EFFICIENT EULERIAN GYROKINETIC SIMULATIONS WITH BLOCK-STRUCTURED GRIDS

DENIS JAREMA

Wissenschaftliches Rechnen
Fakultät für Informatik
Technische Universität München

Vorsitzender:	Prof. Dr. Thomas Neumann
Prüfer der Dissertation:	Prof. Dr. Hans-Joachim Bungartz
	Prof. Dr. Frank Jenko

December 2016

Fakultät für Informatik der Technischen Universität München

Efficient Eulerian Gyrokinetic Simulations with Block-Structured Grids

Denis Jarema

Vollständiger Abdruck der von der promotionsführenden Einrichtung
Fakultät für Informatik
der Technischen Universität München zur Erlangung des akademischen Grades
eines Doktors der Naturwissenschaften (Dr. rer. nat.)
genehmigten Dissertation.

Vorsitzende/-r: Prof. Dr. T. Neumann

Prüfende/-r der Dissertation:

1. Prof. Dr. H.-J. Bungartz

2. Prof. Dr. F. Jenko (University of California, Los Angeles)

Die Dissertation wurde am 20.01.2017 bei der Technischen Universität München
eingereicht und durch die Fakultät für Informatik am 02.03.2017 angenommen.

Anhang I

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die bei der promotionsführenden Einrichtung
Fakultät für Informatik

der TUM zur Promotionsprüfung vorgelegte Arbeit mit dem Titel:
Efficient Eulerian Gyrokinetic Simulations with Block-Structured Grids

in Institut für Informatik, Lehrstuhl für Wissenschaftliches Rechnen
Fakultät, Institut, Lehrstuhl, Klinik, Krankenhaus, Abteilung

unter der Anleitung und Betreuung durch: Prof. Dr. H.-J. Bungartz ohne sonstige Hilfe erstellt und bei der Abfassung nur die gemäß § 6 Ab. 6 und 7 Satz 2 angebotenen Hilfsmittel benutzt habe.

☒ Ich habe keine Organisation eingeschaltet, die gegen Entgelt Betreuerinnen und Betreuer für die Anfertigung von Dissertationen sucht, oder die mir obliegenden Pflichten hinsichtlich der Prüfungsleistungen für mich ganz oder teilweise erledigt.

☒ Ich habe die Dissertation in dieser oder ähnlicher Form in keinem anderen Prüfungsverfahren als Prüfungsleistung vorgelegt.

☐ Die vollständige Dissertation wurde in _____ veröffentlicht. Die promotionsführende Einrichtung

_____

hat der Veröffentlichung zugestimmt.

☒ Ich habe den angestrebten Doktorgrad noch nicht erworben und bin nicht in einem früheren Promotionsverfahren für den angestrebten Doktorgrad endgültig gescheitert.

☐ Ich habe bereits am _____ bei der Fakultät für _____

_____

der Hochschule _____

unter Vorlage einer Dissertation mit dem Thema _____

_____

die Zulassung zur Promotion beantragt mit dem Ergebnis: _____

_____

Die öffentlich zugängliche Promotionsordnung der TUM ist mir bekannt, insbesondere habe ich die Bedeutung von § 28 (Nichtigkeit der Promotion) und § 29 (Entzug des Doktorgrades) zur Kenntnis genommen. Ich bin mir der Konsequenzen einer falschen Eidesstattlichen Erklärung bewusst.

Mit der Aufnahme meiner personenbezogenen Daten in die Alumni-Datei bei der TUM bin ich

☒ einverstanden, ☐ nicht einverstanden.

Ort, Datum, Unterschrift

ABSTRACT

Gaining a deep understanding of plasma microturbulence is of paramount importance for the development of future nuclear fusion reactors, because it causes a strong outward transport of heat and particles. Gyrokinetics has proven itself as a valid mathematical model to simulate such plasma microturbulence effects. In spite of the advantages of this model, nonlinear radially extended (or global) gyrokinetic simulations are still extremely computationally expensive, involving a very large number of computational grid points. Hence, methods that reduce the number of grid points without a significant loss of accuracy are a prerequisite to be able to run high-fidelity simulations.

At the level of the mathematical model, the gyrokinetic approach achieves a reduction from six to five coordinates in comparison to the fully kinetic models. This reduction leads to an important decrease in the total number of computational grid points. However, the velocity space mixed with the radial direction still requires a very fine resolution in grid based codes, due to the disparities in the thermal speed, which are caused by a strong temperature variation along the radial direction. An attempt to address this problem by modifying the underlying gyrokinetic set of equations leads to additional nonlinear terms, which are the most expensive parts to simulate. Furthermore, because of these modifications, well-established and computationally efficient implementations developed for the original set of equations can no longer be used.

To tackle such issues, in this thesis we introduce an alternative approach of block-structured grids. This approach reduces the number of grid points significantly, but without changing the underlying mathematical model. Furthermore, our technique is minimally invasive and allows the reuse of a large amount of already existing code using rectilinear grids, modifications being necessary only on the block boundaries. Moreover, the block-structured grid can be applied to different Eulerian gyrokinetic simulation codes, as the technique relies on a general approach. We implemented and tested our block-structured grids in GENE (http://genecode.org), a highly parallel and heavily used gyrokinetic code, for which it is crucial to keep the good parallel characteristic of the implementation and allow developers to easily port the code written for the original grid to the block-structured counterpart. The presented scenarios clearly demonstrate benefits of the block-structured grids: a high speedup, a tremendously reduced memory footprint and size of diagnostic output data, and the capability to significantly advance the frontier of feasible simulations.

ZUSAMMENFASSUNG

Die Erarbeitung eines tieferen Verständnisses der Plasma-Mikroturbulenz ist für die Entwicklung künftiger Kernfusionsreaktoren, da sie einen starken äußeren Transport von Wärme und Teilchen bewirkt, von zentraler Bedeutung. Die Gyrokinetik hat sich hierbei als ein adäquates mathematisches Modell erwiesen, um derartige Plasma-Mikroturbulenz-Effekte zu simulieren. Trotz der Vorteile dieses Modells sind nichtlineare radial ausgedehnte (oder globale) gyrokinetische Simulationen extrem rechenintensiv und beinhalten eine sehr große Anzahl von Gitterpunkten. Methoden, die deren Anzahl von Gitterpunkten ohne einen signifikanten Verlust an Genauigkeit reduzieren, sind eine Voraussetzung, um akkurate Simulationen durchführen zu können.

Auf der Ebene des mathematischen Modells erreicht die gyrokinetische Vorgehensweise eine Reduktion von sechs auf fünf Koordinaten im Vergleich zu den vollkinetischen Modellen. Dies führt zu einer deutlichen Verringerung der Gesamtzahl der Gitterpunkte. Aufgrund der zusätzlichen radialen Abhängigkeit von globalen Simulationen, erfordert der Geschwindigkeitsraum jedoch eine hohe Anzahl von Gitterpunkten. Der Grund dafür sind die Unterschiede in der thermischen Geschwindigkeit, die durch einen starken Temperaturunterschied entlang der radialen Richtung verursacht werden. Ein Versuch, diese Temperaturvariation durch Modifizierung des gyrokinetischen Gleichungssystems zu adressieren, führt zu zusätzlichen nichtlinearen Termen, deren numerische Behandlung am aufwendigsten ist. Weiterhin können infolge dieser Modifikationen gut etablierte und rechnerisch effiziente Implementierungen, die für das ursprüngliche Gleichungssystem entwickelt wurden, nicht mehr verwendet werden.

Um diese Probleme anzugehen, wird in dieser Dissertation eine alternative Vorgehensweise in Form von blockstrukturierten Gittern vorgestellt. Diese Methode reduziert die Anzahl der Gitterpunkte signifikant, jedoch ohne das zugrundeliegende mathematische Modell zu verändern. Darüber hinaus ist unsere Technik minimal invasiv und ermöglicht die Wiederverwendung einer großen Menge von existierendem Code mit geradlinigen Gittern. Modifikationen sind nur an den Blockgrenzen notwendig. Darüber hinaus kann das blockstrukturierte Gitter auf verschiedene Eulersche gyrokinetische Simulationscodes angewendet werden, da die Technik auf einem allgemeinen Ansatz beruht. Wir haben unsere blockstrukturierten Gitter in einem hochparallelen und stark genutzten gyrokinetischen Code, GENE (http://genecode.org), implementiert und getestet. Für diesen wissenschaftlichen Code ist es entscheidend, die gute Parallelisierbarkeit der Implementierung beizubehalten und es Entwicklern zu ermöglichen, den Code, der für das ursprüngliche Gitter geschrieben wurde, leicht in die blockstrukturierte Entsprechung zu portieren. Die dargestellten Szenarien zeigen deutlich die Vorteile der blockstrukturierten Gitter: ein hohes Beschleunigungspotenzial, eine enorme Reduktion des Speicherbedarfs und der Größe der Diagnosedaten sowie die Fähigkeit, die Grenze der realisierbaren Simulationen signifikant voranzutreiben.

PUBLICATIONS

Some ideas and figures have appeared previously in the following publications:

[1] D. Jarema, H. J. Bungartz, T. Görler, F. Jenko, T. Neckel, and D. Told. "Block-structured grids for Eulerian gyrokinetic simulations." In: *Computer Physics Communications* 198 (2016), pp. 105–117.

[2] D. Jarema, H. J. Bungartz, T. Görler, F. Jenko, T. Neckel, and D. Told. "Block-Structured Grids in Full Velocity Space for Eulerian Gyrokinetic Simulations." In: *Computer Physics Communications* 215 (2017), pp. 49–62.

ACKNOWLEDGMENTS

I would like to express my gratitude towards my thesis advisers Prof. Dr. Hans-Joachim Bungartz and Prof. Dr. Frank Jenko for giving me the opportunity to embark on this research and advising me throughout the entire time. I would like to thank Hans-Joachim Bungartz for his prompt and constructive feedback. Furthermore, I have enjoyed the friendly environment at his research group. I am indebted to Frank Jenko for welcoming me into his research group, which made my working with GENE considerably easier, as well as making my research stay at UCLA (University of California, Los Angeles) possible.

I would like to acknowledge my mentors Dr. Tobias Görler, Dr. Tobias Neckel, and Dr. Daniel Told. This research project would hardly have been possible without their active participation. Tobias Görler and Daniel Told were an indispensable source of knowledge for all aspects of gyrokinetic simulations and beyond. I am very grateful to them for their tremendous support in all theoretical and technical questions. Tobias Neckel was not only very supportive concerning various scientific topics, but also helped with my numerous organizational questions. Furthermore, I enjoyed a lot doing teaching under his guidance. I would like to thank all my mentors for proofreading and improving this thesis.

I am grateful to all my former and current colleagues at the chair of Scientific Computing in Computer Science at the Technische Universität München for interesting discussions, which help me evolve as a researcher and a person.

I am deeply indebted to my family for all their understanding, encouragement, and patience, which was vital for me to bring this thesis to completion.

The work presented in this thesis has been supported by HEPP (the International Helmholtz Graduate School for Plasma Physics).

CONTENTS

1 INTRODUCTION

Computational science has become indispensable to both science and engineering to help gain insight into the behavior of complex systems and support decision making. Running numerical simulations is particularly beneficial in cases when experiments alone would be prohibitively expensive and time-consuming. To this purpose, computational science builds upon both applied mathematics — for instance, to approximate systems using computational models — and computer science — for example, to develop efficient simulation algorithms.

An active area of research where running numerical simulations plays a major role is fusion plasma. Fusion is a practical and essentially inexhaustible source of energy with acceptable environmental qualities. Constructing nuclear fusion reactors requires a deep understanding of hot (fusion-relevant) plasma.

The most daring fusion research project so far is ITER[1], for which the first experiment of burning plasma is planned to take place in 2025. The costs for this experiment are, nonetheless, tremendous. Aside from the initial expenses necessary to construct the reactor, a single plasma discharge is expected to cost approximately one million U.S. dollars [1]. Furthermore, the number of the discharges is limited (about 30,000) and determines the lifetime of the ITER fusion device.

In this context, simulations are mandatory, first of all in order to estimate the physical, economical, and engineering feasibility, and to determine an optimal design. After the fusion device is operational, numerical simulations are useful to select appropriate plasma discharge scenarios and predict performance. They also serve to estimate disruptive limits due to violent instabilities in plasma, which result in a termination of the plasma discharge with large transient forces on adjacent structures. Moreover, as the experimental diagnostics can sample only a small portion of the whole relevant phase space, oftentimes synthetic diagnostics obtained by augmenting the real measurements using computer simulations are necessary to facilitate comprehension and improve existing physical models.

FUSION PLASMA SIMULATION CODES

A vast variety of physical phenomena occurs in magnetically confined plasma, which exhibits complex characteristics such as high anisotropic properties, complicated geometric details, nonlinearities, sensitivity to initial conditions, etc. Running a simulation whose model addressed all relevant phenomena would be prohibitively expensive from a computational perspective. Nevertheless, since many of these phenomena are separated by wide ranges of spatial and temporal scales, they can be modeled individually. To this purpose, several sophisticated computational models are being developed to address particular properties of fusion plasma.

1 International Thermonuclear Experimental Reactor

To enable a fully predictive model, several big scale projects have been initiated, such as FSP[2] [1, 2] in the U.S., ITM[3] [3, 4] in Europe, and BPSI[4] [5] in Japan. The long term goal of such projects is to carry out numerical simulations of a burning magnetically confined plasma over time intervals exceeding several energy-confinement times. The end result is envisioned as an integrated computational tool comprising several coupled self-consistent simulation codes of all important physical phenomena taking place in fusion plasma. Such a tool targets at understanding the underlying physical processes, designing and engineering future fusion reactors, guiding experiments, etc. Furthermore, the existing plasma models are supposed to be updated based on ongoing experimental results.

The aforementioned projects plan to couple several scientific software applications, which can be roughly categorized into two classes:

- *Systems codes* estimate the economic and engineering feasibility of a future fusion device. To make evaluations possible, such codes use simplified and often ad hoc physical models of every part of the reactor systems. These systems codes have a very broad scope, ranging from the fusion plasma-related physics, to the materials used, the actual building of fusion reactors, the costs involved, or the production of electricity. Operation of such software is based on optimizing a set of parameters leading to minimum (or maximum) so-called Figures of Merits, such as: the plasma aspect ratio, the plasma major radius, the divertor heat load, the neutron wall load, the cost of electricity or construction costs. The systems codes are not intended for scientific discovery, but rather concern the engineering and economic aspects of a plausible design of a fusion reactor. For more details on systems codes we refer to [6–9].

- *Specialized physics codes* simulate a particular physical model. In comparison with systems codes, there is a huge diversity of physics codes, which can be roughly classified into the following groups [3]:
  - *Equilibrium and linear MHD[5] stability codes* are used to determine the magnetic geometry of the confinement device. For example, the equilibrium magnetic field configuration helps choose a system of coordinates in the position space (as described in Section 3.2). High resolution and MHD equilibrium reconstruction code examples can be found in [10–13]; for MHD stability code examples, we refer to [14–16].
  - *Nonlinear MHD and disruption codes* are used for simulating sawtooth oscillations, the destabilization of resistive wall modes, neoclassical tearing modes, as well as evaluating the impact of edge localized modes. The sawtooth model is usually implemented in various transport codes, e. g., JETTO [17] and ASTRA [18]. Examples of nonlinear MHD codes are provided in [19–21].
  - *Transport and discharge evolution codes* solve the one-dimensional (i. e., radial direction or magnetic flux coordinate) transport equations for different

2 Fusion Simulation Project
3 Integrated Tokamak Modeling
4 Burning Plasma Simulation Initiative
5 Magnetohydrodynamics

plasma quantities: ion and electron temperatures, particle and impurity densities, current density, plasma momentum, etc. For code examples, see [6, 17, 18, 22, 23].

- *Transport processes and microstability codes* simulate plasma turbulence, based either on the five-dimensional (three coordinates in the position space and two in the velocity space) gyrokinetic model or on the three-dimensional (three coordinates in the position space) fluid model. Such codes are usually further classified into three main types: microstability, microturbulence, and neoclassical transport. Most of these codes consider either the core or the edge plasma. For more details on the core plasma turbulence codes based on a gyrokinetic model, we refer to extensive reviews [24, 25] and also to Section 2.2.
- *Heating, current drive, and fast particles codes* are dedicated, as their name suggests, to investigations and predictions of heating, current drive, and fast particles instabilities. Corresponding code examples are [26–28].

The specialized plasma physics codes are often distinguished not only by what kind of physics they address, but also by the part of magnetically confined plasma that they are dedicated to. The major regions distinguished in a fusion device are the core and edge plasma. The properties of these types of plasma are very different, requiring, thus, different mathematical models. Coupling the core and edge plasma simulations is, however, still an open field of research.

A typical Tokamak[6] cross-section with a schematic distinction of different geometric regions (including the edge and core of the magnetically confined plasma) is illustrated in Figure 1. From this figure, we observe that the topology of the magnetic field lines is different for the core and edge plasma regions. The core region comprises only closed magnetic field lines, whereas, in the edge regions, the topology changes from closed magnetic field lines to magnetic field lines intersecting the material walls separated by the magnetic separatrix (in diverted discharges).

For core plasma simulations, the typical time scales of different physical processes are well-separated, as demonstrated in Figure 2. In accordance with this diagram, four types of specialized plasma core codes exist, each yielding one simulation for the physical models with specific time scales. The situation is, however, different for edge plasma. In many scenarios of interest, the major characteristic time scales of different phenomena overlap, making the physical model extremely complex. For an overview of computational edge plasma models see [29].

THESIS APPROACHES

In this thesis, we address the computational aspects of the gradient driven turbulence in core plasma. Understanding and reliably predicting the plasma microturbulence is of utmost importance for the development of future nuclear fusion reactors, because the microturbulence is the main reason for the strong outward transport of heat and particles in fusion experiments. Multiple spatial and temporal scales are involved in corresponding simulations based on a gyrokinetic model (see [24, 30]). As a result, high

6 Transliteration of a corresponding Russian acronym, which stands for toroidal chamber with magnetic coils.

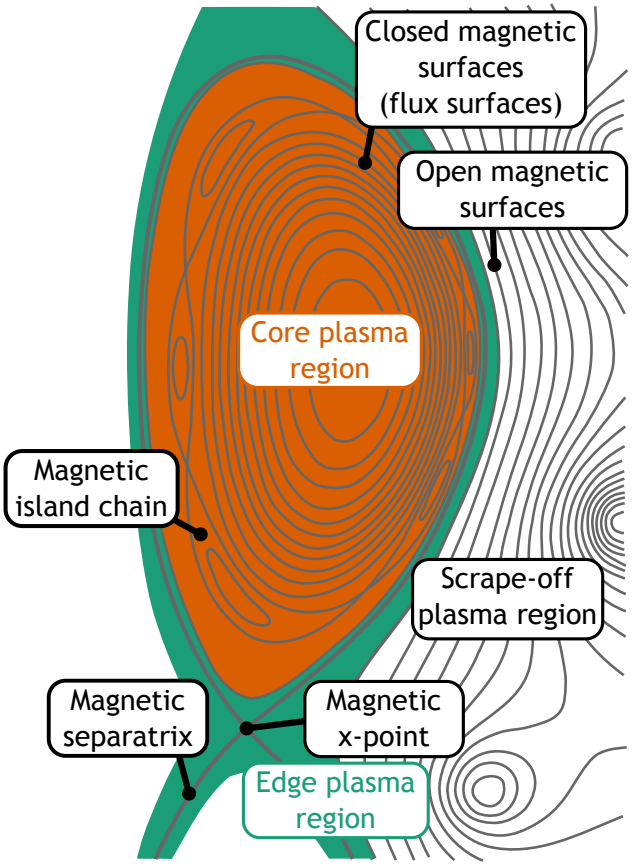


Figure 1: Schematic representation of a Tokamak cross-section with major plasma and magnetic regions, based on [1].

physical-fidelity simulations are not achievable without high performance computing. Furthermore, optimization and UQ[7] applications demand even more computational load. One of the main reasons is that appropriate gyrokinetic simulations require an enormous amount of degrees of freedom for a lot of physical scenarios. For example, the core plasma turbulence simulations in the future large-scale fusion experiment ITER necessitate a computational grid with about ten billion grid points[8] for grid-based (Eulerian) codes. Moreover, according to recent studies, this kind of simulations might require even up to a trillion computational grid points, depending on the radial profiles of temperature and density. In contrast to many other scientific applications, this number of grid points leads to extremely computationally expensive simulations already for the fully parallelized gyrokinetic implementation.

In the current work, we consider a problem of multiple scale structures of distribution functions in the velocity space for radially extended gyrokinetic simulations. More specifically, we consider the temperature spatial variation $T(x)$, which is one of the main causes of the multiple scales along the radial coordinate x. Another reason is the disparity in the thermal speed $v_T = \sqrt{2T/m}$, due to the huge differences in temper-

7 Uncertainty Quantification

8 The number of computational grid points for each coordinate are `n_spec` = 2, `nx0` = 2048, `nky0` = 32, `nz0` = 24, `nv0` = 96, and `nw0` = 32. This mesh does not resolve electron spatial scales. The phase space coordinates are described in Chapter 3.

