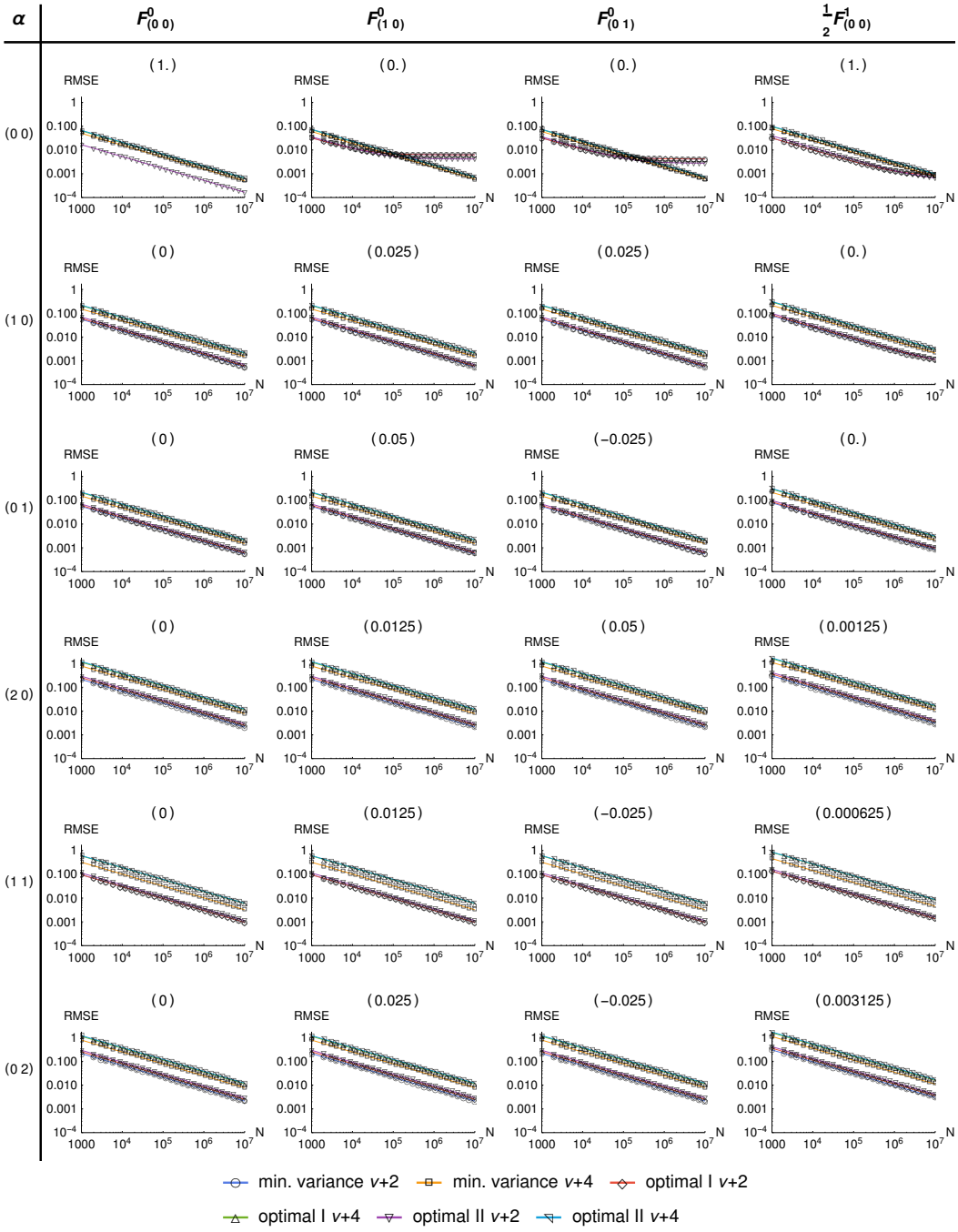


Figure 45: Empirical (symbols) and exact (lines) RMSE for the estimation of the quantities (183) with parameter set 1 using various kernels. Each plot is titled by the true value in parenthesis. The estimation of $F^0_{\mathbf{0}}$ ($\boldsymbol{\alpha} = \mathbf{0}$) using the minimum variance kernel or the optimal kernel of order 2 yields an exact result (RMSE = 0).