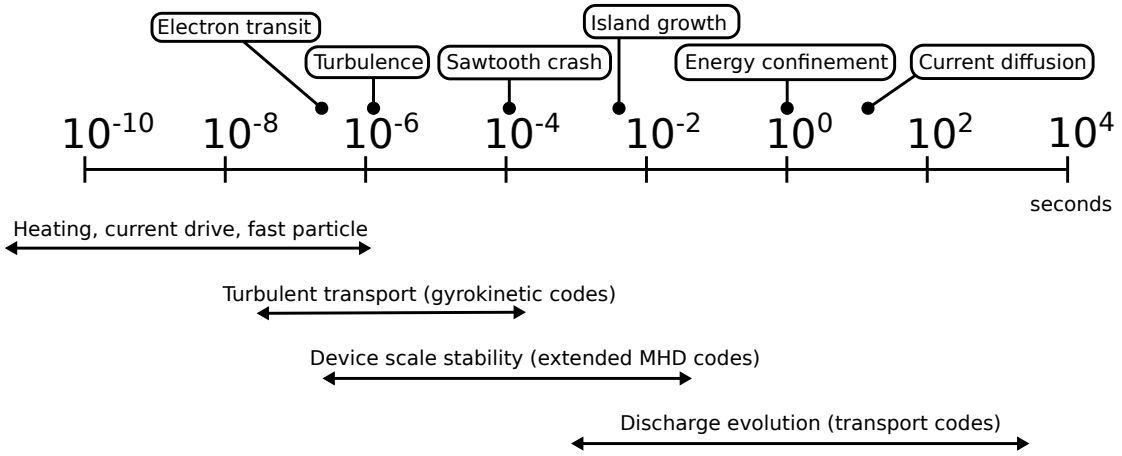


Figure 2: Typical time scales in the core plasma simulations with $B = 10\,T$, $R = 2\,m$, $n_e = 1 \times 10^{14}\,cm^{-1}$, $T = 10\,keV$; based on [1].

ature T and mass m of the species involved in simulations. This disparity necessitates different discretization grids for each species and results in complicated numerical treatments of collisions, see [31–34]. Fortunately, these two problems are decoupled and can be treated separately.

The straightforward way of resolving the multiple scales of the distribution function in the velocity space is setting a computational grid with a very high resolution and a wide range for the velocity coordinates. In the following, we refer to this traditional method as the regular grid approach. Naturally, it leads to very computationally expensive simulations.

To considerably reduce the number of computational nodes in the velocity space grids without a loss of accuracy, we introduce a general concept of adjusting these grids to the background distribution function. The latter depends on the temperature radial variation $T(x)$. We detail two types of computational grids based on this adjustment: grids with transformed velocity coordinates and grids consisting of multiple blocks of rectangular regular grids (block-structured grids). The former approach leads to non-trivial modifications of the underlying mathematical model, which is estimated to cause additional computational overheads (new nonlinear terms, derivatives in the magnetic moment direction, etc.). The latter method of block-structured grids does not require modifications of the governing equations. The changes are technical and mostly related to the geometry of the block-structured grids. For example, the boundaries of grid-blocks require a special treatment.

Based on the comparison of the aforementioned two methods, we have chosen the block-structured grids for the already existing and comprehensive Eulerian gyrokinetic code GENE[9] [35–37]. According to numerical tests, the implementation of these grids in GENE was shown to be accurate with considerably less computational nodes than the original regular grids. Moreover, due to the relative non-invasiveness of our approach, we reuse a large amount of original regular-grid-specific code, which allows elaborating grids and evolving the physical models by different developers simultaneously.

9 Gyrokinetic Electromagnetic Numerical Experiment

Several of the ideas and results presented in this thesis have been published in peer-reviewed articles [38, 39]. The block-structured grids for the gyrokinetic grid-based simulations have been introduced for the first time in [38], which describes the theoretical background for the grids for the whole velocity space (parallel velocity and magnetic moment). The main results of the global nonlinear gyrokinetic simulations have, nevertheless, been shown for the block-structured grids considering the parallel velocity coordinate only. The main numerical operation requiring modification concerned computing the radial derivatives, which appear in many terms of the governing set of equations. This approach has been extended to the whole velocity space in a follow-up study [39]. The construction of the extended block-structured grids differs significantly from the first version (the parallel velocity only), because including the magnetic moment direction introduces different types of numerical operations, such as gyro-averaging and quadrature, which have to be adjusted correspondingly.

STRUCTURE OF THE THESIS

The rest of the thesis is structured as follows. In Chapter 2, we lay out the theoretical background and introduce the set of Vlasov-Maxwell equations solved in GENE. Furthermore, we briefly describe different numerical approaches for plasma simulations: Lagrangian, Eulerian, and Semi-Lagrangian methods. At the end of the chapter, we provide information on the GENE application and discuss its main types of operations.

The system of the phase space coordinates used in GENE and similar grid-based gyrokinetic codes is presented in Chapter 3. In the first part of the chapter, we explain how field-aligned coordinates are used in the position space and how they are discretized in GENE. The second part discusses the velocity space coordinates. Here, we explain the relations between the temperature radial variation, background distribution function, and the choice of range and resolution for the velocity space discretization grids. At the end of this chapter, we detail the problem of spatial temperature variation and present a solution based on transforming the velocity space coordinates.

The approach of block-structured grids, which we implemented and tested in GENE, is explained in Chapter 4. The blocking concept is first introduced for the parallel velocity coordinate and theoretically compared to the aforementioned method of the coordinate transformation. Then, our proposed approach is extended to include the magnetic moment coordinate: we explain how quadrature and gyro-averaging can be performed for block-structured grids. At the end of the chapter, we estimate the theoretical savings in the number of computational nodes achieved by the proposed grids, compared to the original regular grids.

The technical details and implementation of the block-structured grids are presented in Chapter 5. When these grids are ported to the already existing Eulerian gyrokinetic code, the developers have to take care of three types of modifications: prefactors, grid-block boundaries, and parallelization. We detail the implementations associated to these changes in corresponding sections.

The numeral results obtained with the proposed grids are discussed in Chapter 6. Most of the numerical experiments provided in the chapter serve to verify our method. Here, we use the term verification in a scientific computing context: we check whether the new numerical scheme functions correctly. To this purpose, we compare the block-

structured grids with the original reference grids and check whether corresponding results converge to the same values. The attained reduction in the number of grid points depends on the simulation scenario provided (especially on the temperature radial profile). In the test cases at hand, approximately ten times fewer grid points are required for nonlinear simulations carried out with the block-structured grids. Furthermore, we evaluate the performance gain due to the proposed technique. The speed-up found to scale almost reciprocal to the number of grid points.

We conclude and summarize the main results of the thesis in Chapter 7. Additional information on the pre and post processing tools is provided in Appendix A. The software development aspects applied throughout this work are briefly discussed in Appendix B.

2

FUNDAMENTALS OF GYROKINETIC PLASMA MODELING

In this chapter, we consider fusion plasma in magnetic confinement devices, which is known to be hot and dilute. We first mention the main properties of fusion plasma.

To begin with, due to the high temperatures (around 150 MK in the core plasma), the major constituent plasma particles (deuterium, tritium, and helium) are fully ionized. The same holds for the impurities with an atomic number smaller than that of argon. This means that ionization and neutralization processes can be ignored. Furthermore, the relativistic effects can also be neglected in most of the cases, since, in spite of the high temperatures, the rest energy of the electron (500 keV) is still much larger than its thermal energy (around 13 keV).

Collisions in fusion plasma are rare and play a secondary role (while still being important in several other processes). The low collisionality hinders a macroscopic modeling of plasma in the real physical space (three position coordinates). Unlike in typical fluids, particles have long mean free paths of the same scale as the macroscopic observation length. As a result, particles carry information about their initial velocity for considerable distances. Therefore, fusion plasma is usually treated in a phase space consisting of position and velocity coordinates.

The mathematical models describing the behavior of plasma can be roughly subdivided in three categories:

- *Models considering individual particles* take into account the motion of all constituent particles, which are represented by point charges and point masses.

  The simplest model in this category describes the motion of an individual particle (mass m, charge q) in an electromagnetic field. In the cases of interest, the gravitational force is negligible in comparison to the Lorentz force resulting from the electric field $\mathbf{E}$ and magnetic field $\mathbf{B}$. Therefore, the non-relativistic equation of particle motion is given by

  $$m\frac{d\mathbf{v}}{dt} = q\left(\mathbf{E} + \mathbf{v} \times \mathbf{B}\right) . \tag{2.1}$$

  In the case of a uniform magnetic field and no electric field, this equation leads to charged particles moving on helix trajectories, with magnetic field lines as axes. On the plane perpendicular to the magnetic field, the motion is circular and usually referred to as cyclotron motion or gyration. The corresponding angular frequency of gyration is called Larmor frequency and is given by

  $$\Omega = \frac{|q|B}{m} . \tag{2.2}$$

  Although the individual particle description does not include the collective behavior (interaction through self-generated electromagnetic fields), this elementary description is already very helpful. For example, it provides explanations on the different types of drifts occurring due to the natural reasons in the magnetically confining devices such as:

  - the presence of an electric field orthogonal to the magnetic field — $\mathbf{E} \times \mathbf{B}$ drift
  - the curved magnetic field lines — curvature drift
  - the gradients across magnetic field lines — ∇B drift
  - the oscillating electric field — polarization drift

  For more details on individual particle motion, we refer to classical textbooks on plasma physics, such as [40, 41].

  The most detailed description involving all individual particle motions and interactions between them is represented by the Klimontovich-Maxwell system of equations. The Klimontovich equation describes the evolution of a distribution function in the phase space of all particles. The electromagnetic fields produced by all discrete charges in the position space are computed by the Maxwell equations. The distribution function is parameterized by initial positions and velocities of all constituent particles.

  Although the Klimontovich-Maxwell equations describe non-relativistic plasmas completely, when computer simulations are concerned, they are useless for two reasons:

  - the simulations would become computationally too expensive, due to an extremely high number of particles necessary to predict relevant physical processes
  - the simulation results would contain too many details related to discreteness of plasma

  The Klimontovich-Maxwell system is, nevertheless, insightful when deriving the continuum set of equations.

- *Continuum kinetic models* also contain many details, but less so than the previous models, so that computer simulations are feasible. They treat plasma as a phase space continuum. The electromagnetic fields are consistently computed based on a single averaged particle distribution function. If the collisions between particles are considered, the equation describing the evolution of the distribution function is referred to as the Boltzmann equation, or as the Vlasov equation otherwise.

  Both the Boltzmann and the Vlasov equations can be significantly simplified in the presence of a strong magnetic field. This situation occurs in magnetically confined plasma. For example, the fusion plasma turbulence responsible for strong outward particle and heat transport is characterized by frequencies much lower than the gyration or Larmor frequency. Therefore, the low frequency turbulence in plasma is modeled in the five dimensional phase-space, where the information about the cyclotron motion is neglected and the motion of gyrocenters is taken into account.

- *Fluid-like models* are derived from the kinetic description by marginalizing out the velocity components in the distribution function and, thus, neglecting kinetic effects, such as wave-particle interactions. The resulting equations correspond to mass, momentum, and energy conservation and describe the evolution of macroscopic quantities, such as density, fluid velocity, temperature, and the electromagnetic fields. The overall model is similar to an ordinary fluid description.

The macroscopic models are further subdivided in two types:

- *Two component fluid:* one component corresponding to electrons, another to ions. The model is used, for example, to derive an electrostatic response of plasma.
- *One component fluid:* by assuming a local charge neutrality, plasma can be treated as a homogeneous fluid (mass dominated by ions, current carried by electrons). This model is usually referred to as magnetohydrodynamics and is widely used to model plasma in strong inhomogeneous magnetic fields.

Since our project addresses gyrokinetic simulations, in the next section we provide a brief discussion of the underlying mathematical model. We start from the most detailed model described by the Klimontovich-Maxwell equations, which we use further to derive the Vlasov-Maxwell equations. Then we discuss the assumptions on which the gyrokinetic theory is based and describe the steps necessary to derive the final equations solved in GENE (for the simulation results see Chapter 6). Different numerical approaches to treat gyrokinetic equations and associated scientific codes are presented in Section 2.2. In Section 2.3, we shortly discuss the main features of the GENE code, for which our method of block-structured grids is applied. Many theoretical aspects presented in this chapter are useful for a better understanding of the rest of the thesis.

2.1 TOWARDS THE GYROKINETIC MODEL

In this section, we follow the approach of deriving the Vlasov equation presented in [41]. The advantage of this approach is that it clearly demonstrates the connection between the model considering the individual particle motion (the Klimontovich-Maxwell equations) and the continuum kinetic description (the Vlasov-Maxwell equations). Because the Klimontovich-Maxwell and Vlasov-Maxwell equations have a similar structure, in order to distinguish between the various quantities, those in the Klimontovich model are marked by a prime sign, e. g., $F'_s, \mathbf{E}', \mathbf{B}', \ldots$ for the distribution function and the electromagnetic fields, while those in the Vlasov model are written without prime, e. g., $F_s, \mathbf{E}, \mathbf{B}, \ldots$. In this way, we preserve an explicit link between these two models.

2.1.1 *Klimontovich-Maxwell System of Equations*

We start with the definition of the probability distribution function $f'_s(\mathbf{r}, \mathbf{v}, t)$ in the phase space $(\mathbf{r}, \mathbf{v})$ for one particle belonging to species s. The probability to find the particle in the phase space region $([\mathbf{r}, \mathbf{r} + d\mathbf{r}], [\mathbf{v}, \mathbf{v} + d\mathbf{v}])$ is $f'_s(\mathbf{r}, \mathbf{v}, t)\, d^3r\, d^3v$. If we know the trajectory in the phase space ($\mathbf{r}_i(t) = (x_i(t), y_i(t), z_i(t))$ and $\mathbf{v}_i(t) = (v_{xi}(t), v_{yi}(t), v_{zi}(t))$) of a given particle i, then the corresponding distribution function can be expressed in terms of Dirac delta functions:

$$\begin{aligned} f'_{si}(\mathbf{r}, \mathbf{v}, t) &= \delta(x - x_i(t))\delta(y - y_i(t))\delta(z - z_i(t)) \\ &\times \delta(v_x - v_{xi}(t))\delta(v_y - v_{yi}(t))\delta(v_z - v_{zi}(t)) = \delta(\mathbf{r} - \mathbf{r}_i(t))\delta(\mathbf{v} - \mathbf{v}_i(t))\,. \end{aligned} \tag{2.3}$$

We use the last expression to find a distribution function $F'_s(\mathbf{r},\mathbf{v},t)$ for all N_s particles of species s in the system. This is the sum of the individual distribution functions:

$$F'_s(\mathbf{r},\mathbf{v},t) = \sum_{i=1}^{N_s} f'_{si}(\mathbf{r},\mathbf{v},t) = \sum_{i=1}^{N_s} \delta(\mathbf{r}-\mathbf{r}_i(t))\delta(\mathbf{v}-\mathbf{v}_i(t))\,. \tag{2.4}$$

Note that, in the last expression, the distribution function is not normalized to one, but the number of particles N_s is given by its integral over the entire phase space.

The time derivative of the collective distribution function is expressed by

$$\frac{\partial F'_s}{\partial t} = \sum_{i=1}^{N_s}\left(\frac{\partial F'_s}{\partial \mathbf{r}_i}\frac{d\mathbf{r}_i}{dt} + \frac{\partial F'_s}{\partial \mathbf{v}_i}\frac{d\mathbf{v}_i}{dt}\right). \tag{2.5}$$

In this expression, we apply the following substitutions:

$$\frac{d\mathbf{r}_i}{dt} = \mathbf{v}_i\,, \tag{2.6}$$

$$\frac{d\mathbf{v}_i}{dt} = \frac{q_s}{m_s}\left(\mathbf{E}'(\mathbf{r}_i,t) + \mathbf{v}_i\times\mathbf{B}'(\mathbf{r}_i,t)\right) = \mathbf{W}'_s(\mathbf{r}_i,\mathbf{v}_i,t)\,, \tag{2.7}$$

where the first substitution corresponds to the velocity definition and the second to the acceleration caused by the Lorentz force. With these substitutions, the time derivative of the distribution function takes the form

$$\frac{\partial F'_s}{\partial t} = \sum_{i=1}^{N_s}\left(\frac{\partial F'_s}{\partial \mathbf{r}_i}\mathbf{v}_i + \frac{\partial F'_s}{\partial \mathbf{v}_i}\mathbf{W}'_s(\mathbf{r}_i,\mathbf{v}_i,t)\right). \tag{2.8}$$

This expression can be further simplified by taking into account the following properties of the Dirac delta function: $\delta(\mathbf{x}-\mathbf{x}_i)f(\mathbf{x}_i) = \delta(\mathbf{x}-\mathbf{x}_i)f(\mathbf{x})$ and $[\partial\delta(\mathbf{x}-\mathbf{x}_i)/\partial t]f(\mathbf{x}_i) = [\partial\delta(\mathbf{x}-\mathbf{x}_i)/\partial t]f(\mathbf{x})$, where in our particular case $f:\mathbb{R}^3\mapsto\mathbb{R}^3$. Then, because F'_s is expressed as a sum of Dirac delta functions, we obtain

$$\frac{\partial F'_s}{\partial t} = \sum_{i=1}^{N_s}\left(\frac{\partial F'_s}{\partial \mathbf{r}_i}\mathbf{v} + \frac{\partial F'_s}{\partial \mathbf{v}_i}\mathbf{W}'_s(\mathbf{r},\mathbf{v},t)\right). \tag{2.9}$$

From the definition of the distribution function (2.4), it follows that

$$\sum_{i=1}^{N_s}\frac{\partial F'_s}{\partial \mathbf{r}_i} = -\frac{\partial F'_s}{\partial \mathbf{r}}\,,\qquad \sum_{i=1}^{N_s}\frac{\partial F'_s}{\partial \mathbf{v}_i} = -\frac{\partial F'_s}{\partial \mathbf{v}}\,. \tag{2.10}$$

Substituting the latter expressions into (2.9) yields the Klimontovich equation:

$$\frac{\partial F'_s}{\partial t} + \mathbf{v}\frac{\partial F'_s}{\partial \mathbf{r}} + \frac{q_s}{m_s}\left(\mathbf{E}'(\mathbf{r},t) + \mathbf{v}\times\mathbf{B}'(\mathbf{r},t)\right)\frac{\partial F'_s}{\partial \mathbf{v}} = 0\,. \tag{2.11}$$

The fields in (2.11) are computed using the Maxwell equations:

$$\begin{aligned}
\nabla\times\mathbf{E}'(\mathbf{r},t) &= -\frac{\partial\mathbf{B}'(\mathbf{r},t)}{\partial t}\,,\\
\nabla\times\mathbf{B}'(\mathbf{r},t) &= \frac{1}{c^2}\frac{\partial\mathbf{E}'(\mathbf{r},t)}{\partial t} + \mu_0\mathbf{J}'(\mathbf{r},t)\,,\\
\nabla\cdot\mathbf{B}'(\mathbf{r},t) &= 0\,,\\
\nabla\cdot\mathbf{E}'(\mathbf{r},t) &= \frac{1}{\varepsilon_0}\rho'(\mathbf{r},t)\,,
\end{aligned} \tag{2.12}$$

where ε_0 and μ_0 are the permittivity and permeability of free space, and c denotes the speed of light in vacuum. The Maxwell equations are written for vacuum, because the charge $\rho'(\mathbf{r}, t)$ and current $\mathbf{J}'(\mathbf{r}, t)$ densities are computed by taking into account all the particles in the plasma:

$$\rho'(\mathbf{r}, t) = \sum_s q_s \int F'_s(\mathbf{r}, \mathbf{v}, t)\, d^3v\,, \tag{2.13}$$

$$\mathbf{J}'(\mathbf{r}, t) = \sum_s q_s \int \mathbf{v} F'_s(\mathbf{r}, \mathbf{v}, t)\, d^3v\,. \tag{2.14}$$

The structure of the Klimontovich-Maxwell system of equations yields self-consistent computations of the distribution function and fields: given the distribution functions for each species $F'_s(\mathbf{r}, \mathbf{v}, t)$, the charge $\rho'(\mathbf{r}, t)$ and current $\mathbf{J}'(\mathbf{r}, t)$ densities are evaluated, and the results are used in the Maxwell equations to compute the electric $\mathbf{E}'(\mathbf{r}, t)$ and magnetic $\mathbf{B}'(\mathbf{r}, t)$ fields, which, in return, are used to find the Lorentz forces and solve the Klimontovich equation.