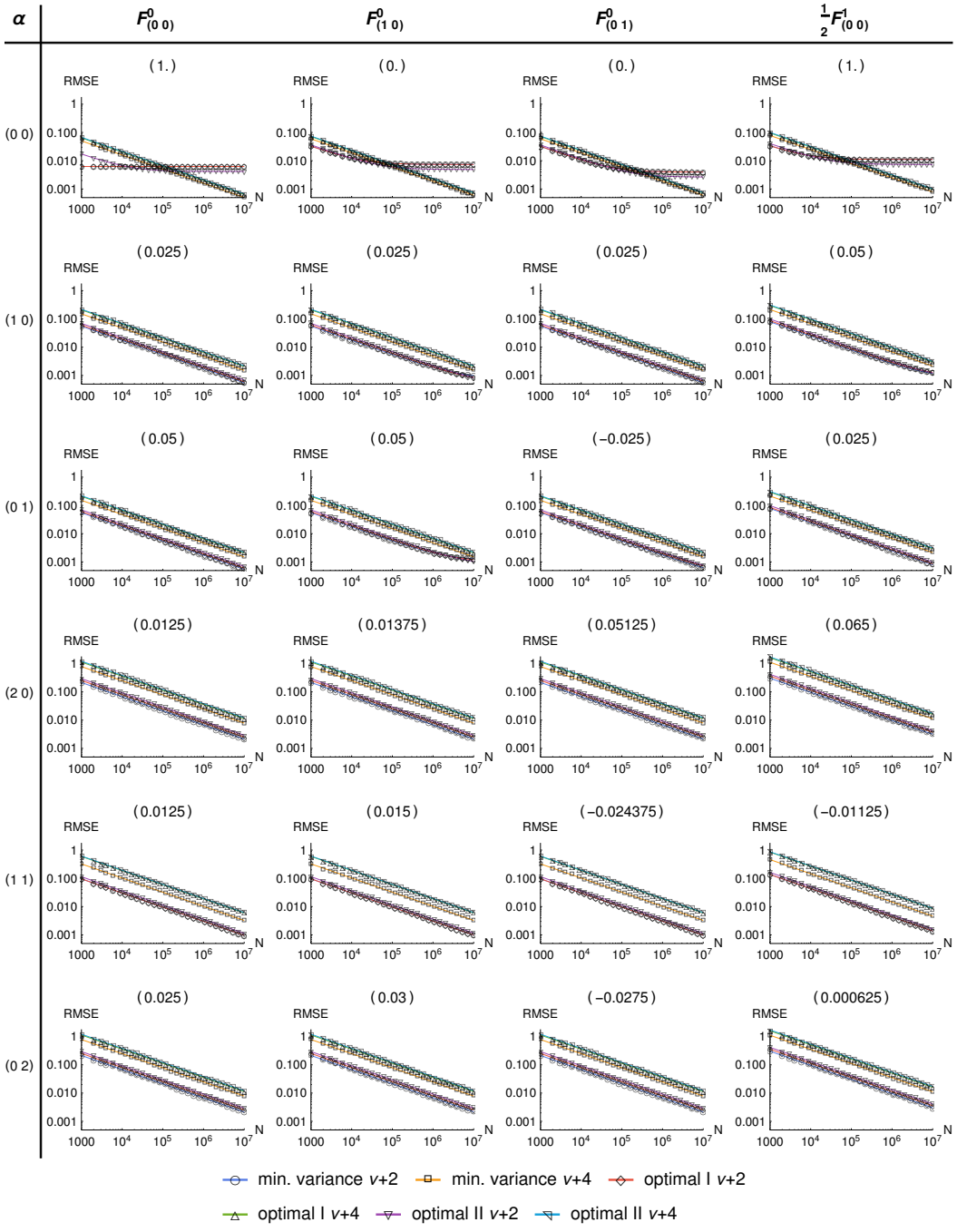


Figure 46: Empirical (symbols) and exact (lines) RMSE for the estimation of the quantities (183) with parameter set 2 using various kernels. Each plot is titled by the true value in parenthesis.

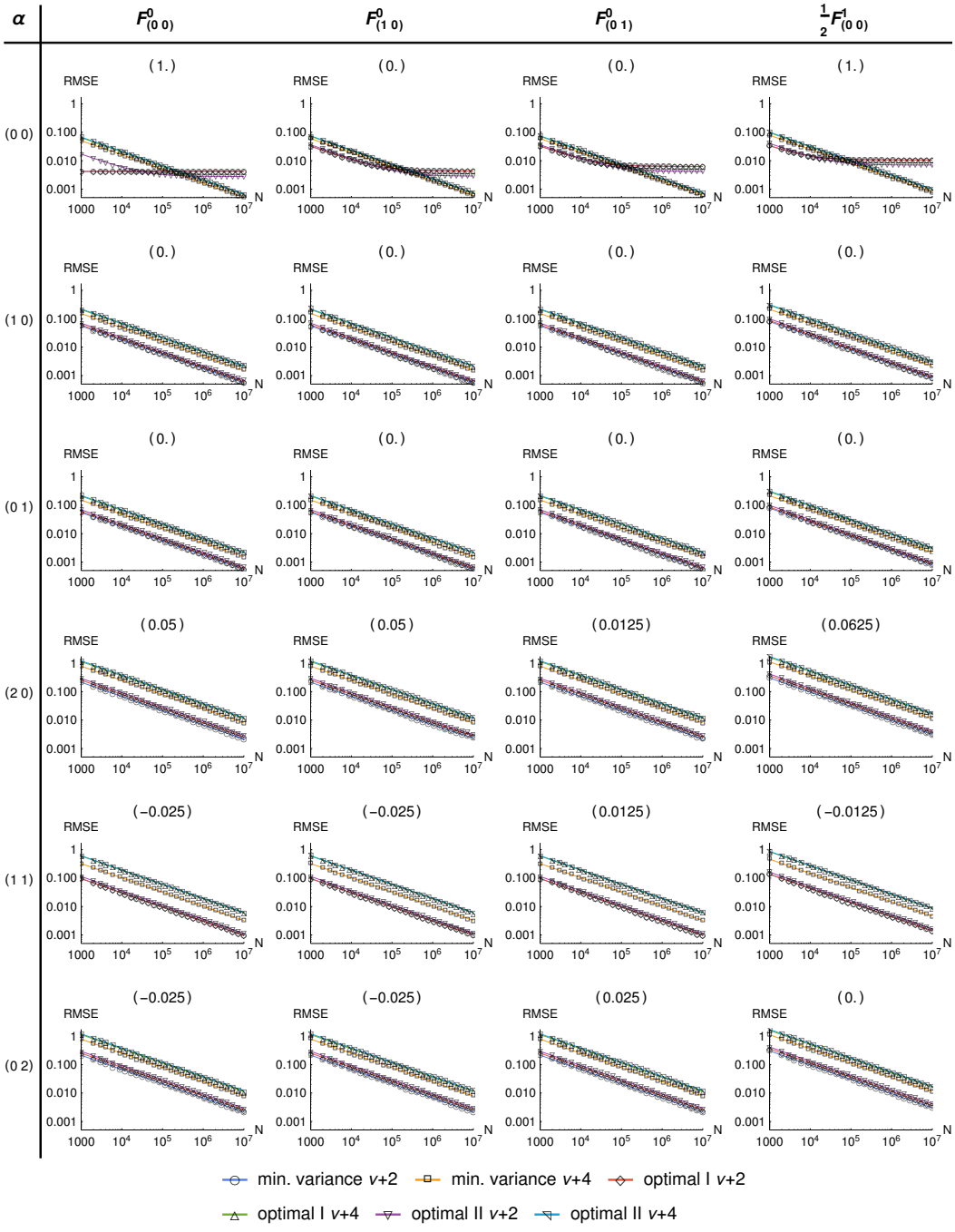


Figure 47: Empirical (symbols) and exact (lines) RMSE for the estimation of the quantities (183) with parameter set 3 using various kernels. Each plot is titled by the true value in parenthesis.