To solve the Klimontovich-Maxwell integro-differential equations, the boundary and initial conditions have to be specified. The boundary conditions are usually determined by settings such as external magnetic fields. The initial conditions are given by all the initial positions and velocities of every constituent particle. In total there are $\sum_s 6N_s$ initial conditions. Consequently, computer simulations on the level of a Klimontovich description are infeasible, since there are too many particles in the system. However, simulating the Vlasov-Maxwell equations is possible by smearing out the discreteness in the Klimontovich distribution F'_s.

2.1.2 *Vlasov-Maxwell System of Equations*

The Vlasov-Maxwell system of equations is obtained using the formal procedure of averaging the Klimontovich-Maxwell equations over the ensemble of initial conditions $\{\mathbf{r}_i, \mathbf{v}_i\}$. We note here that the averaging procedure is not always rigorously defined. For example, if a system is not in thermal equilibrium, there is no definition for the average ensemble. Therefore, the averaging procedure can be considered as smearing out the discreteness of the Klimontovich distribution function F'_s to obtain a smooth Vlasov distribution function F_s. This is schematically shown in Figure 3.

We denote the averaged quantities according to the following expressions:

$$\begin{aligned} F_s(\mathbf{r}, \mathbf{v}, t) &= \left\langle F'_s(\mathbf{r}, \mathbf{v}, t; \{\mathbf{r}_i, \mathbf{v}_i\}) \right\rangle\,, & \mathbf{W}_s(\mathbf{r}, \mathbf{v}, t) &= \left\langle \mathbf{W}'_s(\mathbf{r}, \mathbf{v}, t; \{\mathbf{r}_i, \mathbf{v}_i\}) \right\rangle\,, \\ \mathbf{E}_s(\mathbf{r}, t) &= \left\langle \mathbf{E}'_s(\mathbf{r}, t; \{\mathbf{r}_i, \mathbf{v}_i\}) \right\rangle\,, & \mathbf{B}_s(\mathbf{r}, t) &= \left\langle \mathbf{B}'_s(\mathbf{r}, t; \{\mathbf{r}_i, \mathbf{v}_i\}) \right\rangle\,, \\ \rho_s(\mathbf{r}, t) &= \left\langle \rho'_s(\mathbf{r}, t; \{\mathbf{r}_i, \mathbf{v}_i\}) \right\rangle\,, & \mathbf{J}_s(\mathbf{r}, t) &= \left\langle \mathbf{J}'_s(\mathbf{r}, t; \{\mathbf{r}_i, \mathbf{v}_i\}) \right\rangle\,. \end{aligned} \tag{2.15}$$

The listed averaged terms appear linearly in the Maxwell equations (2.12), and expressions for the charge (2.13) and current (2.14) computations. Therefore, these equations do not change structurally for the Vlasov formalism. It is only the original quantities (with primes) that have to be replaced by their averaged counterparts.

The situation differs for the nonlinear term in the Klimontovich equation (2.11), since

$$\mathbf{W}_s \frac{\partial F_s}{\partial \mathbf{v}} \neq \left\langle \mathbf{W}'_s \frac{\partial F'_s}{\partial \mathbf{v}} \right\rangle\,. \tag{2.16}$$

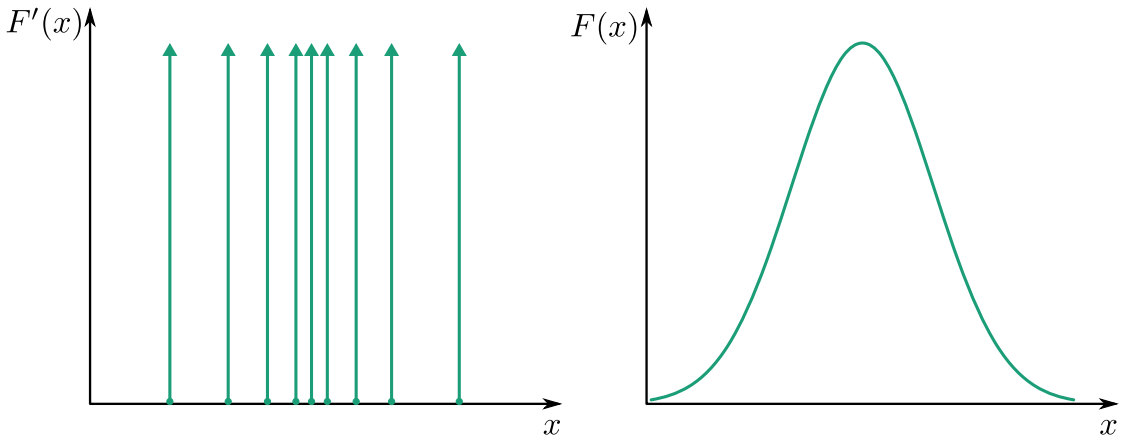


Figure 3: A schematic representation of one-dimensional Klimontovich $F'(x)$ and Vlasov $F(x)$ distribution functions. In the left plot, the Dirac delta functions are represented by green lines with arrow ends.

The difference of these two unequal terms is usually called a two-body correlation term (or collisional term), which represents the discreteness of plasma.

The two-body correlation term is negligibly small in many physical scenarios and, thus, can be neglected. In this case, the Klimontovich equation (2.11) is replaced by the structurally identical Vlasov equation:

$$\frac{\partial F_s}{\partial t} + \mathbf{v}\frac{\partial F_s}{\partial \mathbf{r}} + \frac{q_s}{m_s}\left(\mathbf{E}(\mathbf{r},t) + \mathbf{v}\times\mathbf{B}(\mathbf{r},t)\right)\frac{\partial F_s}{\partial \mathbf{v}} = 0\,. \tag{2.17}$$

If the collisional effect has to be considered, the two-body correlation term is approximated by the so-called collision operator, which appears in the right hand side of (2.17). The expression is then called the Boltzmann equation. In comparison to the Klimontovich equation, the initial condition of the Vlasov or Boltzmann equation is given by some initial smooth distribution function, instead of the positions and velocities of all constituent particles.

The Vlasov-Maxwell equations are already realizable for computer simulations. However, due to the high dimensionality of the problem, such simulations are possible only for a limited number of physical scenarios. A credible discretization of the six-dimensional phase-space for many scenarios yields a prohibitively large number of degrees of freedom. This is often the case for magnetically confined fusion plasma. Fortunately, due to a strong external magnetic field in such plasmas, the full Vlasov model can be reduced to a five-dimensional gyrokinetic model, which we briefly discuss in the next subsection.

2.1.3 *Gyrokinetic System of Equations*

The magnetic confinement devices necessitate a very strong magnetic field to isolate hot fusion plasma from the vessel walls. For example, according to [42], the magnets in the ITER experimental reactor are designed to generate magnetic fields up to 11.8 T.

A strong magnetic field results in a fast cyclotron motion of charged particles. The corresponding Larmor frequencies exceed the characteristic frequencies of the rela-

tively slow microturbulence significantly, which causes a cross-field transport of particles and heat. The particles gyration can be analytically removed from the Vlasov equation. The simplest approach to this purpose is to derive the Vlasov equation for the guiding center coordinates $\{\mathbf{X}, v_{\parallel}, \mu\}$, where $\mathbf{X}$ is the guiding center position (center of circular gyro-trajectory), $v_{\parallel}$ is the guiding center velocity component parallel to the equilibrium magnetic field, and $\mu = mv_{\perp}^2/2B$ is the magnetic moment. In the guiding center approximation, the gyration is averaged out by taking electromagnetic fields at the guiding center position. The resulting equations are called drift kinetic equations; their derivation can be found, for example, in [41].

A more sophisticated approach averages out the electromagnetic field over the gyro-angle, instead of taking the field at the guiding center position. The model based on this approach is described by gyrokinetic equations and, as it predicts the levels of turbulent transport accurately, it is used very often to describe the microturbulence in fusion plasma. For the modern formulation of the gyrokinetic theory, we refer to [30].

The experimentally observed gyrokinetic ordering in $\epsilon_g \ll 1$ in core plasma microturbulence is taken into account in the gyrokinetic approximation:

$$\frac{\omega}{\Omega} \sim \frac{\rho}{L_B} \sim \frac{k_{\parallel}}{k_{\perp}} \sim \frac{\delta n}{n} \sim \frac{B_1}{B_0} \sim \frac{v_E}{v_T} \sim \frac{f_1}{F_0} \sim \mathcal{O}(\epsilon_g)\,, \tag{2.18}$$

where ω is the characteristic frequency of plasma microturbulence, Ω is the Larmor frequency (2.2), ρ is the Larmor radius, L_B is a characteristic scale length of the magnetic field, $k_{\parallel}$ and $k_{\perp}$ are the components of the wave number $\mathbf{k}$ parallel and perpendicular to the equilibrium magnetic field, B_0 and B_1 are the moduli of the perturbed and equilibrium magnetic fields, v_E is the perturbed $\mathbf{E}\times\mathbf{B}$ drift speed, v_T is the thermal speed, f_1 and F_0 are the perturbed and equilibrium distribution functions.

The derivation of the gyrokinetic equations solved in GENE are detailed in [43, 44]. The two major mathematical operations of removing the gyration from the Vlasov equation are the averaging over the gyro-phase angle θ defined by

$$\overline{A}(\mathbf{X}) \equiv \frac{1}{2\pi}\oint A(\mathbf{X}+\mathbf{r}(\theta))\,d\theta\,, \tag{2.19}$$

and the near identity Lie transformations [30, 45].

The five-dimensional gyrokinetic equation (for species s) is expressed by

$$\frac{\partial F_s}{\partial t} + \frac{d\mathbf{X}}{dt}\cdot\boldsymbol{\nabla}F_s + \frac{dv_{\parallel}}{dt}\frac{\partial F_s}{\partial v_{\parallel}} + \frac{d\mu}{dt}\frac{\partial F_s}{\partial \mu} = 0\,. \tag{2.20}$$

The time derivatives of the guiding center coordinates are given by

$$\begin{aligned}
\frac{d\mathbf{X}}{dt} &= v_{\parallel}\mathbf{b}_0 + \frac{B_0}{B^*_{0\parallel}}\left(\mathbf{v}_E + \mathbf{v}_{\nabla B} + \mathbf{v}_c\right)\,,\\
\frac{dv_{\parallel}}{dt} &= -\frac{d\mathbf{X}/dt}{m_s v_{\parallel}}\cdot\left(q_s\boldsymbol{\nabla}\overline{\phi}_1 + \frac{q_s}{c}\frac{\partial\overline{A}_{1\parallel}}{\partial t}\mathbf{b}_0 + \mu\boldsymbol{\nabla}(B_0 + \overline{B}_{1\parallel})\right)\,,\\
\frac{d\mu}{dt} &= 0\,,
\end{aligned} \tag{2.21}$$

where B_0 denotes the modulus of the equilibrium magnetic field vector $\mathbf{B}_0$, $\mathbf{b}_0 = \mathbf{B}_0/B_0$ the corresponding unit vector, $B^*_{0\parallel} = \mathbf{b}_0\cdot\mathbf{B}^*_0$ the parallel component of $\mathbf{B}^*_0 =$

$\mathbf{B}_0 + \nabla \times (\mathbf{B}_0 v_\parallel / \Omega_s)$, $\overline{B}_{1\parallel}$ is the gyro-averaged modulus of the parallel component of the perturbed magnetic field $\mathbf{B}_1$, $\overline{\phi}_1$ is the gyro-averaged perturbed part of the electrostatic potential, $\overline{A}_{1\parallel}$ is the gyro-averaged modulus of the parallel component of the perturbed vector potential, and $\Omega_s = q_s B_0 / m_s c$ is the gyrofrequency of species s with mass m_s and charge q_s.

The time derivative of the guiding center position ($d\mathbf{X}/dt$) includes three characteristic drift terms:

- Generalized $\mathbf{E} \times \mathbf{B}$ drift velocity:

$$\mathbf{v}_E = \frac{c}{B_0^2} \mathbf{B}_0 \times \nabla \overline{\chi}_1 \,. \tag{2.22}$$

- Gradient-B drift velocity:

$$\mathbf{v}_{\nabla B_0} = \frac{\mu c}{q_s B_0^2} \mathbf{B}_0 \times \nabla B_0 \,. \tag{2.23}$$

- Curvature drift velocity:

$$\mathbf{v}_c = \frac{v_\parallel^2}{\Omega_s} (\nabla \times \mathbf{b}_0)_\perp = \frac{v_\parallel^2}{\Omega_s} \left(\mathbf{b}_0 \times \left[\frac{\boldsymbol{\nabla} B_0}{B_0} + \frac{\beta_p}{2} \frac{\boldsymbol{\nabla} p_0}{p_0} \right] \right) , \tag{2.24}$$

  where p_0 is the thermal plasma pressure, $\beta_p = 8\pi p_0 / B_0^2$.

We introduce the gyro-averaged modified potential in the $\mathbf{E} \times \mathbf{B}$ drift velocity:

$$\begin{aligned} \overline{\psi}_1 &= \overline{\phi}_1 - \frac{v_\parallel}{c} \overline{A}_{1\parallel} \,, \\ \overline{\chi}_{1s} &= \overline{\psi}_1 + \frac{\mu}{q_s} \overline{B}_{1\parallel} \,. \end{aligned} \tag{2.25}$$

The described gyrokinetic Vlasov equation (2.20) is the starting point of the so-called full-F codes. Despite the significant simplification leading to a reduced number of dimensions in the phase-space, this equation is still very challenging to solve numerically, mostly because the full-F equation describes physical features at different time scales. For example, it considers the plasma currents responsible for the evolution of the magnetic flux surfaces. This evolution is much slower than the plasma microturbulence and can be treated with MHD models in many cases.

To remove those time scales irrelevant for the plasma microturbulence and, hence, make the physical model less demanding in computational resources, the full distribution function F is split into a static background F_0 and a fluctuating f_1 distribution function, so that $F = F_0 + f_1$. For details on this technique, consult [46, 47]. The gyrokinetic codes applying the splitting technique are generally referred to as δf codes. The GENE code also belongs to this class. The discussion of the background distribution function F_0 used in GENE is postponed to Subsection 3.3.1, since it plays an important role in the construction of the velocity space grids.

The Vlasov equation for the split distribution function is obtained by substituting the full F by $F_0 + f_1$ sum, explicitly computing derivatives of the background part F_0, and keeping only those terms up to the first order of the gyrokinetic ordering ϵ_g. These

steps are detailed in the following works [43, 44]. Here, we provide the resulting δf Vlasov equation solved in GENE:

$$\begin{aligned}\frac{\partial g_{1s}}{\partial t} = &\underbrace{\frac{c}{C}\frac{B_0}{B^*_{0\parallel}}\partial_y \overline{\chi}_{1s}\partial_x^{n,T}F_{0s}}_{\text{drive term}} \underbrace{- \frac{c}{C}\frac{B_0}{B^*_{0\parallel}}\left(\partial_x\overline{\chi}_{1s}\Gamma_{ys} - \partial_y\overline{\chi}_{1s}\Gamma_{xs}\right)}_{\mathbf{E}\times\mathbf{B}\text{ nonlinear term}} \\ &+ \underbrace{\frac{c}{C}\frac{B_0}{B^*_{0\parallel}}\frac{\mu B_0 + m_s v_\parallel^2}{q_s B_0}\left(K_x\Gamma_{xs} - K_y\Gamma_{ys}\right)}_{\text{curvature term}} \underbrace{- \frac{c}{C}\frac{B_0}{B^*_{0\parallel}}\frac{m v_\parallel^2}{q_s}\frac{\beta}{2}\frac{\partial_x p}{p}\Gamma_{ys}}_{\text{pressure term}} \\ &- \underbrace{\frac{C}{JB_0}v_\parallel\Gamma_{zs}}_{\text{parallel dynamics}} + \underbrace{\frac{\mu}{m_s}\frac{C}{JB_0}\partial_z B_0\frac{\partial f_{1s}}{\partial v_\parallel}}_{\text{trapping term}} + \underbrace{\frac{c}{C}\frac{B_0}{B^*_{0\parallel}}\frac{\mu B_0 + m_s v_\parallel^2}{qB_0}K_x\partial_x^{n,T}F_{0s}}_{\text{correction term (constant)}}.\end{aligned} \tag{2.26}$$

The provided equation is split into conventional terms with corresponding name tags. Furthermore, for the sake of convenience, we used the following notations:

$$\Gamma_s = \nabla f_{1s} + q_s\nabla\overline{\psi}_1\frac{F_{0s}}{T_{0s}}, \tag{2.27}$$

$$K_x = \left(\partial_y B_0 - \frac{g_{yz}}{g_{zz}}\partial_z B_0\right), \tag{2.28}$$

$$K_y = \left(\partial_x B_0 - \frac{g_{xz}}{g_{zz}}\partial_z B_0\right), \tag{2.29}$$

$$\partial_x^{n,T}F_{0s} = \left[\partial_x \ln n_s + \partial_x \ln T_s\left(\frac{E}{T_{0s}} - \frac{3}{2}\right)\right]F_{0s}. \tag{2.30}$$

In (2.26), to avoid having two time derivatives (one for the distribution function f_{1s} and another for the perturbed magnetic potential $\overline{A}_{1\parallel}$), we introduced the modified distribution function, which is defined by

$$g_{1s} = f_{1s} - \frac{q_s}{m_s c}\frac{\partial F_{0s}}{\partial v_\parallel}\overline{A}_{1\parallel} = f_{1s} + \frac{q_s v_\parallel}{c}\frac{F_{0s}}{T_{0s}}\overline{A}_{1\parallel}. \tag{2.31}$$

Expressions (2.26,2.28,2.29) include notations related to the choice of a curvilinear coordinate system: C is the radially dependant constant length factor, J is the Jacobian, while $g_{xz}, g_{yz}, g_{zz}, \ldots$ are the metric coefficients. For more details on the curvilinear coordinates, see Section 3.2.

The perturbed electrostatic ϕ_{1s} and vector $\mathbf{A}_{1s}$ potentials are self-consistently computed from the Maxwell equations:

$$-\boldsymbol{\nabla}^2\cdot\phi_1 = 4\pi\sum_s q_s n_{1s}, \tag{2.32}$$

$$-\boldsymbol{\nabla}^2\cdot\mathbf{A}_1 = \frac{4\pi}{c}\sum_s q_s n_{1s}\mathbf{u}_{1s}, \tag{2.33}$$

where the Coulomb gauge is used, n_{1s} and $\mathbf{u}_{1s}$ are the density and fluid velocity perturbations.

The density $n_{1s}(\mathbf{x}) = M_{00}(\mathbf{x})$ is the zeroth order space moment of the distribution function, and the fluid velocity $\mathbf{u}_{1s}(\mathbf{x}) = M_{10}(\mathbf{x})$ is the first order velocity space mo-

ment. The definition of the ath moment in $v_{\parallel}$ and the bth moment in $v_{\perp}$ is given by

$$M_{ab}(\mathbf{x}) = \int \delta\left(\mathbf{X} - \mathbf{x} + \boldsymbol{\rho}\right) T^* f_1\left(\mathbf{X}, v_{\parallel}, \mu\right) \frac{B_{\parallel}^*\left(\mathbf{X}, v_{\parallel}\right)}{m} v_{\parallel}^a v_{\perp}^b \, d^3X \, dv_{\parallel} \, d\mu \, d\theta \,, \tag{2.34}$$

where T^* is a pull-back operator transforming the gyro-center distribution function (whose evolution is described by the gyrokinetic Vlasov equation) to the particle distribution function. The exact expression of the pull-back operator applied in GENE can be found in [43, 44].

In GENE, the computation of the moments is done formally in three stages: first, the integration over $v_{\parallel}$ is performed; then the following gyro-phase averaging

$$\langle f_1 \rangle = \frac{1}{2\pi} \int \delta\left(\mathbf{X} - \mathbf{x} + \boldsymbol{\rho}\right) f_1\left(\mathbf{X}\right) \, d^3\mathbf{X} \, d\theta \tag{2.35}$$

is done; finally, the quadrature over the magnetic moment coordinate is computed. This order is justified, because, due to the magnitude of the gyro-radius

$$|\boldsymbol{\rho}| = \frac{c}{q} \sqrt{\frac{2\mu m}{B}} \,, \tag{2.36}$$

the gyro-phase averaging (which is the most expensive stage from the three) is independent from $v_{\parallel}$ and depends only on μ. In the presented order, we first remove the $v_{\parallel}$-dependence by integrating over this coordinate; then we compute the gyro-phase averaging at each fixed μ from the set of given magnetic moment grid points (for example, Gauss-Laguerre nodes), and then integrate the gyro-averaged results over μ.

The existing gyrokinetic codes solve either the full-F or δf Vlasov-Maxwell set of equations. These codes are based on numerical approaches allowing to express the gyrokinetic theory as a computational model. These approaches are briefly discussed in the next section.