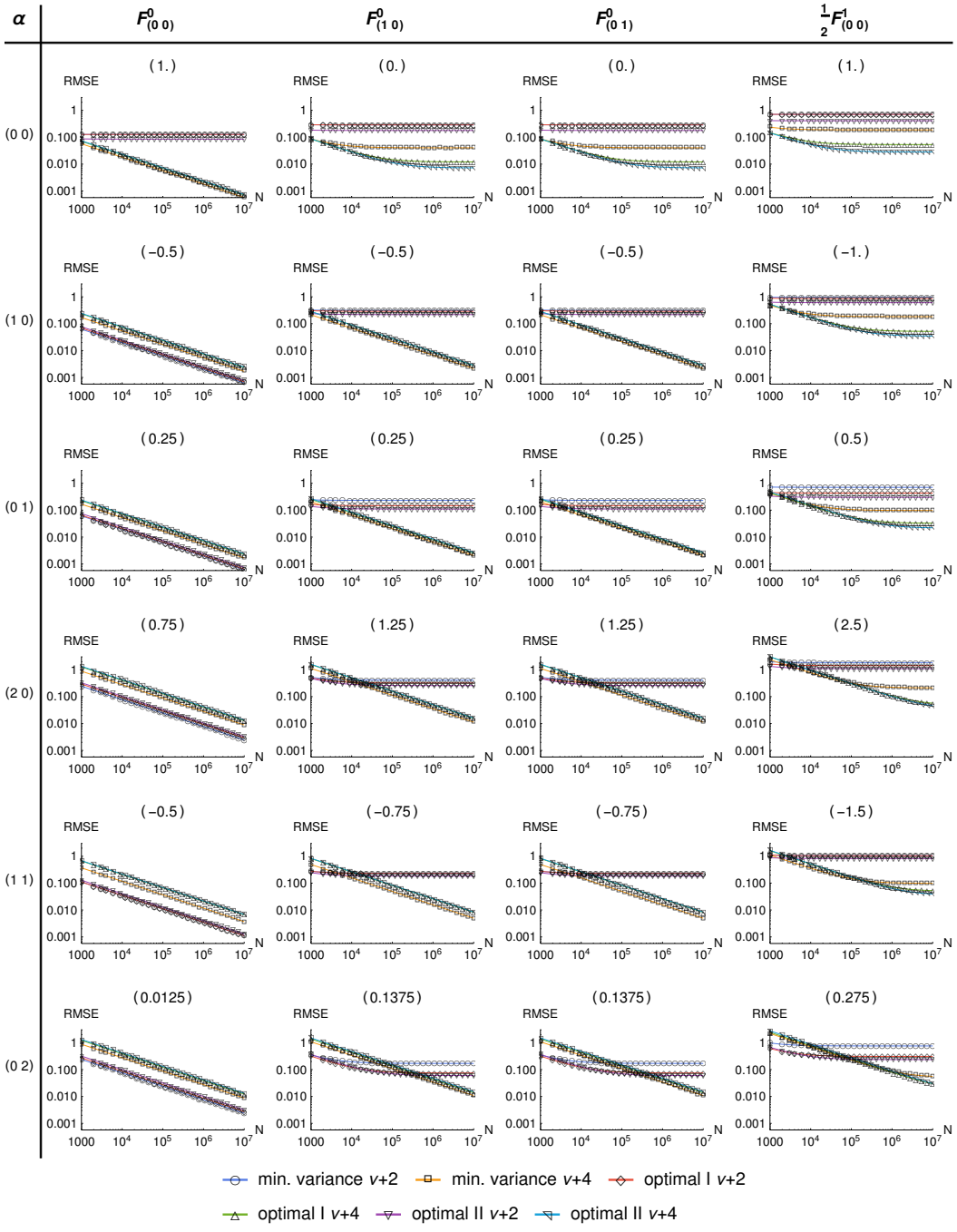


Figure 48: Empirical (symbols) and exact (lines) RMSE for the estimation of quantities (183) with parameter set 4 using various kernels. Each plot is titled by the true value in parenthesis.

Bibliography

[ABL97a] J. Allègre, D. Bisch, and J. Lengrand. "Experimental Rarefied Density Flowfields at Hypersonic Conditions over 70-Degree Blunted Cone". In: *Journal of Spacecraft and Rockets* 34.6 (Nov. 1997), pp. 714–718. DOI: 10.2514/2.3300.

[ABL97b] J. Allègre, D. Bisch, and J. Lengrand. "Experimental Rarefied Heat Transfer at Hypersonic Conditions over 70-Degree Blunted Cone". In: *Journal of Spacecraft and Rockets* 34.6 (Nov. 1997), pp. 724–728. DOI: 10.2514/2.3302.

[ABM04] M. J. Aftosmis, M. J. Berger, and S. M. Murman. "Applications of space-filling curves to Cartesian methods for CFD". In: *AIAA Paper* 1232 (2004), p. 2004. DOI: 10.2514/6.2004-1232.

[Ale+02] A. A. Alexeenko, D. A. Levin, S. F. Gimelshein, R. J. Collins, and B. D. Reed. "Numerical modeling of axisymmetric and three-dimensional flows in microelectromechanical systems nozzles". In: *AIAA journal* 40.5 (2002), pp. 897–904. DOI: 10.2514/2.1726.

[Ale03] A. A. Alexeenko. "Modeling of microscale gas flows using the direct simulation Monte Carlo method". PhD thesis. Pennsylvania State University, 2003.

[Bad13] M. Bader. *Space-Filling Curves – An Introduction with Applications in Scientific Computing*. Vol. 9. Texts in Computational Science and Engineering. Springer Berlin Heidelberg, 2013. DOI: 10.1007/978-3-642-31046-1.

[BH71] A. Bailey and J. Hiatt. *Free-flight measurements of sphere drag at subsonic, transonic, supersonic, and hypersonic speeds for continuum, transition, and near-free-molecular flow conditions*. Tech. rep. AEDC-TR-70-291. Arnold Engineering Development Center, Arnold Air Force Station, Tenn., 1971.

[Bir07] G. A. Bird. "Sophisticated DSMC". In: *Notes prepared for a short course at the DSMC07 meeting, Santa Fe, USA*. 2007.

[Bir94] G. A. Bird. *Molecular Gas Dynamics and the Direct Simulation of Gas Flows*. Oxford University Press, 1994.

[Boy03] T. B. Boykin. "Derivatives of the Dirac delta function by explicit construction of sequences". In: *American Journal of Physics* 71.5 (May 2003), pp. 462–468. DOI: 10.1119/1.1557302.

[Cam+03] P. M. Campbell, K. D. Devine, J. E. Flaherty, L. G. Gervasio, and J. D. Teresco. "Dynamic octree load balancing using space-filling curves". In: *Williams College Department of Computer Science, Technical Report CS-03* 1 (2003), p. 68.

[Cer00] C. Cercignani. *Rarefied gas dynamics: from basic concepts to actual calculations*. Vol. 21. Cambridge University Press, 2000.

[Cha43] S. Chandrasekhar. "Stochastic problems in physics and astronomy". In: *Reviews of modern physics* 15.1 (Jan. 1943), pp. 1–89.

[CM02] D. Comaniciu and P. Meer. "Mean shift: a robust approach toward feature space analysis". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24.5 (May 2002), pp. 603–619. DOI: 10.1109/34.1000236.

[DB96] S. Dietrich and I. D. Boyd. "Scalar and Parallel Optimized Implementation of the Direct Simulation Monte Carlo Method". In: *Journal of Computational Physics* 126.2 (1996), pp. 328–342. DOI: 10.1006/jcph.1996.0141.

[DeV78] C. L. DeVito. *Pure and Applied Mathematics*. Vol. 81: *Functional Analysis*. Academic Press, 1978.

[Duo+08] T. Duong, A. Cowling, I. Koch, and M. Wand. "Feature significance for multivariate kernel density estimation". In: *Computational Statistics & Data Analysis* 52.9 (2008), pp. 4225–4242. DOI: 10.1016/j.csda.2008.02.035.

[Epa69] V. A. Epanechnikov. "Non-parametric estimation of a multivariate probability density". In: *Theory of Probability & Its Applications* 14.1 (1969), pp. 153–158.

[FH75] K. Fukunaga and L. Hostetler. "The estimation of the gradient of a density function, with applications in pattern recognition". In: *IEEE Transactions on Information Theory* 21.1 (Jan. 1975), pp. 32–40. DOI: 10.1109/TIT.1975.1055330.

[Gal+14] M. A. Gallis, J. R. Torczynski, S. J. Plimpton, D. J. Rader, and T. Koehler. "Direct simulation Monte Carlo: The quest for speed". In: *Proceedings of the 29th International Symposium on Rarefied Gas Dynamics*. Vol. 1628. AIP Publishing. 2014, pp. 27–36.

[Gar09] C. W. Gardiner. *Handbook of Stochastic Methods*. Springer-Verlag GmbH, 2009.

[GJ12] M. H. Gorji and P. Jenny. "A Kinetic Model for Gas Mixtures Based on a Fokker-Planck Equation". In: *Journal of Physics: Conference Series* 362.1 (2012), p. 012042.

[GJ13] M. H. Gorji and P. Jenny. "A Fokker-Planck based kinetic model for diatomic rarefied gas flows". In: *Physics of Fluids (1994-present)* 25.6, 062002 (2013). DOI: 10.1063/1.4811399.

[GJ14] M. H. Gorji and P. Jenny. "An efficient particle Fokker-Planck algorithm for rarefied gas flows". In: *Journal of Computational Physics* 262 (2014), pp. 325–343. DOI: 10.1016/j.jcp.2013.12.046.

[GJ15] M. H. Gorji and P. Jenny. "Fokker-Planck-DSMC algorithm for simulations of rarefied gas flows". In: *Journal of Computational Physics* 287 (Apr. 2015), pp. 110–129. DOI: 10.1016/j.jcp.2015.01.041.

[GKJ13] M. H. Gorji, S. Küchlin, and P. Jenny. "A Hybrid Fokker-Planck-DSMC Solution Algorithm for the Whole Range of Knudsen Numbers". In: *Proceedings of the ASME 2013 11th International Conference on Nanochannels, Microchannels, and Minichannels*. ASME. 2013. DOI: 10.1115/ICNMM2013-73141.

[GMM85] T. Gasser, H.-G. Muller, and V. Mammitzsch. "Kernels for Nonparametric Curve Estimation". In: *Journal of the Royal Statistical Society. Series B (Methodological)* 47.2 (1985), pp. 238–252.

[Gre08] S. Green. *Cuda particles*. NVIDIA Whitepaper. 2008.

[Gro+14] W. Gropp, T. Hoefler, R. Thakur, and E. Lusk. *Using advanced MPI: Modern features of the message-passing interface*. MIT Press, 2014.