2.2 DIFFERENT APPROACHES TO GYROKINETIC PLASMA SIMULATIONS

The analytic methods of solving the five-dimensional gyrokinetic Vlasov equation (2.20) coupled with the three dimensional Maxwell equations (the Poisson equation for electric potential and the Ampere law — in case magnetic perturbations are considered) are applicable only for a few simplified cases. Hence, obtaining numerical solutions for the set of gyrokinetic equations is of paramount importance, the analytic methods serving usually as a verification tool for numerical codes.

Determining numerical solutions is, nevertheless, feasible only by using high performance computing, because typical gyrokinetic turbulence simulations are extremely demanding in computational resources. High performance computing codes for gyrokinetic simulations can be classified by the numerical schemes they are using. The highest level classification is based on the reference frame and comprises three types of schemes: Eulerian, Lagrangian, and semi-Lagrangian. These three types of numerical schemes are applicable not only to the gyrokinetic equations, but also to other types of mathematical models, for example, to the full kinetic description.

In order to lessen the computational demands of the gyrokinetic simulations, alternative gyrofluid methods have been developed. These techniques are similar to computational fluid dynamics: they do not work directly with the distribution function,

but operate instead on the velocity moments (corresponding to macroscopic quantities) of the gyrokinetic set of equations. The gyrofluid methods, however, encounter inevitably a closure problem (as plasma is almost collisionless). A popular solution is the Landau-fluid closure [48]. For more details on the gyrofluid models see [49–52]. Modeling plasma turbulence based on the fluid description is intricate and, to choose a proper closure, results from self-consistent gyrokinetic simulations may be used.

In the following subsections we briefly describe the numerical aspects of the three gyrokinetic numerical approaches (Eulerian, Lagrangian, and semi-Lagrangian) and provide references to several representative gyrokinetic high performance computing codes. For a more detailed overview on which physical phenomena are covered by these codes and associated challenges, we refer to [24, 25].

2.2.1 *Eulerian Scheme*

The Eulerian or grid-based approach applies a fixed reference frame. In other words, the differential and integral operators are discretized on a (usually structured) computational grid fixed in the phase space. Typical combinations of discretization schemes used to discretize derivatives involve: finite differences, finite volumes, spectral methods, and finite elements (for field solvers). Moreover, discrete versions of integral-like operators such as gyro-phase averaging apply different quadrature rules in conjunction with interpolation schemes (for details see Section 4.3). A schematic representation of a time iteration in an Eulerian code using a finite difference scheme is shown in Figure 4. In this scheme, we present the three steps characteristic to all applications in the Eulerian class:

1. First, a computational grid is constructed in the phase space, for more details see Chapters 3 and 4.
2. At each time step, the right hand-side of the gyrokinetic Vlasov equation is computed by applying different numerical schemes (in the figure, we demonstrated a simple finite difference stencil computation).
3. The distribution function is updated by means of an ODE[1] solver. For example, for explicit time integration schemes, Runge-Kutta methods are often used, see [53]. Then, the moments of the updated distribution function are computed, which serve as inputs to the field equations.
4. The field equations are solved and the time iteration is repeated starting from Step 2.

According to [24], the Eulerian approach is particularly good for a stable and accurate treatment of the $\mathbf{E} \times \mathbf{B}$ nonlinear term (see (2.26)). There are several gyrokinetic codes following this approach, among which GENE [35, 37], GS2 [54–56], GYRO [57, 58], GKV [59], GKW [60, 61], and GT5D [62]. These codes differ in the particular numerical methods they apply, the physical scenarios they support, and the operation modes they can work in. A short description and explanation of operation of the representative code GENE is provided in Section 2.3.

1 Ordinary Differential Equation

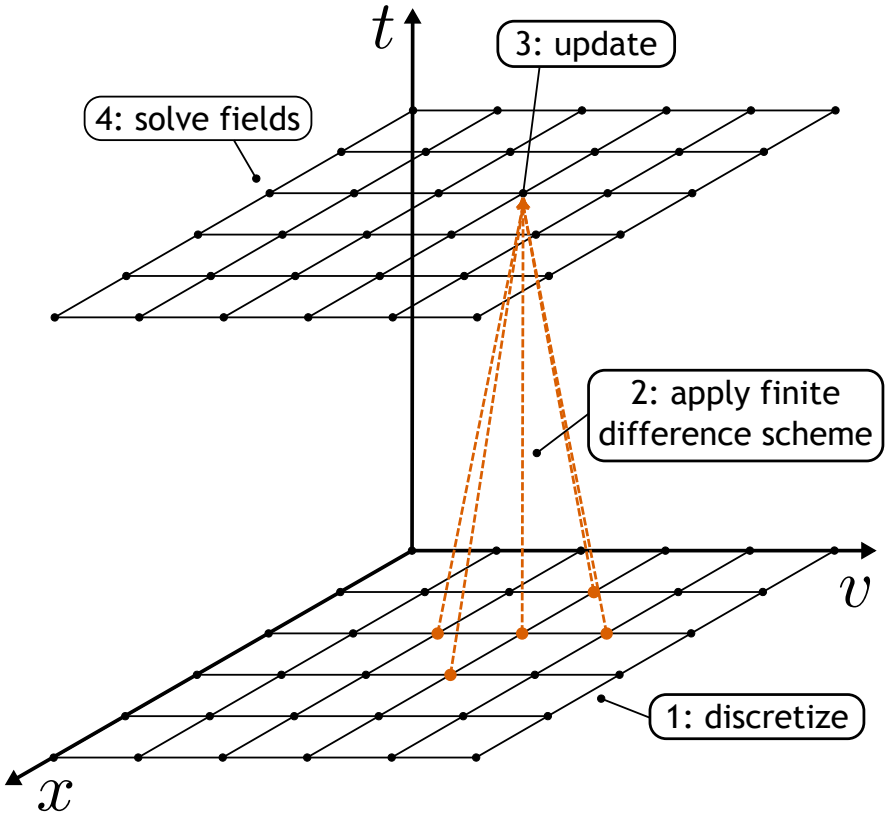


Figure 4: Schematic representation of a time step integration in an Eulerian code. A fixed grid is defined in the phase space: **x** — position space, **v** — velocity space. Finite difference schemes are often used to compute derivatives in the right hand-side of the Vlasov equation. After the new values of the distribution function are computed, the field equations are solved, which take the moments of the distribution function as input. Based on [24].

One of the weaknesses of the grid-based schemes is that they are subject to the CFL[2] stability condition [63]. However, this problem can be circumvented by applying implicit or semi-implicit time integration schemes, e. g., see [62]. Another disadvantage is related to the application of finite difference schemes, which introduce numerical dissipation. Nonetheless, this numerical dissipation helps achieve steady-state simulations for long time periods. It is, therefore, important to balance keeping numerical dissipation as small as possible and damping nonphysical oscillations (smoothness of the solution). For more details on numerical dissipation in GENE, see [64].

To parallelize the Eulerian codes, a domain decomposition approach is usually used, as discussed in Section 5.3.

2.2.2 *Lagrangian Scheme*

Another popular approach for numerical plasma simulations is based on solving the Vlasov equation in a Lagrangian frame. This method is often referred to as PIC[3]. The

2 Courant-Friedrichs-Lewy
3 Particle-in-Cell

name stems from the abstract super-particles used to model plasma behavior, which are assigned with plasma macro-quantities like number density, current density, etc. As physical systems contain extremely many real particles, representing them directly in computer simulations is infeasible and, furthermore, not necessary for predicting certain types of physical behavior. A super-particle (also called a marker) is a computational entity representing a cluster of real particles; for plasma simulations, it may correspond to millions of electrons or ions. The PIC technique for plasma physics is detailed in [65]. The first Lagrangian method to be applied to gyrokinetic simulations is described in [66].

We schematically represent a time step iteration for the Lagrangian method in Figure 5. Four characteristic actions are marked in this diagram:

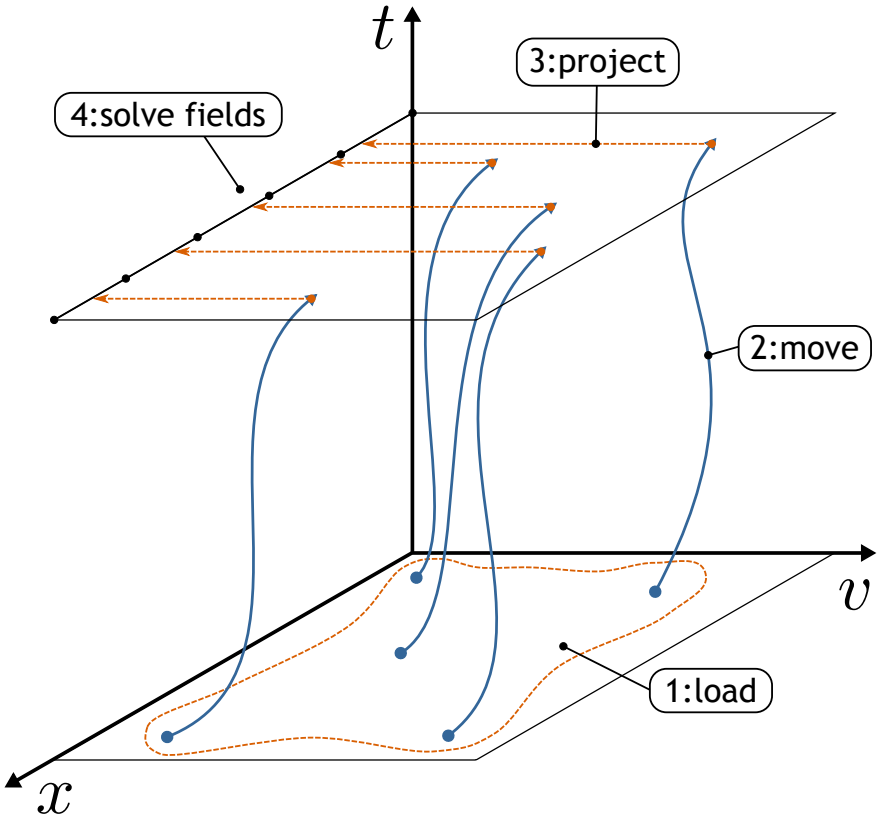


Figure 5: Schematic representation of the major steps performed in a Lagrangian scheme. At the beginning of a simulation, the markers are loaded. During time iterations, the following steps are cyclically repeated: the markers are moved, the field sources are computed, the field equations are solved. Based on [24].

1. The simulation is initialized by loading markers (sampling initial conditions). Choosing the distribution of the markers adequately in the phase space (more markers in regions with larger probability of perturbations) can significantly reduce statistical noise throughout simulations. From many aspects, this step is equivalent to choosing a proper phase space discretization grid in Eulerian codes.

2. Even though the super-particles represent clusters of real particles, in gyrokinetic simulations, their total number is still very high ($\sim 10^9$), in order to achieve an

acceptable signal/noise ratio. Therefore, moving markers is one of the most expensive steps during a time step iteration. To advance the particles explicit (using old field values) or implicit (requiring new field values) time schemes are used. An explicit scheme mover is usually much cheaper than an implicit one. However, it requires much smaller time steps to obtain credible simulations.

3. After the new positions of markers are determined, they are used to compute the source terms for fields (charge and current). To project the super-particles on the computational grid for fields, we assume that particles are of a given shape. Then, computations of macro-quantities (distribution function moments) correspond to weighted sums over shape functions of all markers.

4. At the end of a time iteration, the field equations are solved.

There is a number of gyrokinetic codes that successfully apply the Lagrangian approach, for example: GTC [67, 68], GT3D [69], ORB5 [70], PG3EQ [71], GTS [72, 73], XGC1 [74, 75], GEM [47, 76], ELMFIRE [77, 78], etc. An advantage of these codes in comparison to Eulerian implementations is that they are not subject to the CFL condition. However, the results obtained with PIC codes can be affected by statistical noise. The reason is that the charge and current computations are by nature similar to Monte Carlo integrations. Therefore, these procedures lead to an error proportional to $\sqrt{\gamma/N}$ (where N — number of markers, γ – variance of the estimator). This error tends to accumulate in the course of a simulation.

The simplest way to increase the signal/noise ratio in Lagrangian simulations is to enlarge the number of markers N. However, due to the CPU[4] time limitation, this is not always possible, because the nonlinear gyrokinetic simulations are extremely computationally expensive. Therefore, several methods have been applied to reduce the variance of the estimator (γ). One approach solves the gyrokinetic equations for the fluctuating part of the distribution function δf only. The full distribution is then given by $f = F + \delta f$, where F is the background distribution function, which is treated analytically (for example, see [79]). Another technique to reduce γ is introduced in [80]. The main idea is to use as much as possible analytic computations instead of the Monte Carlo integration in the estimations of the charge and current. Moreover, the statistical noise can be weakened by an intelligent loading of markers, so that there are more of them in the regions with strong perturbations. This approach is called importance sampling, for more details see [81].

The domain decomposition technique is also applicable in the position space for Lagrangian codes. Moreover, a domain cloning technique described in [82, 83] is often used. According to this method, each processor of a clone family is associated to the same part of the positional space (accessing the same field values), but operates on different markers.

2.2.3 *Semi-Lagrangian Scheme*

The last type of numerical schemes is called semi-Lagrangian. The first semi-Lagrangian methods were developed for simulating atmospheric flows; for a review in this

4 Central Processing Unit

context we refer to [84]. The pioneering works on the semi-Lagrangian approach for the magnetized plasma turbulence are described in [85–87].

The goal of the semi-Lagrangian approach is to take the best features from both schemes introduced previously, by removing the CFL restriction characteristic to the explicit Eulerian method and circumventing the problem of statistical noise inherent to the Lagrangian codes. The semi-Lagrangian codes incorporate elements of both Eulerian and Lagrangian approaches. For example, they use fixed grids for the Vlasov equation (like Eulerian codes do), but integrate this equation along trajectories where the distribution function is constant (similarly to Lagrangian codes). A schematic representation of a step iteration of a standard or backward semi-Lagrangian numerical scheme is demonstrated in Figure 6. In this simplified diagram we represent four typi-

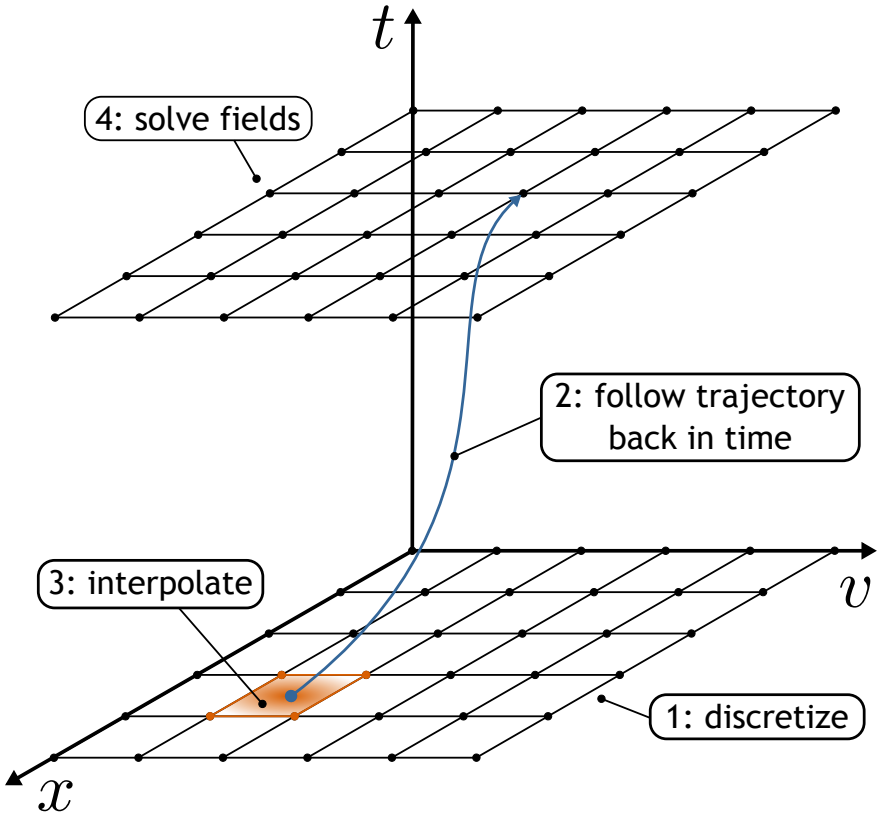


Figure 6: Schematic representation of the major steps in the backward semi-Lagrangian method. This method adopts a fixed phase space grid (like in the Eulerian approach) and integrates backward in time along trajectories where the distribution function is invariant to find the solution at each time step (similar to the Lagrangian approach). Based on [24].

cal steps:

1. Similarly to the Eulerian approach, a fixed grid is defined in the five dimensional space at the beginning of a semi-Lagrangian simulation.

2. In the backward semi-Lagrangian approach, the trajectories along which the distribution function is invariant start at each grid point and are propagated backward in time for one time step interval by solving advection equations.

3. Naturally, the characteristics (trajectories) do not end up exactly at grid points. Consequently, interpolation e. g., cubic spline interpolation, is performed to obtain updated values of the distribution function at fixed grid points.

4. After the distribution function values are updated, the fields are computed in a manner similar to the Eulerian scheme.

In addition to the backward semi-Lagrangian numerical approach presented here, other schemes have also been proposed: a forward semi-Lagrangian approach (see [88]) and a conservative semi-Lagrangian approach (see [89]). The difference between the backward and forward schemes is that, as the name suggests, the latter solves the advection equations forward in time. The conservative semi-Lagrangian method treats a conservative form of the Vlasov equation.

The semi-Lagrangian schemes have been proved to be accurate and efficient for low-dimensional systems. For high-dimensional problems, like the gyrokinetic model, a direct interpolation in the multidimensional space is, however, extremely expensive. This problem is addressed by the time-splitting procedure introduced in [90], in which the five dimensional Vlasov equation is replaced by a set of equations with lower dimensionality.

The main difficulty of the semi-Lagrangian method is to find an efficient and accurate interpolation scheme. A trade-off between numerical dissipation and validity of the results is necessary. For example, high order interpolation schemes are less dissipative, but tend to lead to spurious oscillations.

The development of the gyrokinetic semi-Lagrangian numerical schemes is still an active field of research. Therefore, there are not so many existing codes. One example of a mature code applying the described approach is GYSELA [91–93].

2.3 GENE — GYROKINETIC ELECTROMAGNETIC NUMERICAL EXPERIMENT

Since the proposed method of block-structured grids targets the Eulerian gyrokinetic code GENE, in this section, we shortly detail the major facets of this code. The project was first initiated by F. Jenko [35] and has since been continuously enhanced by introducing new physics, improving numerical methods, and optimizing the implementation for high performance computing. Meanwhile, GENE has become a part of the "Unified European Application Benchmark Suite" by PRACE, see [94, 95]. Its major parts are implemented in Fortran programming language, which is adequate for a massively parallel simulation code run on state-of-the-art supercomputers.

The integro-differential equations in GENE are solved by applying the method of lines. The essence of this method is to separate the discretization of the time from that of the phase space. The phase space is discretized in the following manner:

- Finite difference methods are used in combination of spectral methods to compute derivatives on a grid in the phase space.

- Integral-like operators represented by a composite quadrature rule (e. g., Simpson rule) in the parallel velocity direction and by a Gauss-Laguerre type quadrature in the magnetic moment direction are used to compute velocity moments for field equations.

After the discretization of the phase space is performed, the governing equation can be formally treated as an ODE. To integrate the time evolution of the distribution function for GENE, different time stepping algorithms have been examined, as described in [53, 96]. The fourth-order explicit Runge-Kutta method is currently used as a default time stepping scheme, as it proved well-suited for a wide range of scenarios.

In order to mitigate the computational requirements, the GENE code employs several well-established techniques targeting the mathematical model. The most prominent are:

- A δf-splitting technique: the fluctuating part of the distribution function is considered smaller than the equilibrium part from the gyrokinetic order point of view. The gyrokinetic equations are solved only for the perturbed part. The operations involving the background distribution are replaced by analytic computations when possible.

- Field-aligned coordinate system: the strong anisotropy of the turbulent fluctuations in the directions parallel and perpendicular to the background magnetic field is exploited in the choice of the computational grid for the position space. For more detailed explanations on the field-aligned coordinate system in GENE, consult Section 3.2.

Gyrokinetic turbulence codes can be classified by what type of operation modes they support: local or global. As the names suggest, the extent of the simulation domain is different in these types of operation modes. At the beginning, GENE was operating only in the local mode. Meanwhile, its capability was extended to also support more challenging global simulations, see [36]. Next, we briefly describe the local and global operation modes:

- *Local:* In the local or flux-tube simulations [97], the radial extent of the simulation domain is relatively small compared to the minor radius of the magnetic confinement device. The gyrokinetic equations are solved in the vicinity of a magnetic field line. The radial variations of the temperature and density are insignificant within the computational domain and can be neglected. This allows assigning periodic boundary conditions to both spacial directions parallel to the background magnetic field lines and using efficient spectral methods to discretize those directions.

- *Global:* To detect effects related to the radial changes of plasma parameters, the radial temperature and density profiles have to be taken into account. In this case, the grid radial range may extend up to the full machine size. Furthermore, the periodic boundary conditions in this direction become inadequate. Thus, instead of using the spectral method for the radial coordinate, e. g., a finite difference method can be used to allow setting an arbitrary Dirichlet or Neumann boundary condition. Another significant modification is in the construction of the field solver, which now requires involving finite element interpolation techniques when computing gyro-phase averaging. The changes are reflected in the computational grids. A straightforward approach of extending the equidistant regular grids of the local simulations to the full machine size comes at a high cost. For example, to capture fairly well the structures of the perturbed distribution function for all involved temperatures, the velocity space grids necessitate

wider ranges with finer resolutions (for more details on the velocity space resolution see [59, 98–100] and Chapters 3 and 4). One way to circumvent the described issues is to couple multiple local simulations, e. g., as realized in the gyrokinetic codes TRINITY and TGYRO [101, 102]. However, true global simulations become unavoidable when the turbulence correlation length approaches the scale lengths of the background temperature and density profiles.

In the rest of the thesis, we exclusively consider the global operation mode. To address the problem of expensive computational grids in the velocity space, we propose an efficient blocking method, for explanations see Chapter 4.

The GENE code was significantly enhanced throughout multiple PhD research projects. Hence, comprehensive information on different aspects of this simulation tool can be found in the following theses [32, 43, 44, 96, 103–107].

2.4 SUMMARY

In this chapter, we introduced notions fundamental to the following chapters. We started by briefly discussing the mathematical models of plasma, which can be subdivided into three categories: models considering individual particles, continuum kinetic models, and fluid-like models. Since our project is aimed at gyrokinetic simulations, we focused on the second category (kinetic models). The kernel of the kinetic model — the Vlasov-Maxwell equations — were derived based on the Klimontovich-Maxwell equations, which consider all constituent plasma particles. The kinetic description demands too high computational costs. Fortunately, based on the gyrokinetic ordering, which is valid for the core plasma microturbulence, the equations can be further simplified, resulting in the gyrokinetic model.

There are three major numerical approaches (Eulerian, Lagrangian, and semi-Lagrangian) of transforming the mathematical model into gyrokinetic simulations. We briefly introduced all these approaches and explained their main characteristics. Finally, we described the Eulerian code GENE.

In the next chapter, we introduce the five-dimensional phase space coordinates used in GENE and other Eulerian codes. We also detail the discretization of the coordinates in the position and velocity spaces. Furthermore, we explain how the background distribution function influences the construction of the computational grids in the velocity space and present the problem of the spatial variations in temperature, which results in a lot of grid points in the velocity directions.

3

DISCRETIZATION OF PHASE SPACE COORDINATES

The gyrokinetic Vlasov equation describes the evolution of the particle distribution function in a five dimensional space. Three of the coordinates form a position space and are denoted by (x, y, z), where x corresponds to the flux surface label, y to the binormal coordinate or field line label on a given flux surface, and z is the coordinate parallel to the magnetic field. For local GENE simulations, the flux surface label and binormal coordinate direction are discretized using a fast spectral method. For global GENE simulations, however, due to the Dirichlet boundary condition imposed in the radial direction, a finite difference method is used to discretize the flux surface label.

The remaining two coordinates, the parallel velocity $v_{\parallel}$ and the magnetic moment μ, describe the velocity space. The parallel velocity coordinate is discretized using a finite difference method. For the magnetic moment direction, however, no finite difference schemes are necessary in collisionless plasma simulations, see (2.26). In order to couple the Vlasov and Maxwell equations (and also for diagnostic purposes), only quadrature-like operations involving μ are required. Due to the exponential decay (3.32) of the background distribution in the magnetic moment direction, Gauss-Laguerre nodes are an optimal choice to perform such operations.

Table 1 summarizes the default grids used in GENE.

Table 1: GENE global version coordinates

symbol	name	space type	discretization
x	radial direction	position-configuration	equidistant
y, k_y	binormal direction	position-Fourier	equidistant
z	parallel direction	position-configuration	equidistant
$v_{\parallel}$	parallel velocity	velocity-configuration	equidistant
μ	magnetic moment	velocity-configuration	Gauss-Laguerre

This chapter consists of three sections. The first short section presents the normalization of the physical quantities and coordinates, which affect the grid construction, especially in the velocity space. The normalization is not necessary for understanding the theoretical background. However, it is useful for working with the pre-processing tool, which generates the grid. The grid generation tool is described in Appendix A.1. Section 3.2 deals with the position space coordinates. The basic concept of the field-aligned coordinates based on the Clebsch formulation are presented and demonstrated on a simple circular geometry model. Section 3.3 describes the velocity space coordinates and presents the problem of the spatial temperature variation.

3.1 NORMALIZATION

All coordinates and physical units used in GENE are normalized, in order to obtain physically meaningful scales, thus, making it simpler to recognize physical phenomena happening in simulations. During the post-processing step, the variables can be transformed back to physical units to allow comparisons with experimental data, etc.

The geometrical space perpendicular to the magnetic line directions x and y is scaled by the reference gyrokinetic radius ρ_{ref}, which is given by

$$\rho_{ref} = \frac{m_{ref} c_{ref} c}{e B_{ref}} . \tag{3.1}$$

The normalization of phase space coordinates and time is expressed by

$$\begin{aligned} x &= \hat{x}\rho_{ref} , & y &= \hat{y}\rho_{ref} , & z &= \hat{z} , \\ v_{\parallel} &= \hat{v}_{\parallel}\hat{v}_{T_s}\,|_{x_0}\, c_{ref} , & \mu &= \hat{\mu}\hat{T}_{0s}\,|_{x_0}\,\frac{T_{ref}}{B_{ref}} , & t &= \hat{t}\frac{L_{ref}}{c_{ref}} . \end{aligned} \tag{3.2}$$

Here, hats denote normalized quantities and $|_{x_0}$ indicates evaluation at the reference position x_0 in the middle of the simulation box, which usually corresponds to the middle of the fusion device's minor radius. The velocity space normalization is species dependent (marked by an s subscript, e. g., $\hat{v}_{T_s}$ and $\hat{T}_{0s}$ in (3.2)), because of potentially different temperatures among species. Furthermore, we use the following notations

$$c_{ref} = \sqrt{\frac{T_{ref}}{m_{ref}}} , \qquad \hat{v}_{T_s} = \sqrt{\frac{2\hat{T}_{0s}}{\hat{m}_s}} . \tag{3.3}$$

The macroscopic reference length L_{ref} is usually chosen to correspond the minor or major radius of a confinement device. In simulations with two species (electrons — e subscript and ions — i subscript), the standard choices for the reference temperature, density, and mass are

$$T_{ref} = T_e , \qquad n_{ref} = n_e , \qquad m_{ref} = m_i . \tag{3.4}$$

The dimensionless or normalized mass $\hat{m}_s$ is defined from the relation $m_s = m_{ref}\hat{m}_s$.

The profile quantities such as temperature $T_s(x)$ and density $n_s(x)$ radial dependencies are products of three terms. For example, the temperature is determined by

$$T_s(x) = T_{ref}\hat{T}_{0s}\hat{T}_s(x) . \tag{3.5}$$

Here, the first term defines the main scale, e. g., $T_{ref} = T_e$, while the second and third terms are dimensionless and correspond to the species temperature dependence at a reference position x_0 and the radial dependence of the profile.

3.2 POSITION SPACE COORDINATES

The governing gyrokinetic equation (2.26) is given in a Cartesian coordinate system, in which the z coordinate line is parallel to the magnetic field line $\mathbf{B}$, while the $x - y$ coordinate plane is orthogonal to it. In a complex magnetic field geometry, where magnetic

field lines are not straight lines, this Cartesian coordinate system is oriented differently at each point of the position space. Attempting to use a fixed Cartesian coordinate system in the gyrokinetic computer simulations would lead to a very high number of grid points in the position space, because this coordinate system would disregard the physical properties of the magnetically confined plasma. In a strongly magnetized plasma, the background magnetic field $\mathbf{B}$ causes highly anisotropic particle mobility, which, in turn, results in elongated turbulence structures in the direction of the magnetic field lines. Therefore, it is natural to use a curvilinear coordinate system instead of a Cartesian one. In this curvilinear coordinate system, one of the coordinate lines (we denote the corresponding coordinate by z) is aligned with the magnetic field lines, whereas the coordinate plane of the two other coordinates (we denote them by x and y) is orthogonal to the magnetic field $\mathbf{B}$. Therefore, this coordinate system is usually called a field-aligned coordinate system. The application of the field-aligned coordinate system to the gyrokinetic Vlasov-Maxwell equations is reflected in the introduction of metric coefficients and the Jacobian matrix into computations of derivatives and integrals, respectively (for the final form of these equations we refer to [43, 106]). Aligning the position space coordinates to the background magnetic field helps reduce the number of necessary grid points in the position space, compared to the fixed Cartesian coordinate system. According to estimates provided in [24, 108], the reduction of the grid points depends on the safety factor with aspect ratio and amounts to $10 - 100$ times due to the alignment of the coordinate system to the magnetic field lines.

A vast theoretical background and detailed explanations on the field-aligned (or flux) coordinates are provided in [109], while further details on the position space coordinates for gyrokinetic simulations are explained in [97, 110].

Next, we outline the procedure of obtaining the field-aligned coordinates for a toroidal axisymmetric magnetic geometry. Furthermore, we illustrate the procedure on a circular model often used for theoretical turbulence studies, e. g., see [43], which is represented schematically in Figure 7. We start with a general approach of the Clebsch system of coordinates, which was introduced for the first time in [111] for magnetic fusion applications, and is also used in GENE for the position space. The Gauss law for magnetism $\boldsymbol{\nabla} \cdot \mathbf{B} = 0$ allows us to write the magnetic field in the Clebsch form

$$\mathbf{B} = \boldsymbol{\nabla} x \times \boldsymbol{\nabla} y \,. \tag{3.6}$$

Because the divergence of the vector product is always zero, the Gauss law for magnetism is satisfied automatically. From the Clebsch form (3.6), it is apparent that $\mathbf{B} \cdot \boldsymbol{\nabla} x = 0$, which means that the magnetic field lines lie on a $x = \text{const}$ surface. The same conclusion can also be drawn for the y coordinate. Consequently, each magnetic field line corresponds geometrically to the intersection of two x and y constant surfaces. Conventionally, the x coordinate is called a flux surface label, and $x = \text{const}$ corresponds to a surface spanned by a magnetic field with an irrational safety factor (number of toroidal turns per one poloidal). An example of such a surface for the circular model is shown in Figure 7, which corresponds to the fixed minor radius r value — the distance from the flux surface to the magnetic axis. Therefore, for the circular model, the flux surface coordinate x is usually set to the minor radius r. For other geometries, the flux surface label increases monotonically with the minor radius. Therefore, from now on, we refer to the flux surface label x as a radial distance or radial coordinate. The second coordinate y is called a field line label.

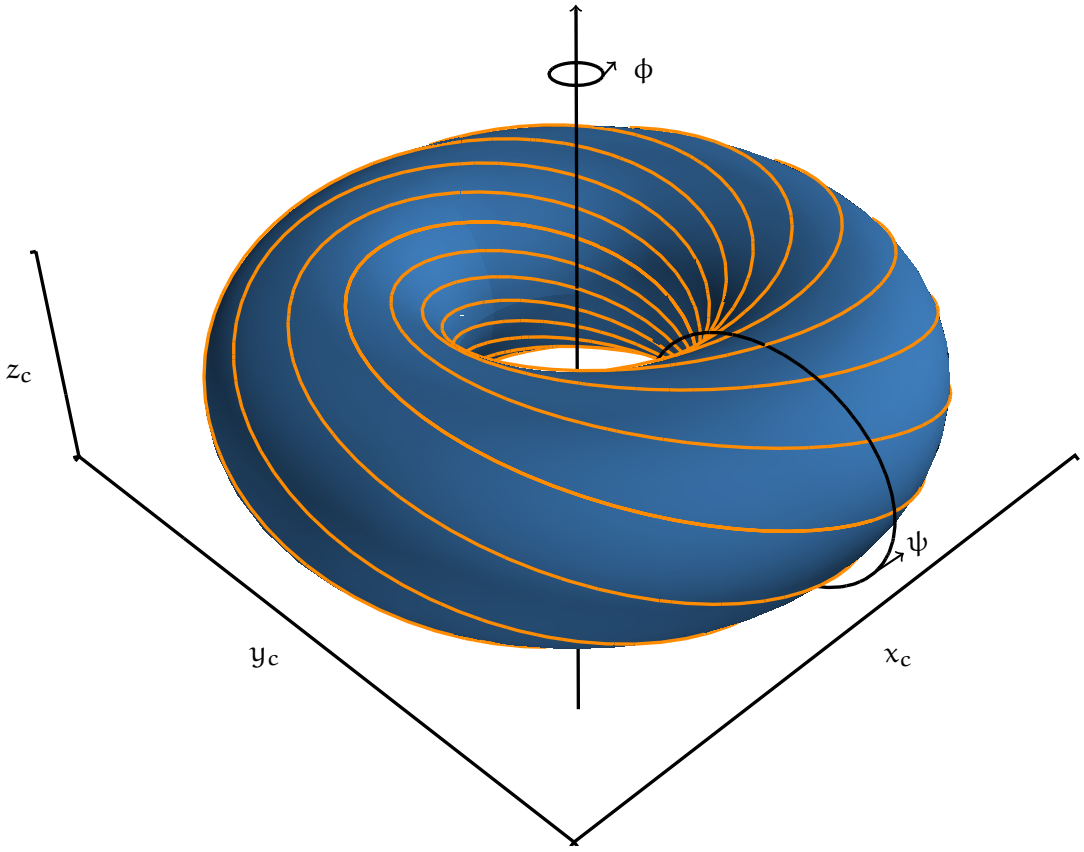


Figure 7: Magnetic geometry of a circular model. The blue surface represents a flux surface, the orange curve a magnetic field line with safety factor $q = 1.7$, and ψ and ϕ are the poloidal and toroidal angles.

It is convenient to relate the field-aligned coordinates to the generalized toroidal coordinate system denoted by (ρ, θ, ζ). In this coordinate system, ρ has been already introduced as x — the flux surface label. Coordinates θ and ζ are related to the poloidal and toroidal angles ψ and ϕ, see Figure 7. For example, the physical quantities are also 2π-periodic in θ and ζ directions. The preserved periodicity helps set the boundary conditions in the field-aligned position space coordinates, for details see [44]. Furthermore, the generalized poloidal and toroidal angles are defined in such a way that the magnetic field lines appear as straight lines in the system of these coordinates.

As it was shown above, the magnetic field line can be expressed as the intersection of two planes, the flux surface and another plane defined by fixing the binormal coordinate $y = \text{const}$. If we fix the flux surface label, then the equation of the magnetic field line on the flux surface ρ is given by $y(\rho, \theta, \zeta) = \text{const}$, which describes the magnetic field line in the generalized poloidal and toroidal coordinates system. According to the definition of the generalized toroidal coordinate system, the corresponding magnetic field line is a straight line. Thus, we conclude that

$$y = C(\rho)\,(\,q(\rho)\theta - \zeta\,)\,, \tag{3.7}$$

where C is called the constant length factor and q is the safety factor.

The generalized poloidal and toroidal angles for which the magnetic field lines appear straight are, however, not unique. For the tokamak magnetic geometry, it is cus-

tomary to keep ζ as the original toroidal angle ϕ and transform the poloidal angle $\theta = \theta(\psi, \phi)$ (for the stellarator magnetic geometry, the original poloidal angle ψ is usually preserved).

The distance along the magnetic field line is measured by θ. Thus, the relation of the field-aligned coordinates to the generalized toroidal coordinates is defined by

$$\begin{aligned} x &= \rho\,, \\ y &= C(\rho)\,(\,q(\rho)\theta - \zeta)\,, \\ z &= \theta\,. \end{aligned} \tag{3.8}$$

Next, we consider the introduced circular geometry model, see Figure 7, and determine the field-aligned coordinates for this system in terms of the cylindrical-toroidal coordinates (r, ψ, ϕ). which are, in turn, related to the Cartesian coordinates through the following transform

$$\begin{aligned} x_c &= (R_0 + r\cos\psi)\sin\phi\,, \\ y_c &= (R_0 + r\cos\psi)\cos\phi\,, \\ z_c &= r\sin\psi\,, \end{aligned} \tag{3.9}$$

where R_0 denotes the major radius or the radius of magnetic axis. According to [43], the magnetic field in the circular geometry model is given by

$$\mathbf{B} = \frac{B_{\text{ref}}}{R}\left(\frac{1}{q(r)}\frac{r}{\sqrt{1-\varepsilon^2}}\hat{\mathbf{e}}_\psi + R_0\hat{\mathbf{e}}_\phi\right)\,, \tag{3.10}$$

where $R = R_0 + r\cos\psi$ and $\varepsilon = r/R_0$ is an inverse aspect ratio. The unit length covariant base vectors $\hat{\mathbf{e}}_\psi$ and $\hat{\mathbf{e}}_\phi$ are expressed in terms of the radius vector $\mathbf{R}_c = x_c\mathbf{i} + y_c\mathbf{j} + z_c\mathbf{k}$

$$\begin{aligned} \hat{\mathbf{e}}_\psi &= \frac{\mathbf{e}_\psi}{|\mathbf{e}_\psi|} = \frac{\partial\mathbf{R}}{\partial\psi}\Big/\left|\frac{\partial\mathbf{R}}{\partial\psi}\right| = \frac{1}{r}\frac{\partial\mathbf{R}}{\partial\psi} = \frac{\mathbf{e}_\psi}{r}\,, \\ \hat{\mathbf{e}}_\phi &= \frac{\mathbf{e}_\phi}{|\mathbf{e}_\phi|} = \frac{\partial\mathbf{R}}{\partial\phi}\Big/\left|\frac{\partial\mathbf{R}}{\partial\phi}\right| = \frac{1}{R}\frac{\partial\mathbf{R}}{\partial\phi} = \frac{\mathbf{e}_\phi}{R}\,. \end{aligned} \tag{3.11}$$

The tangent to the magnetic field line is parallel to the vector $\mathbf{B}$ and , therefore,

$$\mathbf{B} = c\,d\mathbf{R}_c\,, \tag{3.12}$$

where c is a proportionality constant. From (3.12) and the contravariant components of vectors $\mathbf{B}\,(B^r, B^\psi, B^\phi) = B^r\mathbf{e}_r + B^\psi\mathbf{e}_\psi + B^\phi\mathbf{e}_\phi$ and $d\mathbf{R}_c\,(dr, d\psi, d\phi)$, it follows that

$$\frac{B^r}{dr} = \frac{B^\psi}{d\psi} = \frac{B^\phi}{d\phi} = c\,. \tag{3.13}$$

For the magnetic field defined by (3.10), we find the contravariant components

$$\begin{aligned} B^r &= 0\,, \\ B^\psi &= \frac{B_{\text{ref}}}{R}\frac{r}{q(r)}\frac{1}{\sqrt{1-\varepsilon^2}}\frac{1}{|\mathbf{e}_\psi|} = \frac{B_{\text{ref}}}{R}\frac{1}{q(r)}\frac{1}{\sqrt{1-\varepsilon^2}}\,, \\ B^\phi &= \frac{B_{\text{ref}}}{R}R_0\frac{1}{|\mathbf{e}_\phi|} = B_{\text{ref}}\frac{R_0}{R^2}\,. \end{aligned} \tag{3.14}$$

Therefore, the magnetic field line is given by the solution of the ordinary differential equation

$$\frac{\partial\psi}{\partial\phi} = \frac{B^{\psi}}{B^{\phi}} = \frac{1}{q(r)}\frac{1}{\sqrt{1-\varepsilon^2}}\frac{R}{R_0} = \frac{1+\varepsilon\cos\psi}{q(r)\sqrt{1-\varepsilon^2}}\,. \tag{3.15}$$

The solution takes the form

$$\phi = q(r)\sqrt{1-\varepsilon^2}\int\frac{d\psi}{1+\varepsilon\psi} = q(r)\left(2\arctan\left[\sqrt{\frac{1-\varepsilon}{1+\varepsilon}}\tan\left(\frac{\psi}{2}\right)\right]\right) - \text{const}\,. \tag{3.16}$$

The magnetic field line for the circular geometry in the cylindrical-toroidal coordinates is then given by

$$q(r)\left(2\arctan\left[\sqrt{\frac{1-\varepsilon}{1+\varepsilon}}\tan\left(\frac{\psi}{2}\right)\right]\right) - \phi = \text{const}\,. \tag{3.17}$$

An example of this curve for const $= 0$ is shown in Figure 8, which demonstrates that the field line is not straight in the (ψ,ϕ) coordinate system. However, the magnetic

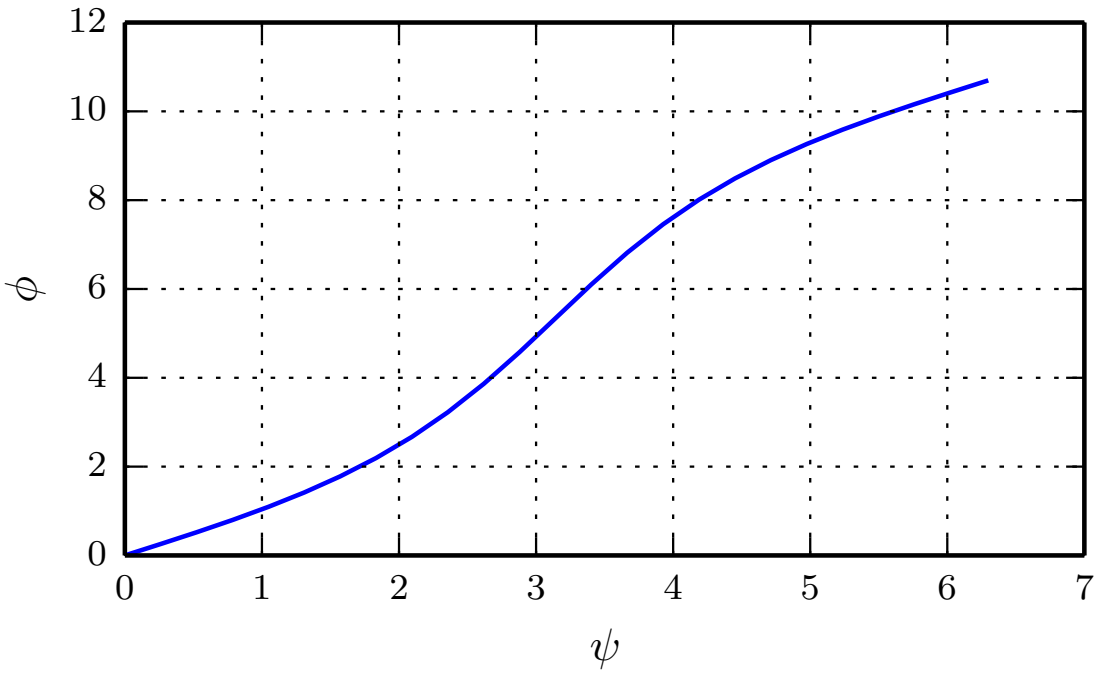


Figure 8: Magnetic field line of the circular model in the coordinate system of the poloidal (ψ) and toroidal (ϕ) angles.

field line becomes straight with the following generalized poloidal and toroidal angles

$$\begin{aligned}\theta &= 2\arctan\left[\sqrt{\frac{1-\varepsilon}{1+\varepsilon}}\tan\left(\frac{\psi}{2}\right)\right]\,,\\ \zeta &= \phi\,.\end{aligned} \tag{3.18}$$

These results allow us to construct the field-aligned coordinates by applying the transform (3.8).