[GS10] D. Gao and T. E. Schwartzentruber. "Parallel implementation of the direct simulation Monte Carlo method for shared memory architectures". In: *Proceedings of the AIAA Aerospace Sciences Meeting, Orlando, FL*. 2010. DOI: 10.2514/6.2010-451.

[GS11] D. Gao and T. E. Schwartzentruber. "Optimizations and OpenMP implementation for the direct simulation Monte Carlo method". In: *Computers & Fluids* 42.1 (2011), pp. 73–81. DOI: 10.1016/j.compfluid.2010.11.004.

[GT12] V. K. Gupta and M. Torrilhon. "Automated Boltzmann collision integrals for moment equations". In: *AIP Conference Proceedings* 1501.1 (2012), pp. 67–74. DOI: 10.1063/1.4769474.

[GT16] M. H. Gorji and M. Torrilhon. "A Fokker-Planck model of hard sphere gases based on H-theorem". In: *30th International Symposium on Rarefied Gas Dynamics: RGD 30*. Ed. by A. Ketsdever and H. Struchtrup. Vol. 1786. AIP Conference Proceedings 1. Nov. 2016, p. 090001. DOI: 10.1063/1.4967607.

[GT17] M. H. Gorji and M. Torrilhon. "The Entropic Fokker-Planck Kinetics". Unpublished preprint. 2017.

[GTJ11] M. H. Gorji, M. Torrilhon, and P. Jenny. "Fokker-Planck model for computational studies of monatomic rarefied gas flows". In: *Journal of Fluid Mechanics* 680 (Aug. 2011), pp. 574–601. DOI: 10.1017/jfm.2011.188.

[GW06] A. L. Garcia and W. Wagner. "Generation of the Maxwellian inflow distribution". In: *Journal of Computational Physics* 217.2 (2006), pp. 693–708. DOI: 10.1016/j.jcp.2006.01.025.

[GZS10] D. Gao, C. Zhang, and T. Schwartzentruber. "A Three-Level Cartesian Geometry-Based Implementation of the DSMC Method". In: *Aerospace Sciences Meetings*. American Institute of Aeronautics and Astronautics, Jan. 2010. DOI: 10.2514/6.2010-450.

[GZS11] D. Gao, C. Zhang, and T. E. Schwartzentruber. "Particle Simulations of Planetary Probe Flows Employing Automated Mesh Refinement". In: *Journal of Spacecraft and Rockets* 48.3 (May 2011), pp. 397–405. DOI: 10.2514/1.52129.

[Har+12] D. F. Harlacher, H. Klimach, S. Roller, C. Siebert, and F. Wolf. "Dynamic Load Balancing for Unstructured Meshes on Space-Filling Curves". In: *Parallel and Distributed Processing Symposium Workshops PhD Forum (IPDPSW), 2012 IEEE 26th International*. May 2012, pp. 1661–1669. DOI: 10.1109/IPDPSW.2012.207.

[Hav17] H. J. Haverkort. "How many three-dimensional Hilbert curves are there?" In: *Journal of Computational Geometry* 8.1 (2017), pp. 206–281. DOI: 10.20382/jocg.v8i1a10.

[HSL10] T. Hoefler, C. Siebert, and A. Lumsdaine. "Scalable communication protocols for dynamic sparse data exchange". In: *ACM Sigplan Notices* 45.5 (2010), pp. 159–168.

[HT09] T. Hoefler and J. L. Traeff. "Sparse Collective Operations for MPI". In: *Proceedings of the 2009 IEEE International Symposium on Parallel&Distributed Processing*. IPDPS '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 1–8. DOI: 10.1109/IPDPS.2009.5160935.

[IMG98] M. Ivanov, G. Markelov, and S. Gimelshein. "Statistical simulation of reactive rarefied flows – Numerical approach and applications". In: *Fluid Dynamics and Co-located Conferences*. American Institute of Aeronautics and Astronautics, June 1998. DOI: 10.2514/6.1998-2669.

[IR88] M. Ivanov and S. Rogasinski. "Analysis of numerical techniques of the direct simulation Monte Carlo method in the rarefied gas dynamics". In: *Russian Journal of Numerical Analysis and Mathematical Modelling* 3 (1988), pp. 453–466. DOI: 10.1515/rnam.1988.3.6.453.

[JL15] R. Jambunathan and D. A. Levin. "A Hybrid CPU-GPU Parallel Octree Direct Simulation Monte Carlo Approach". In: *AIAA Aviation*. American Institute of Aeronautics and Astronautics, June 2015. DOI: 10.2514/6.2015-3057.

[JL16a] R. Jambunathan and D. A. Levin. "Forest of octree DSMC simulations of flow through porous media". In: *AIP Conference Proceedings* 1786.1 (2016), p. 050009. DOI: 10.1063/1.4967559.