A field-aligned curvilinear coordinate system can be constructed based on an analytic model (as it was done for the circular model) or on an arbitrary MHD equilibrium. For the latter case, GENE implements an interface with codes CHEASE [11, 112] and EFIT [110, 113, 114].

As aforementioned, the periodicity of physical quantities in the straight field line angles helps determine the boundary conditions in the binormal and parallel directions. For example, if we have a physical quantity given in the generalized toroidal coordinates $f(\rho, \theta, \zeta)$, then the following expressions are valid

$$f(\rho, \theta, \zeta) = f(\rho, \theta, \zeta + 2\pi), \tag{3.19}$$

$$f(\rho, \theta, \zeta) = f(\rho, \theta + 2\pi, \zeta). \tag{3.20}$$

The first periodicity condition (3.19), together with the definition of the binormal coordinate (3.7), yields the periodicity condition in y

$$f(x, y, z) = f(x, y + 2\pi C, z). \tag{3.21}$$

In practice, however, the binormal turbulence correlation length is smaller than the whole toroidal turn $\zeta + 2\pi$, and the periodicity is assigned to an integer number fraction of the toroidal turn $\zeta + 2\pi/n_0$. The modified periodicity condition is then given by

$$f(x, y, z) = f\left(x, y + \frac{2\pi C}{n_0}, z\right). \tag{3.22}$$

This periodic boundary condition allows switching to the Fourier representation of the binormal coordinate and using fast spectral methods in GENE.

The second periodicity condition (3.20) results in the quasi-periodic condition in the parallel direction. The exact form of this boundary condition is, for the already Fourier transformed y coordinate, given by

$$f(x, k_y, z + L_z) = f(x, k_y, z) \exp\left[-2\pi i n_0 q(x) j\right], \tag{3.23}$$

where $k_y = j k_{y,\min}$ is the toroidal wave number. For the parallel direction coordinate, we employ a finite difference method on an equidistant grid.

In the radially global GENE version, a finite difference method is also used for the radial coordinate x, due to the Dirichlet or Neumann boundary conditions. In the case of the Dirichlet boundary condition, the temperature and density radial profiles are fixed on the boundaries. Therefore, this hinders the natural process of flattening the initial temperature and density profiles and, thus, limits the magnitude of the turbulent fluctuations. As a result, small profile variations near the x boundaries might yield large gradients and cause numerical instabilities. However, even if the full profile flattening (like in the radially local simulations) were possible, because of significant radial variations of the temperature and density, it would lead to a severe violation of the gyrokinetic ordering. The problem can be alleviated by introducing source- or sink-like terms, which are added to the right hand side of the Vlasov equation. This technique is meant to artificially dampen the fluctuations in the vicinity of the radial boundaries. In GENE, the introduced terms are of Krook type and they are active (not

zero) only close to the boundaries in the regions called Krook-type buffer zones; the realization of Krook terms in GENE is explained in [43, 44, 106].

The parallel and, in many cases, the binormal directions require less grid points than the radial distance mixed with the velocity space subdomain. On the one hand, the fast spectral method for the y direction and the elongated structures of turbulence in the z direction help reduce the number of points for these coordinates. On the other hand, the radial temperature variation requires a lot of grid points in the velocity space. This problem is addressed in the following section on the velocity space coordinates and is also going to be a main motivation for the newly developed grids presented in Chapter 4.

3.3 VELOCITY SPACE COORDINATES

To decide upon suitable discretization schemes in the velocity space, the following operations involving the velocity space coordinates have to be considered. The velocity space coordinates $v_{\parallel}$ and μ appear only in the Vlasov equation (2.26) and are integrated out in the moment computations for the Maxwell equations (2.32,2.33), which are solved in the position space. In collisionless plasma, the derivatives in the velocity space are computed only along the parallel velocity direction and appear in the nonlinear and trapping terms. Both velocity coordinates are involved in the computations of the moments, which couple the Vlasov to the Maxwell solver. In return, the fields computed by the Maxwell solver are gyro-averaged and become part of the Vlasov equation. Due to the last operation, the resulting gyro-averaged fields pick up a magnetic moment dependence.

The section is structured as follows. First, we introduce projections of the background distribution functions, which are important to understand the choice of an appropriate range and resolution in the velocity space grid. Next, we discuss how a regular velocity grid is constructed in a rectangular domain. Furthermore, we introduce the challenge of an excess of velocity space grid points caused by the spatial temperature variation. We finish the section with a discussion on the solution to this problem (by normalizing the velocity coordinates), as well as of the related computational costs.

3.3.1 *Background Distribution Function*

As it is explained in Subsection 2.1.3, the gyrokinetic Vlasov equation in so-called δf-codes such as GENE is simplified by splitting the distribution function into an equilibrium and a fluctuating part $F = F_0 + f_1$. From the gyro-kinetic ordering requirements, it follows that, if the background distribution function in a certain phase space area is negligible, then we cannot expect strong fluctuations in that area. Furthermore, the shape of the perturbed distribution function retains the shape of the background distribution approximately, see Chapter 6. Therefore, we choose the background distribution function as the criterion for constructing computational grids.

The background distribution function used in GENE is a local (i.e., radial-dependent) Maxwellian in the velocity space. It depends on the radial density $n(x)$ and temperature $T(x)$ profiles, and is given by

$$F_0(x, v_\parallel, v_{\perp x}, v_{\perp y}) = \frac{n(x)}{\pi^{3/2} v_T^3(x)} \exp\left[-\frac{m\left(v_\parallel^2 + v_{\perp x}^2 + v_{\perp y}^2\right)}{2T(x)}\right], \tag{3.24}$$

where x is the radial distance, $v_\parallel$ the parallel velocity component, $v_\perp$ the perpendicular velocity components, and $v_T(x) = \sqrt{2T(x)/m}$ the thermal speed. In the five dimensional gyrokinetic codes, we have only two velocity directions. In GENE, we use a magnetic moment coordinate μ instead of the perpendicular velocity components $v_{\perp x}$ and $v_{\perp y}$. The magnetic moment can be expressed in terms of the perpendicular velocity:

$$\mu = \frac{m\left(v_{\perp x}^2 + v_{\perp y}^2\right)}{2B}. \tag{3.25}$$

To derive the background distribution function $F_0(x, v_\parallel, \mu)$ depending on only two velocity variables instead of three, we substitute the perpendicular velocity components according to

$$\begin{aligned} \mu &= \frac{m\left(v_{\perp x}^2 + v_{\perp y}^2\right)}{2B}, & v_{\perp x} &= \pm\sqrt{\frac{2B}{m}\mu_x}, \\ \mu_x &= \frac{m v_{\perp x}^2}{2B}, & v_{\perp y} &= \pm\sqrt{\frac{2B}{m}(\mu - \mu_x)}. \end{aligned} \tag{3.26}$$

Because one point (μ, μ_x) is mapped to four points in the orthogonal velocity space $(\pm v_{\perp x}, \pm v_{\perp y})$, the distribution function for μ and μ_x variables is given by

$$F_0(x, v_\parallel, \mu, \mu_x) = \frac{n(x)}{\pi^{3/2} v_T^3(x)} \exp\left[-\frac{m v_\parallel^2}{2T(x)} - \frac{\mu B}{T(x)}\right] \sum_{i=1}^{4} |J_i(\mu, \mu_x)|, \tag{3.27}$$

where $J_i(\mu, \mu_x)$ is the determinant of the Jacobian matrix for the i-th quadrant of the orthogonal velocity plane. The absolute value of the Jacobian matrix determinant is equal for all four points $(\pm v_{\perp x}, \pm v_{\perp y})$. Therefore, we consider only the first quadrant, where the Jacobian matrix is given by

$$\begin{bmatrix} \frac{\partial v_{\perp x}}{\partial \mu_x} & \frac{\partial v_{\perp x}}{\partial \mu} \\ \frac{\partial v_{\perp y}}{\partial \mu_x} & \frac{\partial v_{\perp y}}{\partial \mu} \end{bmatrix} = \begin{bmatrix} \pm\sqrt{\frac{B}{2m\mu_x}} & 0 \\ \mp\sqrt{\frac{B}{2m(\mu-\mu_x)}} & \pm\sqrt{\frac{B}{2m(\mu-\mu_x)}} \end{bmatrix}. \tag{3.28}$$

The absolute value of the determinant is

$$|J(\mu, \mu_x)| = \frac{B}{2m} \frac{1}{\sqrt{\mu_x(\mu - \mu_x)}}. \tag{3.29}$$

Finally, the distribution function for μ_x, μ space is given by

$$F_0(x, v_\parallel, \mu, \mu_x) = \frac{2Bn(x)}{m\pi^{3/2} v_T^3(x)} \exp\left[-\frac{m v_\parallel^2}{2T(x)} - \frac{\mu B}{T(x)}\right] \frac{1}{\sqrt{\mu_x(\mu - \mu_x)}}. \tag{3.30}$$

To have the distribution function dependent on the magnetic moment only, we integrate $F_0(x, v_\parallel, \mu, \mu_x)$ over all acceptable values of μ_x corresponding to the range $[0, \mu]$. Taking into consideration the elementary integral

$$\int_0^\mu \frac{1}{\sqrt{\mu_x\,(\mu - \mu_x)}}\,d\mu_x = \pi\,. \tag{3.31}$$

we obtain the desired distribution function

$$F_0(x, v_\parallel, \mu) = \frac{2Bn(x)}{m\sqrt{\pi}v_T^3(x)} \exp\left[-\frac{mv_\parallel^2}{2T(x)} - \frac{\mu B}{T(x)}\right] = n(x) \times \sqrt{\frac{m}{2\pi T(x)}} \exp\left[-\frac{mv_\parallel^2}{2T(x)}\right] \times \frac{B}{T(x)} \exp\left[-\frac{\mu B}{T(x)}\right]. \tag{3.32}$$

The background distribution function receives the radial distance x-dependence from the profiles of the density $n(x)$ and temperature $T(x)$. In practice, these profiles can be analytically generated or directly obtained from real experiments. An example of $n(x)$ and $T(x)$ for a particular TCV[1] discharge, for details see [44], is shown in Figure 9. Strong temperature and density variations lead to different shapes of the background distribution functions.

In Figures 10 and 11, we demonstrate the projections of (3.32) on the $x - v_\parallel$ and $x - \mu$ planes, which are plotted using the introduced TCV profiles. The projections of the distribution function are given by

$$F_0\left(x, v_\parallel\right) = n(x)\sqrt{\frac{m}{2\pi T(x)}} \exp\left[-\frac{mv_\parallel^2}{2T(x)}\right], \tag{3.33}$$

$$F_0(x, \mu) = n(x)\frac{B}{T(x)} \exp\left[-\frac{\mu B}{T(x)}\right]. \tag{3.34}$$

and obtained by marginalizing out (discarding) μ for (3.33) or $v_\parallel$ for (3.34), which is done by integrating the full Maxwellian (3.32). Furthermore, to visualize the distribution functions in Figures 10 and 11, we use normalized quantities, in the same way it is done in GENE simulations. We assume that the density and temperature are already provided for the normalized radial coordinate $\hat{x}$, and the distribution functions (3.33) and (3.34). Therefore, we can simply substitute x by $\hat{x}$ in the formulas of the background distribution functions. Moreover, according to Section 3.1, we apply the following substitutions

$$\begin{aligned} n(\hat{x}) &= \hat{n}_0\hat{n}(\hat{x})n_{ref}\,, & T(\hat{x}) &= \hat{T}_0\hat{T}(\hat{x})T_{ref}\,, & B &= \hat{B}B_{ref}\,, \\ v_\parallel &= \hat{v}_\parallel\sqrt{\frac{2\hat{T}_0}{\hat{m}}}\,\sqrt{\frac{T_{ref}}{m_{ref}}}\,, & \mu &= \hat{\mu}\hat{T}_0\frac{T_{ref}}{B_{ref}}\,, \end{aligned} \tag{3.35}$$

1 Tokamak à Configuration Variable

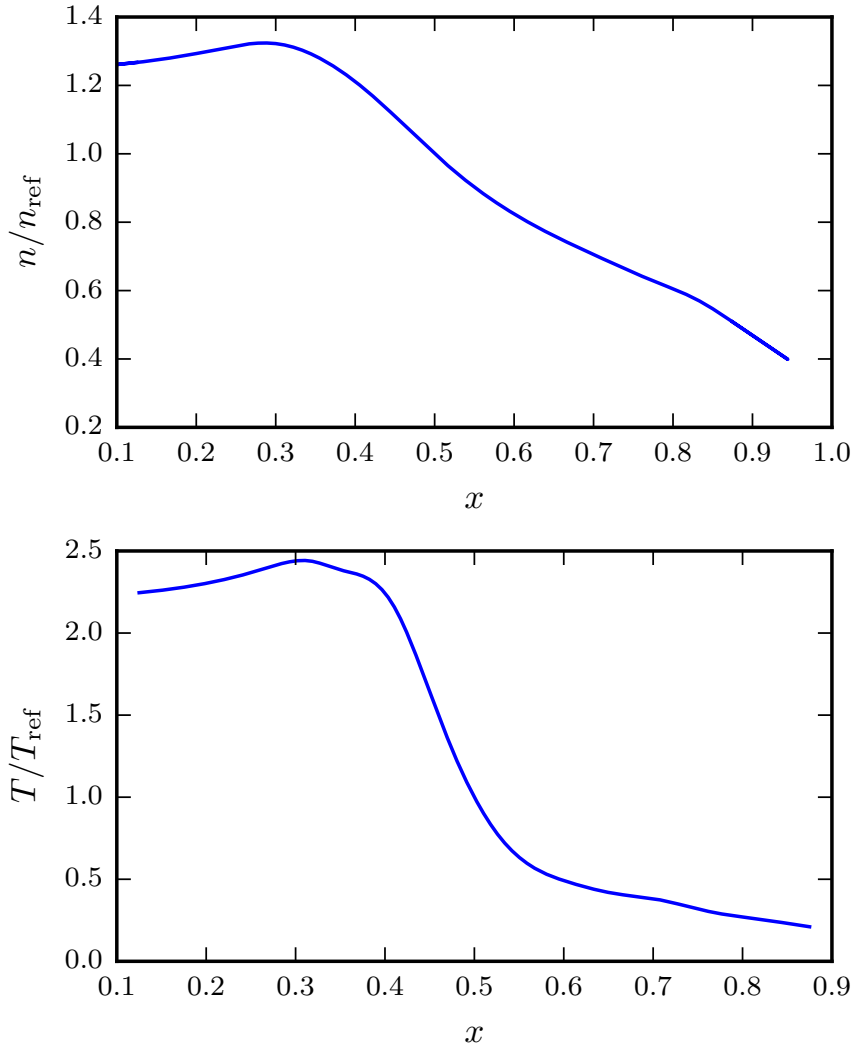


Figure 9: Examples of electron density (top) and temperature (bottom) profiles of a TCV discharge. The radial distance is shown in the minor radius units, the density and temperature values are shown in units relative to the reference values $n_{\mathrm{ref}}, T_{\mathrm{ref}}$ taken at the reference radial distance ($x_{\mathrm{ref}} = 0.5$).

in expressions (3.33) and (3.34), where, for the sake of simplicity, we omit the species index s. To summarize, we obtain the following normalized equilibrium distribution functions

$$\hat{F}_0\left(\hat{x}, \hat{v}_\parallel\right) = F\left(\hat{x}, v_\parallel\right) \frac{dv_\parallel}{d\hat{v}_\parallel} = \hat{n}(\hat{x})\hat{n}_0 \sqrt{\frac{1}{\hat{T}(\hat{x})}} \exp\left[-\frac{\hat{v}_\parallel^2}{\hat{T}(\hat{x})}\right], \tag{3.36}$$

$$\hat{F}_0\left(\hat{x}, \hat{\mu}\right) = F\left(\hat{x}, \mu\right) \frac{d\mu}{d\hat{\mu}} = \hat{n}(\hat{x})\hat{n}_0 \frac{\hat{B}}{\hat{T}(\hat{x})} \exp\left[-\frac{\hat{\mu}\hat{B}}{\hat{T}(\hat{x})}\right]. \tag{3.37}$$

Here, we omit n_{ref} to gain dimensionless distribution functions. Furthermore, for the visualization of the equilibrium distribution function in the $x - \mu$ subspace, $\hat{B} \approx 1$, so it can be neglected. Nevertheless, the temperature and density radial dependencies usually given as input are not scaled and correspond to $\hat{T}_p(\hat{x}) = \hat{T}(\hat{x})\hat{T}_0$ and $\hat{n}_p(\hat{x}) =$

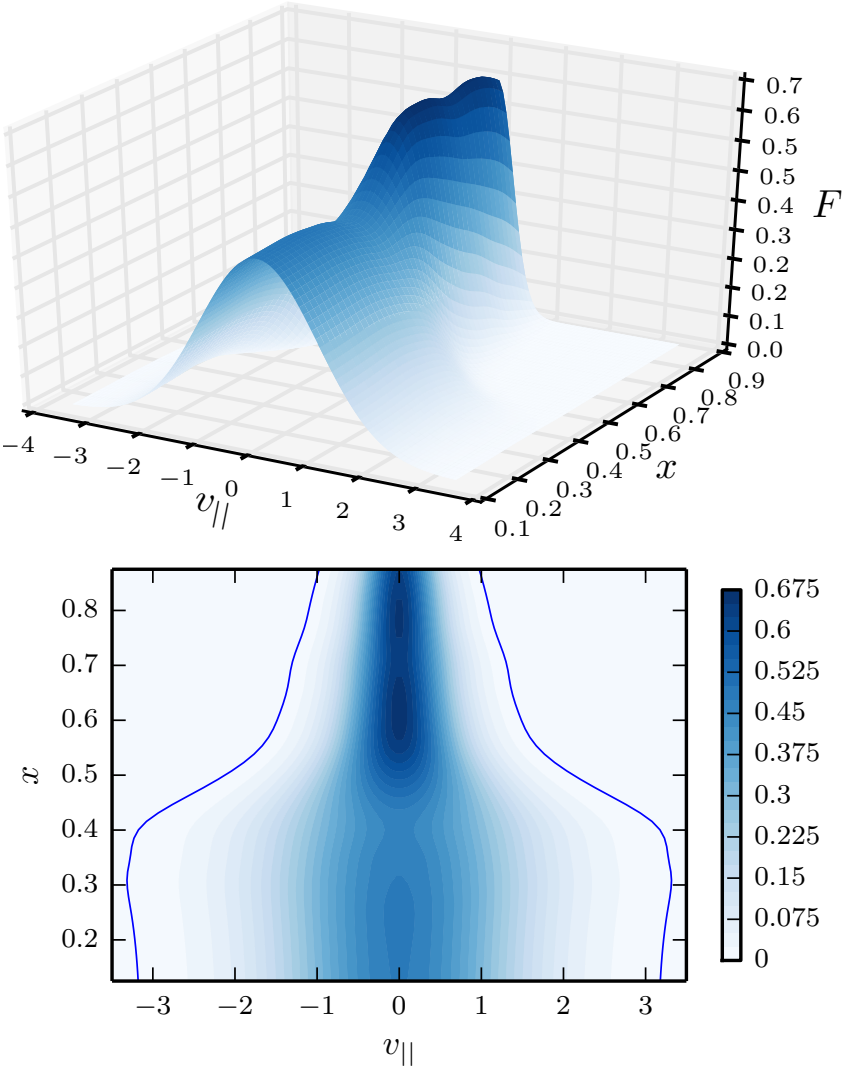


Figure 10: Surface (top) and contour (bottom) plots of background distribution function $F\left(x, v_{\|}\right)$. Blue lines on the contour plot corresponds to a confidence range of three standard deviations.

$\hat{n}(\hat{x})\hat{n}_0$, where the subscript p refers to the input profile values. Therefore, the original expressions that we use to compute the background distribution functions are given by

$$\hat{F}_0\left(\hat{x}, \hat{v}_{\|}\right) = \hat{n}_p(\hat{x}) \sqrt{\frac{\hat{T}_p\left(\hat{x}_0\right)}{\hat{T}_p(\hat{x})}} \exp\left[-\hat{v}_{\|}^2 \frac{\hat{T}_p\left(\hat{x}_0\right)}{\hat{T}_p(\hat{x})}\right], \tag{3.38}$$

$$\hat{F}_0\left(\hat{x}, \hat{\mu}\right) = \hat{n}_p(\hat{x}) \frac{\hat{T}_p\left(\hat{x}_0\right)}{\hat{T}_p(\hat{x})} \exp\left[-\hat{\mu} \frac{\hat{T}_p\left(\hat{x}_0\right)}{\hat{T}_p(\hat{x})}\right]. \tag{3.39}$$

As shown in Figures 10 and 11 for the previously introduced TCV profiles, the projections $\hat{F}_0\left(\hat{x}, \hat{v}_{\|}\right)$ and $\hat{F}_0\left(\hat{x}, \hat{\mu}\right)$ have a smooth shape with long wings in the high temperature regions (low $\hat{x}$), but take a peak-like shape in the low temperature regions (high $\hat{x}$). Because these disparities in the background shape are met frequently, they

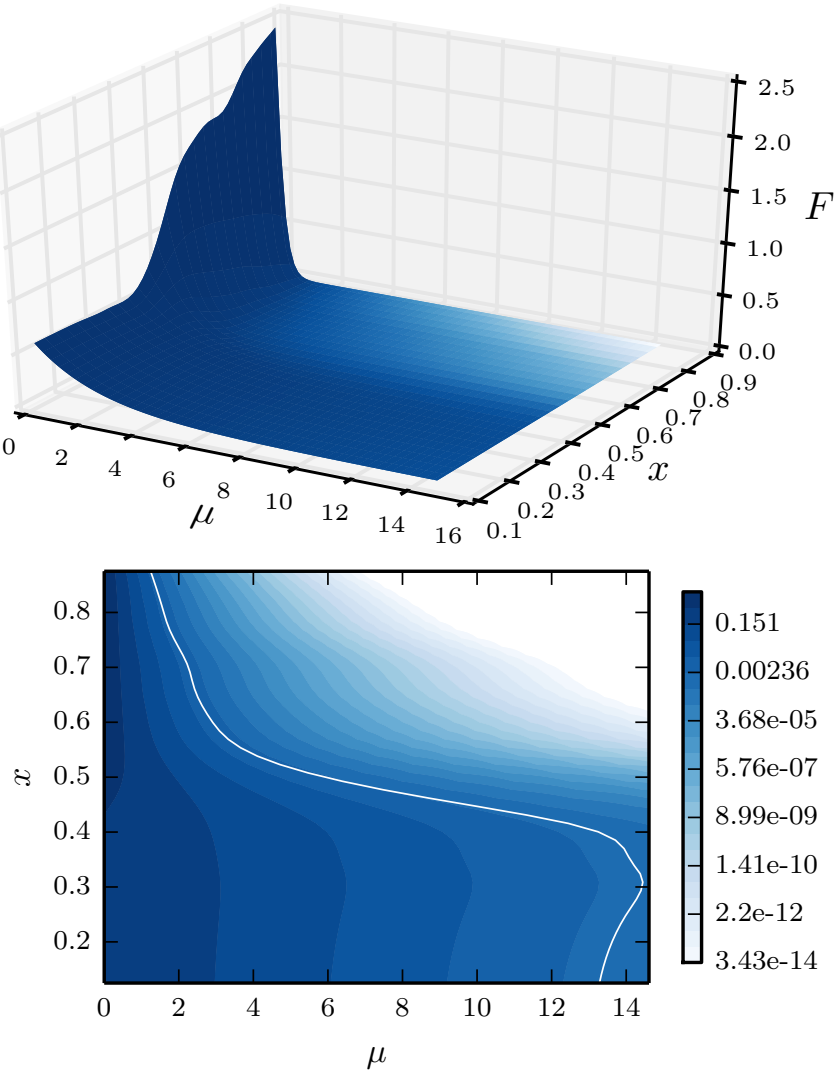


Figure 11: Surface (top) and contour (bottom) plots of background distribution function $F(x, \mu)$. White line on the contour plot correspond to a confidence range of three standard deviations.

have to be reflected in the choice of range and resolution of the computational grids in the $x - v_{\parallel} - \mu$ subspace.

3.3.2 *Default Approach: Regular Velocity Space Grid*

The default approach to construct the velocity space grid in gyrokinetic codes such as GENE is to choose the cuboid shape domain in the $x - v_{\parallel} - \mu$ subspace, and discretize $v_{\parallel}$ and μ according to the numerical operations applied in these directions. To perform discretizations in GENE, we compute derivatives with a finite difference scheme in the parallel velocity direction and perform quadrature-like operations in the magnetic moment direction. Therefore, to discretize the parallel velocity coordinate we use an equidistant mesh and nodes of a Gauss-type quadrature for the magnetic moment coordinate. The default choice for the μ mesh is the Gauss-Laguerre quadrature rule,

because it is optimal for integrating functions of the class exponential-multiplied-by-polynomial, which represents the expected perturbed distribution function due to the background (3.34). An example of $v_{\parallel} - \mu$ grid with (for the sake of visibility) a very coarse resolution is shown in Figure 12.

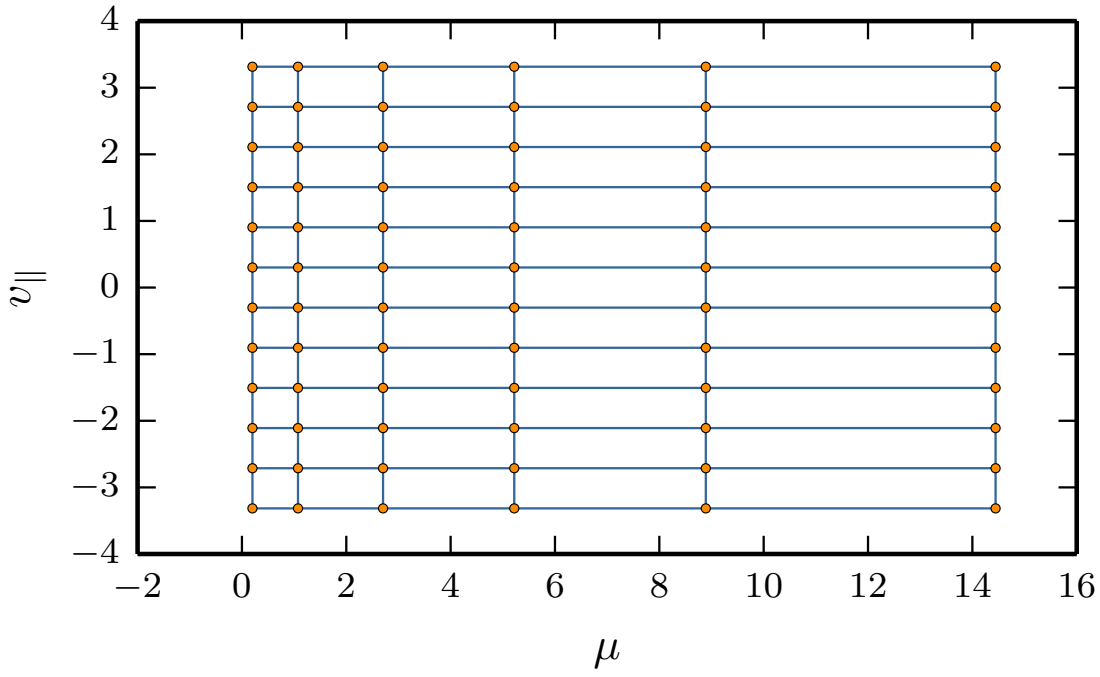


Figure 12: An example of a grid in $v_{\parallel} - \mu$ subspace with a very coarse resolution.

For the default domain of the cuboid shape, it is very important to choose a range and a resolution that are wide and fine enough in the velocity space, so that the results of the simulations are sufficiently accurate to rely on. Before introducing a standard procedure of choosing the range and resolution parameters, we first need to present the case of an optimal shape of the $x - v_{\parallel} - \mu$ subdomain with a corresponding resolution.

To choose the appropriate range of the parallel velocity $(-\mathtt{lv}, \mathtt{lv})$ and the magnetic moment $(0, \mathtt{lw})$, we rely on the confidence interval approach. Assuming a normal distribution, the standard initial choice, if more precise ranges are not available from previous simulations, is three standard deviations, which corresponds to a 99.7% confidence level. Occasionally, however, to capture all velocity space structures, it is necessary to take even wider ranges than that. In such cases, we have to proceed with care, because too wide ranges may lead to spurious results, due to the insufficient precision of floating point operations in the areas where the background distribution function is negligibly small and the fluctuating part is, according to the gyrokinetic ordering, even smaller. We find the contour of the desired simulation domain in the $x - v_{\parallel} - \mu$ subspace by computing the confidence intervals at each radial distance. The resulting contours are shown in Figures 10 and 11 (bottom). The simulation domain surface in the $x - v_{\parallel} - \mu$ space is shown in Figure 13.

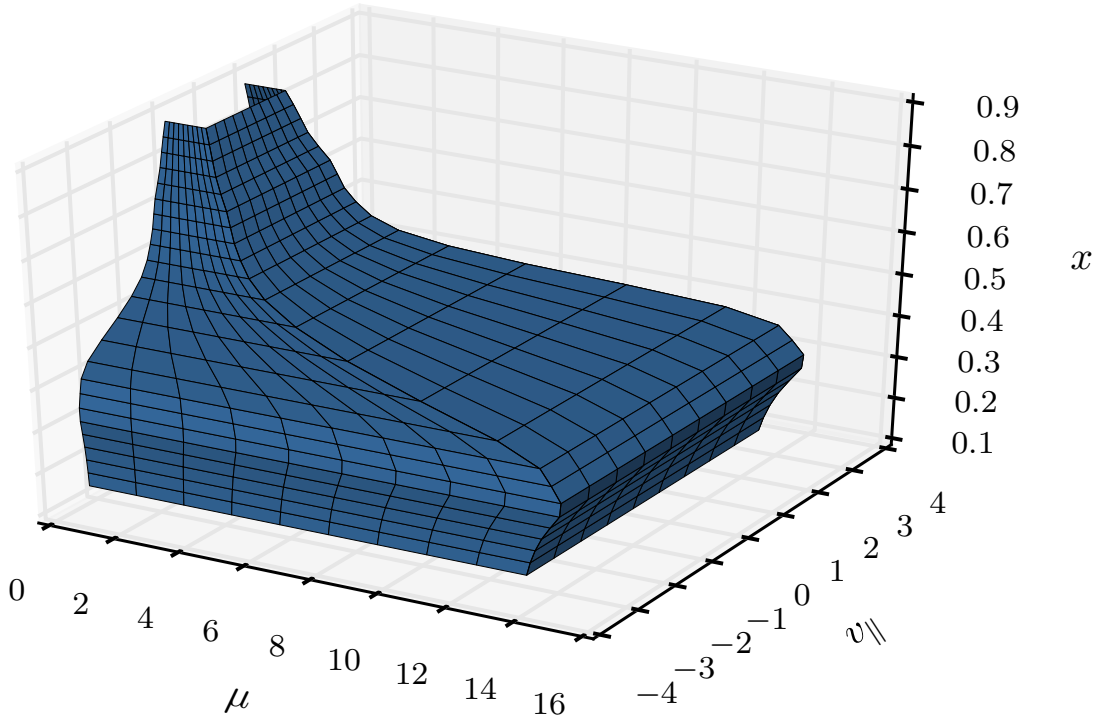


Figure 13: An example of a surface of the $x - v_{\parallel} - \mu$ subdomain corresponding to a confidence interval of three standard deviations for a normal distribution.

To compute ranges for the corresponding equilibrium distribution functions (3.38) and (3.39) we apply the following formulas. The probability p of the confidence interval $(-lv, lv)$ is given by

$$p = \Phi(lv) - \Phi(-lv) = \operatorname{erf}\left(\frac{lv}{\sigma\sqrt{2}}\right), \tag{3.40}$$

where Φ denotes the cumulative distribution function (CFD), which for a normal distribution is given by

$$\Phi(lv) = \frac{1}{2}\left(1 + \operatorname{erf}\left[\frac{lv}{\sqrt{2}\sigma}\right]\right), \tag{3.41}$$

where σ is the standard deviation

$$\sigma = \sqrt{\frac{1}{2}\frac{\hat{T}_p(\hat{x})}{\hat{T}_p(\hat{x}_0)}} \tag{3.42}$$

and erf denotes the error function

$$\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}}\int_0^x e^{-t^2}\,dt\,. \tag{3.43}$$

If we have to determine the range of the parallel velocity coordinate from a given confidence level p, we apply the following formula

$$lv = \sqrt{2}\sigma\,\operatorname{erf}^{-1}(p)\,. \tag{3.44}$$

Otherwise, if the range is specified by a number of standard deviations n, then we simply compute

$$lv = n\sigma . \tag{3.45}$$

The confidence probability of the interval $(0, lw)$ for the magnetic moment equilibrium distribution (3.39) is computed from

$$p = \int_0^{lw} \lambda \exp\left[-\lambda\mu\right] \, d\mu = 1 - \exp\left[-\lambda \cdot lw\right] , \tag{3.46}$$

where λ is given by

$$\lambda = \frac{\hat{T}_p(\hat{x}_0)}{\hat{T}_p(\hat{x})} . \tag{3.47}$$

If we compute lw from the given probability p, we apply the inverse of (3.46)

$$lw = -\frac{\ln(1-p)}{\lambda} . \tag{3.48}$$

In case the number of standard deviations n for the normal distribution is given, we first compute the probability according to (3.40)

$$p = \operatorname{erf}\left(\frac{n}{\sqrt{2}}\right) , \tag{3.49}$$

and then substitute the result in (3.48), which yields

$$lw = -\frac{1}{\lambda} \ln\left(1 - \operatorname{erf}\left(\frac{n}{\sqrt{2}}\right)\right) . \tag{3.50}$$

In practice, we do not use the last expression, but instead replace the error function by using the following scaled complementary error function erfcx; then erf is replaced according to

$$1 - \operatorname{erf}\left(\frac{n}{\sqrt{2}}\right) = \exp\left(-\frac{n^2}{2}\right) \operatorname{erfcx}\left(\frac{n}{\sqrt{2}}\right) , \tag{3.51}$$

which leads to

$$lw = \frac{1}{\lambda}\left(\frac{n^2}{2} - \ln\left(\operatorname{erfcx}\left(\frac{n}{\sqrt{2}}\right)\right)\right) . \tag{3.52}$$

The last expression is more convenient, because now we distinguish between the main term, which is proportional to $n^2/2$, and the correction.

A proper resolution is obtained by fixing the number of grid points at each radial distance and assigning the ranges as described previously. The number of necessary grid points may vary from one scenario to another; therefore, the choice usually depends on experience.

To demonstrate how the regular grid is constructed, we consider an example in the $x - v_{\parallel}$ subdomain. We assume that we know how many parallel velocity grid points are sufficient for the local simulations (narrow radial range) with the same temperature

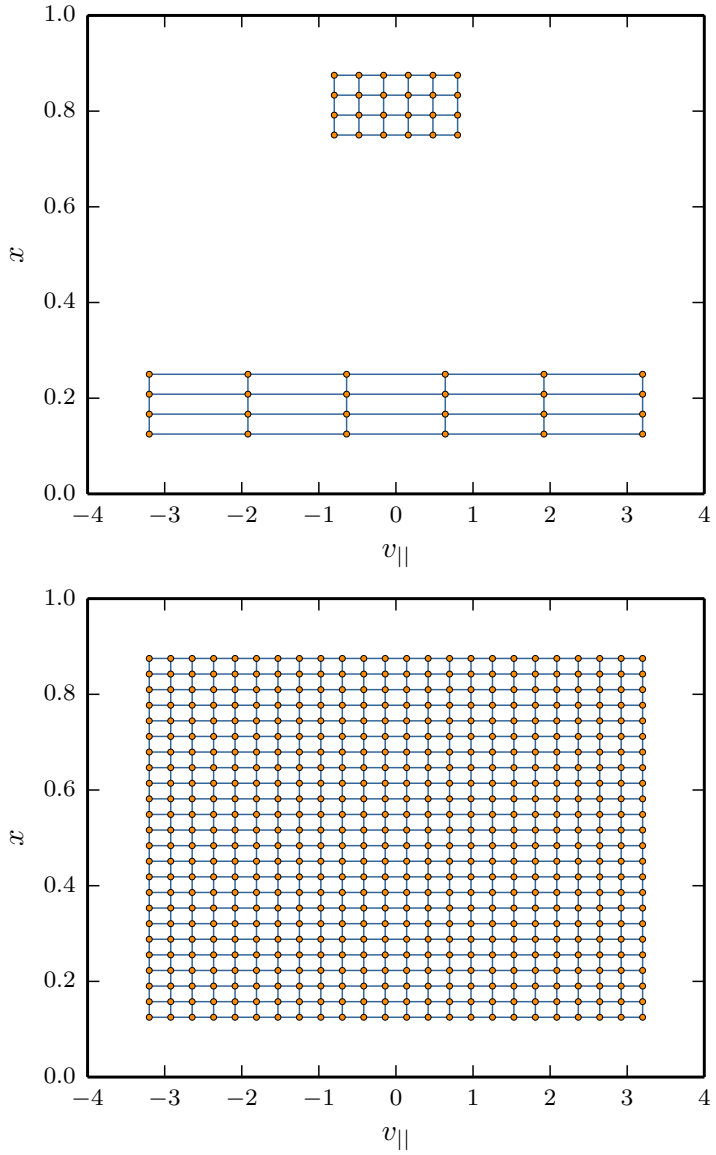


Figure 14: Construction of the regular grid (bottom) based on radially local grids (top).

profiles as for the target global simulations (almost full machine size radial range). Knowing the correct parallel velocity ranges at the highest and lowest temperature regions, we schematically demonstrate these two grids in Figure 14 (top). To span the rectangular grid over the whole radial coordinate range, we choose the widest range and the finest resolution of the local grids. As a result, we obtain the grid demonstrated in Figure 14 (bottom), which satisfies all prerequisites for the range and resolution at all x. Nonetheless, despite its simplicity, such a regular grid has too many computational nodes.

In the default approach, the $x - \mu$ grid is constructed by choosing the Gauss-Laguerre quadrature rule adjusted to the exponentially decaying equilibrium distribution function at the reference point x, usually at the middle of the interval of the radial coordinate. As a consequence, the precision of the quadrature rule decays drastically close to the lower and upper bounds of the radial range, as shown in Section 4.2. To compen-

sate this effect, we have to invest again many more μ nodes than one would expect for the radially local simulations.