[JL16b] R. Jambunathan and D. A. Levin. "Grid-Free Octree Approach for Modeling Heat Transfer to Complex Geometries". In: *Journal of Thermophysics and Heat Transfer* 30.2 (Jan. 2016), pp. 379–393. DOI: 10.2514/1.T4653.

[JTH10] P. Jenny, M. Torrilhon, and S. Heinz. "A solution algorithm for the fluid dynamic equations based on a stochastic model for molecular motion". In: *Journal of Computational Physics* 229.4 (2010), pp. 1077–1098. DOI: 10.1016/j.jcp.2009.10.008.

[Kan98] K. C. Kannenberg. "Computational Methods for the Direct Simulation Monte Carlo Technique with Application to Plume Impingement". PhD thesis. Cornell University, 1998.

[KB00] K. C. Kannenberg and I. D. Boyd. "Strategies for Efficient Particle Resolution in the Direct Simulation Monte Carlo Method". In: *Journal of Computational Physics* 157.2 (Jan. 2000), pp. 727–745. DOI: 10.1006/jcph.1999.6397.

[KJ17] S. Küchlin and P. Jenny. "Parallel Fokker-Planck-DSMC algorithm for rarefied gas flow simulation in complex domains at all Knudsen numbers". In: *Journal of Computational Physics* 328 (Jan. 2017), pp. 258–277. DOI: 10.1016/j.jcp.2016.10.018.

[KJ18a] S. Küchlin and P. Jenny. "Automatic mesh refinement and parallel load balancing for Fokker-Planck-DSMC algorithm". In: *Journal of Computational Physics* 363 (June 2018), pp. 140–157. DOI: 10.1016/j.jcp.2018.02.049.

[KJ18b] S. Küchlin and P. Jenny. "Kernel Estimation of Mixed Spatial Derivatives of Statistics of Scattered Data". Manuscript submitted for publication. 2018.

[Knu98] D. E. Knuth. *The Art of Computer Programming*. Vol. 3: *Sorting and Searching*. 2nd ed. Addison-Wesley, 1998.

[Kre10] G. M. Kremer. *An Introduction to the Boltzmann Equation and Transport Processes in Gases*. Springer Berlin Heidelberg, 2010. DOI: 10.1007/978-3-642-11696-4.

[LeB99] G. LeBeau. "A parallel implementation of the direct simulation Monte Carlo method". In: *Computer Methods in Applied Mechanics and Engineering* 174.3–4 (1999), pp. 319–337. DOI: 10.1016/S0045-7825(98)00302-8.

[Lin+14] A. Lintermann, S. Schlimpert, J. Grimmen, C. Günther, M. Meinke, and W. Schröder. "Massively parallel grid generation on HPC systems". In: *Computer Methods in Applied Mechanics and Engineering* 277 (2014), pp. 131–153. DOI: 10.1016/j.cma.2014.04.009.

[LW07] M. Lin and M. Weber. “Weighted ergodic theorems and strong laws of large numbers”. In: *Ergodic Theory and Dynamical Systems* 27.02 (Feb. 2007), p. 511. DOI: 10.1017/s0143385706000769.

[Ma09] T.-W. Ma. “Higher chain formula proved by combinatorics”. In: *The electronic journal of combinatorics.* 16.1 (2009).

[McC70] F. J. McCormack. “Kinetic Moment Equations for a Gas of Polyatomic Molecules with Many Internal Degrees of Freedom”. In: *Physics of Fluids* 13.6 (1970), pp. 1446–1451. DOI: 10.1063/1.1693100.

[Mon92] J. J. Monaghan. “Smoothed particle hydrodynamics”. In: *Annual review of astronomy and astrophysics* 30.1 (1992), pp. 543–574.

[MT00] G. Marsaglia and W. W. Tsang. “The Ziggurat Method for Generating Random Variables”. In: *Journal of Statistical Software* 5.8 (2000). DOI: 10.18637/jss.v005.i08.

[NT08] E. E. Nikitin and J. Troe. “70 years of Landau-Teller theory for collisional energy transfer. Semiclassical three-dimensional generalizations of the classical collinear model”. In: *Physical Chemistry Chemical Physics* 10.11 (2008), pp. 1483–1501. DOI: 10.1039/b715095d.

[Øks03] B. Øksendal. *Stochastic Differential Equations*. Springer Berlin Heidelberg, 2003. DOI: 10.1007/978-3-642-14394-6.

[PA04] A. Pinar and C. Aykanat. “Fast optimal load balancing algorithms for 1D partitioning”. In: *Journal of Parallel and Distributed Computing* 64.8 (2004), pp. 974–996. DOI: 10.1016/j.jpdc.2004.05.003.

[Par59] J. G. Parker. “Rotational and Vibrational Relaxation in Diatomic Gases”. In: *Physics of Fluids* 2.4 (1959), p. 449. DOI: 10.1063/1.1724417.

[PG16] M. Pfeiffer and M. H. Gorji. “Adaptive particle-cell algorithm for Fokker-Planck based rarefied gas flow simulations”. In: *Computer Physics Communications* (2016). DOI: 10.1016/j.cpc.2016.11.003.

[PTA08] A. Pinar, E. K. Tabak, and C. Aykanat. “One-dimensional partitioning for heterogeneous systems: Theory and practice”. In: *Journal of Parallel and Distributed Computing* 68.11 (2008), pp. 1473–1486. DOI: 10.1016/j.jpdc.2008.07.005.

[Sal+11] J. K. Salmon, M. A. Moraes, R. O. Dror, and D. E. Shaw. “Parallel Random Numbers: As Easy As 1, 2, 3”. In: *Proceedings of the 2011 International Conference for High Performance Computing, Networking, Storage and Analysis - SC ’11*. ACM Press, Nov. 2011. DOI: 10.1145/2063384.2063405.

[Sch+15] L. Schneiders, J. H. Grimmen, M. Meinke, and W. Schröder. "An efficient numerical method for fully-resolved particle simulations on high-performance computers". In: *PAMM* 15.1 (2015), pp. 495–496. DOI: 10.1002/pamm.201510238.

[Sco15] D. Scott. *Multivariate Density Estimation: Theory, Practice, and Visualization*. Wiley Series in Probability and Statistics. Wiley, 2015.

[Sei10] W. M. Seiler. *Algorithms and Computation in Mathematics*. Vol. 24: *Involution*. Springer Berlin Heidelberg, 2010. DOI: 10.1007/978-3-642-01287-7.

[SGV13] K. Stephani, D. Goldstein, and P. Varghese. "A non-equilibrium surface reservoir approach for hybrid DSMC/Navier-Stokes particle generation". In: *Journal of Computational Physics* 232.1 (2013), pp. 468–481. DOI: 10.1016/j.jcp.2012.08.017.

[Sin76] R. Singh. "Nonparametric estimation of mixed partial derivatives of a multivariate density". In: *Journal of Multivariate Analysis* 6.1 (1976), pp. 111–122. DOI: 10.1016/0047-259X(76)90023-3.

[Str05] H. Struchtrup. *Macroscopic Transport Equations for Rarefied Gas Flows*. Springer Berlin Heidelberg, 2005. DOI: 10.1007/3-540-32386-4.

[TAS03] M. Torrilhon, J. Au, and H. Struchtrup. "Explicit fluxes and productions for large systems of the moment method based on extended thermodynamics". In: *Continuum Mechanics and Thermodynamics* 15.1 (2003), pp. 97–111. DOI: 10.1007/s00161-002-0107-z.

[TG05] M. W. Tysanner and A. L. Garcia. "Non-equilibrium behaviour of equilibrium reservoirs in molecular simulations". In: *International Journal for Numerical Methods in Fluids* 48.12 (2005), pp. 1337–1349. DOI: 10.1002/fld.983.

[TS04] M. Torrilhon and H. Struchtrup. "Regularized 13-moment equations: shock structure calculations and comparison to Burnett models". In: *Journal of Fluid Mechanics* 513 (2004), pp. 171–198.

[Vas08] V. A. Vasiliev. "Non-parametric adaptive estimation of a multivariate density". In: *IFAC Proceedings Volumes* 41.2 (2008). 17th IFAC World Congress, pp. 12436–12441. DOI: 10.3182/20080706-5-KR-1001.02105.

[Wal+14] I. Wald, S. Woop, C. Benthin, G. S. Johnson, and M. Ernst. "Embree: A Kernel Framework for Efficient CPU Ray Tracing". In: *ACM Trans. Graph.* 33.4 (July 2014), 143:1–143:8. DOI: 10.1145/2601097.2601199.

[WJ94] M. P. Wand and M. C. Jones. *Kernel smoothing*. CRC Press, 1994.

[WL03] J.-S. Wu and Y.-Y. Lian. “Parallel three-dimensional direct simulation Monte Carlo method and its applications”. In: *Computers & Fluids* 32.8 (2003), pp. 1133–1160. DOI: 10.1016/S0045-7930(02)00083-X.

[WTW04] J.-S. Wu, K.-C. Tseng, and F.-Y. Wu. “Parallel three-dimensional DSMC method using mesh refinement and variable time-step scheme”. In: *Computer Physics Communications* 162.3 (Oct. 2004), pp. 166–187. DOI: 10.1016/j.cpc.2004.07.004.