There are two main approaches to tackle the issue with the excessive number of computational nodes in the velocity space. One stems from the modification of the underlying governing equations and is described in Subsection 3.3.3. Another approach, which we consider the better option and which allows reusing the already existent code for regular grids, is explained in Chapter 4.

3.3.3 *Normalization Approach: Transformed Velocity Space Grids*

In the following, we present a method to solve the problem with the excessive number of grid points required for the velocity space discretization by transforming the velocity coordinates. First, we derive the transformation equations. Then, we briefly explain complications of applying this technique to an existing gyrokinetic Eulerian code like GENE. Although we do not apply this method in practice, it serves us as a good indication for designing the block-structured grids, which are presented in Chapter 4.

Transformations of Velocity Coordinates

The simplest way of addressing the aforementioned problem is to use the parallel velocity and magnetic moment coordinates normalized by radially dependent thermal speed and corresponding thermal magnetic moment

$$v_T(x) = \sqrt{\frac{2T(x)}{m}} \qquad \text{and} \qquad \mu_T(x) = \frac{m v_T^2(x)}{2B} = \frac{T(x)}{B}\,. \tag{3.53}$$

After the following transform of the velocity coordinates

$$v_\parallel' = \frac{v_\parallel}{v_T} \qquad \text{and} \qquad \mu' = \frac{\mu}{\mu_T} \tag{3.54}$$

we remove the temperature profiles from the background distribution function. The equilibrium distribution functions in $x - v_\parallel'$ and $x - \mu'$ are given by

$$F_0'\left(x, v_\parallel\right) = F_0\left(x, v_\parallel\left(v_\parallel'\right)\right)\frac{dv_\parallel}{dv_\parallel'} = \frac{n(x)}{\sqrt{2}}\exp\left(v_\parallel'^2\right)\,, \tag{3.55}$$

$$F_0'\left(x, \mu\right) = F_0\left(x, \mu\left(\mu'\right)\right)\frac{d\mu}{d\mu'} = n(x)\exp\left(\mu'\right)\,. \tag{3.56}$$

With the new velocity coordinates, the scales of the distribution function structures are the same at all radial positions. Therefore, the number of grid points necessary for the velocity space is approximately the same as for the local simulations. A rough estimate shows that we have to use $\left(v_{T_{max}}/v_{T_{min}}\right)^3 = \left(T_{max}/T_{min}\right)^{3/2}$ less points with the new velocity coordinates in comparison to the regular grid introduced in 3.3.2. Here, by $v_{T_{max}}$ and $v_{T_{min}}$ we denote the maximum and minimum values of the thermal speed, which is a function of the radial coordinate. For example, for a realistic temperature ratio $T_{max}/T_{min} \sim 10$, the theoretical reduction of grid points is ≈ 30.

The normalization transform, which we introduce in (3.54), can be derived by using the transfinite mapping technique, described in [115] and [116]. This method provides more control over the mesh structure than the pure normalization. For this reason and

for the sake of completeness, next, we provide the derivation of the transfinite mapping for our particular problem.

The transfinite mapping technique allows us to transform the velocity space coordinates, so that the original subdomain shown in Figure 13 is mapped to the new (logical) computational subdomain having the shape of a rectangular cuboid. With this method, we can directly work in the three-dimensional domain. However, in our case, we demonstrate a simpler procedure by first transforming the x and $v_{||}$ coordinates and then repeating the procedure for μ.

The grids of the logical and physical computational domains for our case are shown in Figure 15. The velocity range in the physical domain corresponds to the scaled stan-

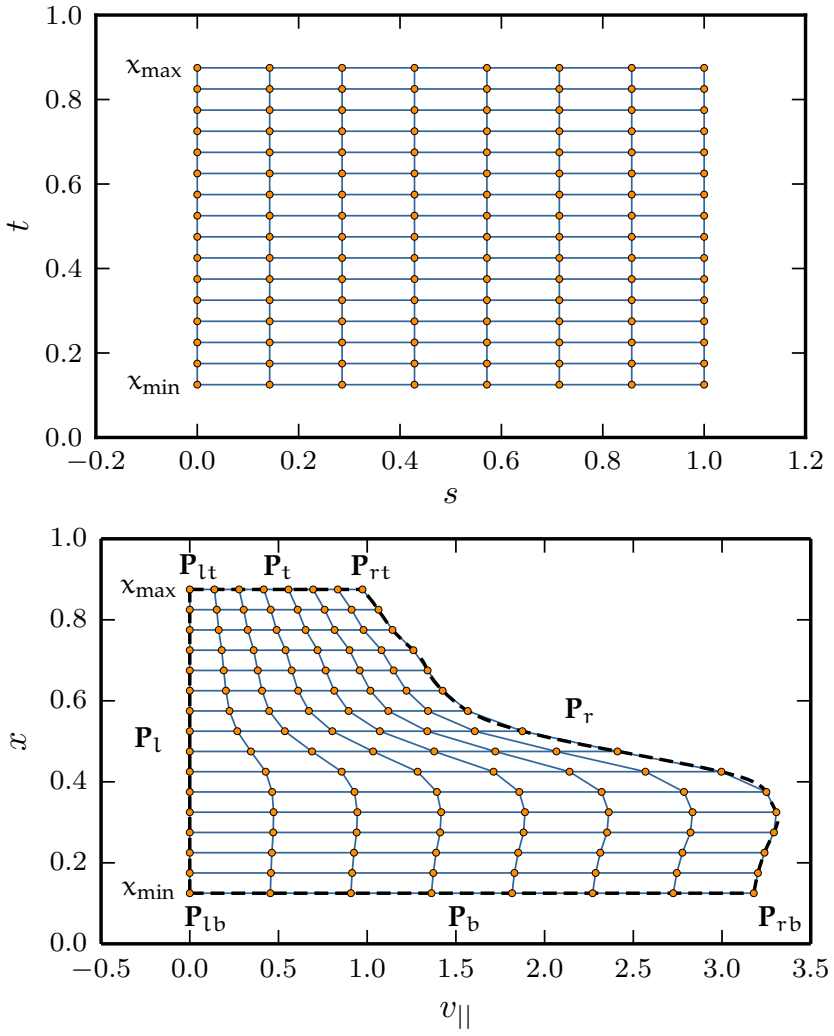


Figure 15: Grids in logical $\mathcal{L}(s,t)$ (top) and physical $\mathcal{P}(x,v_{||})$ (bottom) domains. The transfinite interpolation technique provides an efficient way to find mapping $\mathbf{T}:\mathcal{L}\mapsto\mathcal{P}$.

dard deviation $c_1\sigma(x)$, for details see 3.3.2. Our problem is to find a bijective function $\mathbf{T}:\mathcal{L}\mapsto\mathcal{P}$, which maps the logical space $\mathcal{L}$ with coordinates (s,t) to the physical space $\mathcal{P}$ with coordinates $(x,v_{||})$. Due to the symmetry, we are looking for the mapping only in the half of the parallel velocity range $[0,-lv]$.

Following the procedure described in [115, 116], we define four vector functions mapping the boundaries (projections of the boundaries) of the physical domain

$$\mathbf{P}_l = \begin{bmatrix} 0 \\ t \end{bmatrix}, \qquad \mathbf{P}_r = \begin{bmatrix} c_1\sigma(t) \\ t \end{bmatrix}, \tag{3.57}$$

$$\mathbf{P}_b = \begin{bmatrix} sc_1\sigma(x_{min}) \\ x_{min} \end{bmatrix}, \qquad \mathbf{P}_t = \begin{bmatrix} sc_1\sigma(x_{max}) \\ x_{max} \end{bmatrix}, \tag{3.58}$$

where subscripts l, r, b, t denote the left, right, bottom, and top boundaries of the domain, c_1 is a constant determining the width of the parallel velocity domain, σ is the standard deviation of the distribution from Equation (3.33)

$$\sigma(x) = \sqrt{\frac{T(x)}{m}} = \frac{v_T(x)}{\sqrt{2}}. \tag{3.59}$$

Next, we also need four projections of the corners of the physical domain

$$\mathbf{P}_{lb} = \begin{bmatrix} 0 \\ x_{min} \end{bmatrix}, \qquad \mathbf{P}_{lt} = \begin{bmatrix} 0 \\ x_{max} \end{bmatrix}, \tag{3.60}$$

$$\mathbf{P}_{rb} = \begin{bmatrix} c_1\sigma(x_{min}) \\ x_{min} \end{bmatrix}, \qquad \mathbf{P}_{rt} = \begin{bmatrix} c_1\sigma(x_{max}) \\ x_{max} \end{bmatrix}. \tag{3.61}$$

One way to find the mapping $\mathbf{T}$ is to use bilinear interpolation based on the values at the boundaries and in the corners; the transfinite bilinear interpolation is then given by

$$\begin{aligned}\mathbf{T}(s,t) = (x_{max}-s)\mathbf{P}_l + (s-x_{min})\mathbf{P}_r + (1-t)\mathbf{P}_b + t\mathbf{P}_t - (x_{max}-s)(1-t)\mathbf{P}_{lb} \\ - (x_{max}-s)t\mathbf{P}_{lt} - (s-x_{min})(1-t)\mathbf{P}_{rb} - (s-x_{min})t\mathbf{P}_{rt}.\end{aligned} \tag{3.62}$$

After substituting the projection functions in (3.62), we determine the mapping

$$\mathbf{T}(s,t) = \begin{bmatrix} sc_1\sigma(t) \\ t \end{bmatrix} = \begin{bmatrix} v_\parallel(s,t) \\ x(s,t) \end{bmatrix}. \tag{3.63}$$

From this transform, it follows that t corresponds to x and

$$s = \frac{v_\parallel}{c_1\sigma(x)} = \frac{\sqrt{2}}{c_1}\frac{v_\parallel}{v_T}. \tag{3.64}$$

If we choose the constant $c_1 = \sqrt{2}$, we obtain that $s = v'_\parallel$.

The same procedure can be performed for the $x - \mu$ domain shown in Figure 16. In this case, we introduce a transformation of magnetic moment coordinate

$$u = \frac{\lambda(x)}{c_2}\mu = \frac{B}{c_2 T(x)}\mu = \frac{1}{c_2}\frac{\mu}{\mu_T}. \tag{3.65}$$

By setting $c_2 = 1$, we return to the introduced normalized magnetic moment $u = \mu'$.

To summarize, the transfinite bilinear mapping to the desirable physical domain shown in Figure 13 leads to a coordinate transform equivalent to the normalization

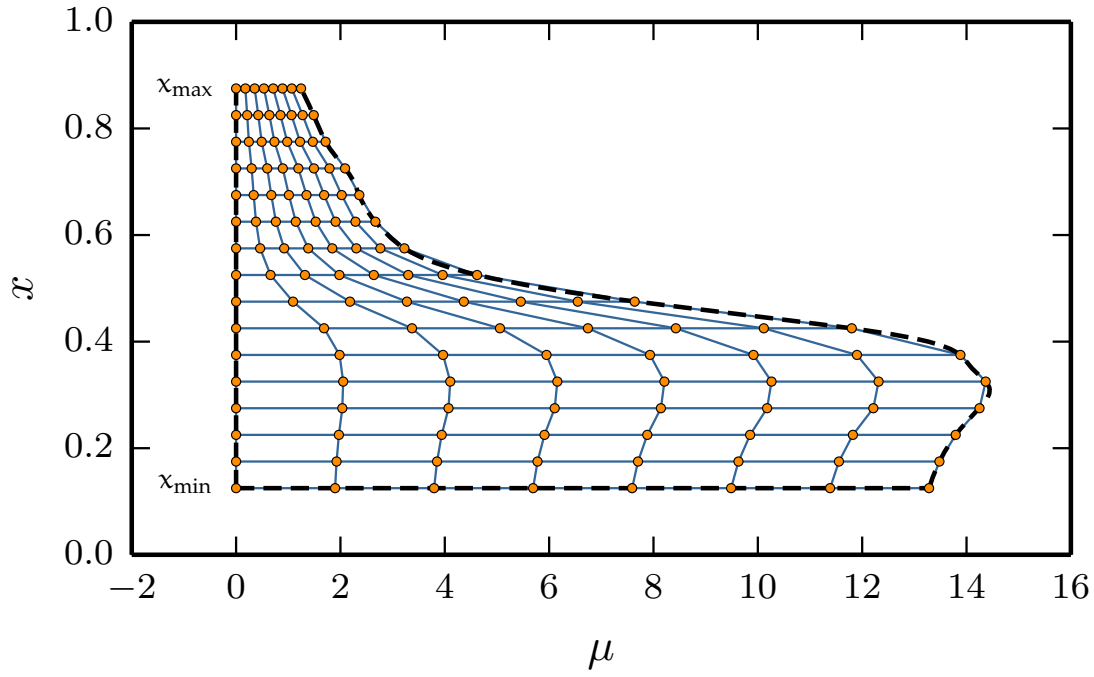


Figure 16: Physical space x – μ grid achieved by means of transfinite interpolation mapping.

introduced in (3.54). However, the transfinite interpolation technique provides more control over the resulting grid than mere normalization. For example, we can force the coordinate lines to go through certain points or even correspond to prescribed lines inside the domain (for details see [115, 116]). It should be noted that, in the provided transforms, it is assumed that the projection functions **P** are continuous. When this assumption does not hold, which happens frequently, for instance when temperatures given at discrete points (experimental measurements) lead to projection values **P** known only at these points, we can apply the discrete transfinite mapping techniques first introduced in [117, 118]. Moreover, bilinear interpolation, which was used in (3.62), is not the only choice and other interpolation techniques leading to different mappings can be applied.

Application to Gyrokinetic Equations

Because the so far used grids are based on a transformation of the physical coordinates to the logical ones with a rectangular domain, the underlying equations need to be modified. For the sake of simplicity, we consider the normalization transform (3.54) and point out the consequences of introducing a corresponding grid.

The introduction of new coordinates $v_{\parallel}'$ and μ' causes the transformation of the distribution functions, changes in derivatives and integrals. The last two modifications lead to non-trivial changes in the governing gyrokinetic equation, which might require an extensive revision of the existing implementation.

The radial derivative becomes a mixture of the radial and velocity coordinates derivatives:

$$\frac{\partial}{\partial x} = \frac{\partial}{\partial x'} + \frac{\partial v_\parallel'}{\partial x}\frac{\partial}{\partial v_\parallel'} + \frac{\partial \mu'}{\partial x}\frac{\partial}{\partial \mu'}\,. \tag{3.66}$$

We compute the radial derivatives of the modified potential $\bar{\chi}_1$ (see (2.25)) and "convenience" function denoted by Γ (see (2.27)). These derivatives appear in two terms of the gyrokinetic equation (2.26): the curvature and nonlinear terms.

Due to (3.66), the derivative in the magnetic moment direction has to be computed. This derivative is not present in the original equation, where only integration in the μ direction is performed. Therefore, with the new coordinates, the discretization of the magnetic moment must suit well both the quadrature computations and the high order[2] finite difference schemes used to discretize derivatives. As a consequence, the default choice of the Gauss-Laguerre nodes, which had led to significant savings, is not usable anymore; the transformed grid requires now more nodes in the magnetic moment direction. For example, some convergence tests reveal that for an equidistant grid instead we would need around $128 - 256$ points in order to obtain the same results as with eight Gauss-Laguerre nodes; this is $16 - 32$ as many grid points.

The nonlinear term is the most expensive term in the gyrokinetic equation affected by the new velocity coordinates. The computations associated with the nonlinear term might take more than half cf. the total computational time. The default discretization scheme of the nonlinear term in GENE uses an Arakawa scheme in a mixed Fourier and real space, see [119, 120]. The details on how this scheme is applied to the nonlinear term in GENE are provided in [44]. The form of the nonlinear term, which is discretized in this scheme, is given by

$$\begin{aligned} N = \frac{1}{3}\Bigg[&\left(\frac{\partial \bar{\chi}_1}{\partial y}\frac{\partial g_1}{\partial x} - \frac{\partial \bar{\chi}_1}{\partial x}\frac{\partial g_1}{\partial y}\right) + \frac{\partial}{\partial y}\left(\bar{\chi}_1\frac{\partial g_1}{\partial x} - \frac{\partial \bar{\chi}_1}{\partial x}g_1\right) \\ &+ \frac{\partial}{\partial x}\left(\frac{\partial \bar{\chi}_1}{\partial y}g_1 - \bar{\chi}_1\frac{\partial g_1}{\partial y}\right)\Bigg]\,. \end{aligned} \tag{3.67}$$

The derivatives in the binormal direction y are computed in the Fourier space, but they are transformed to the real space before multiplications are performed. After the Arakawa scheme is applied, another transform is necessary to return to the Fourier space in the y direction. If we apply directly the expression (3.67) to the nonlinear term after the normalization (3.54), we have to replace all radial derivatives by (3.66) and introduce velocity space derivatives. Even if the radial and velocity derivatives are computed before they are used in the nonlinear term, the third term in (3.67) still requires computing the derivatives in all three directions. It is hard to estimate the resulting overhead in the computations of the nonlinear term beforehand, due to the transform of the velocity space coordinates. However, if this overhead reaches even only ten percent, a similar slowdown is expected for the whole application just due to the nonlinear term.

The moments of the distribution function (2.34), which appear in the field equations and numerous diagnostics, are also affected by the transform of the velocity space coordinates, because they involve integration in the velocity and radial distance directions.

2 by default fourth in GENE

The new coordinates $v_{\parallel}'$ and μ' introduce further complexity into the moment computations. First, we have to transform the integral (2.34) to the new coordinates by introducing the Jacobian. Second, the gyro-phase averaging, which along with the original velocity coordinates corresponds to the path-integral on the projection of the physical trajectory of a particle on $x - y$ plane, loses its initial physical meaning, because we do not have fixed physical magnetic moments μ anymore, but rather transformed and radially dependent counterparts μ'. Instead of the gyro-radius defined in (2.36), we have to use the radially dependent expression:

$$|\boldsymbol{\rho}(\mathbf{x})| = \frac{c}{q}\sqrt{\frac{2\mu' B_T(\mathbf{x})}{B}}. \tag{3.68}$$

Therefore, the paths of the gyro-phase averaging integrals are not circular anymore. This makes the already expensive calculations of the gyro-matrix, described in Section 4.3, even more challenging.

One way to circumvent this issue with the modified gyro-phase averaging is to keep the original magnetic moment in the moment integral (2.34) and perform the gyro-averaging (2.35) for a fixed selected set of μ. In this case, however, we have to interpolate the distribution function, which is defined and solved at μ' points, at each time step. The advantage of this approach is that we can reuse already implemented and well-established moment computations of the original code.

To conclude, the elegant approach of the velocity coordinates normalization or the transfinite interpolation technique seem very promising to address the problem of the spatial temperature variation. However, the grids based on the coordinates transformation lead to non-trivial modifications of the underlying system of equations. First, we introduce originally absent derivatives in the magnetic moment direction, which require significantly more grid points. Second, the most expensive nonlinear term gets even more complicated, which might cause a further slowdown of computations. Furthermore, the computations of the gyrophase averaging are getting more challenging. Moreover, the introduction of new velocity coordinates requires a revision and reimplementation of the well-established and already tested existing codes for the classical velocity coordinates. In view of these complications, we propose instead the block-structured grids (see Chapter 4), which lead to similar savings in the number of grid points and allow reusing the original code.

3.4 SUMMARY

In this chapter, we introduced the coordinates in the position and velocity spaces, which are used to describe the gyrokinetic equation in GENE. For the position space, the three coordinates are aligned to the magnetic field lines; a mixture of fast spectral and finite difference methods was used to discretize these position coordinates. Regarding the velocity space grid, the default grid is of rectangular shape, with an equidistant mesh in the parallel velocity direction and Gauss-Laguerre nodes in the magnetic moment direction. These types of discretization were justified by the type of operations performed in the velocity space. Furthermore, the shape of the background distribution function, which has a strong influence on the structure of the fluctuating part, was shown to make Gauss-Laguerre nodes preferable for the μ discretization. We also demonstrated that the strong temperature variation along the radial distance leads to a

lot of nodes in the velocity space grid. To alleviate this issue, we normalized the velocity space coordinates by the thermal speed. However, this approach was shown to lead to non-trivial modifications of the governing equation, which could potentially significant computational overheads due to introduction of the magnetic moment derivatives, changes in the nonlinear term, and etc.

In the next chapter, we introduce an alternative approach to the thermal speed normalization. This technique results in the same theoretical savings as the normalization method and does not require discarding the physical velocity coordinates.

4

BLOCK-STRUCTURED GRIDS

In Chapter 3, we described the resolution challenges introduced to grid-based gyrokinetic simulations when incorporating radial temperature variations. Furthermore, we provided a solution by normalizing the velocity coordinates or, equivalently, introducing the transfinite interpolation grid, which leads to the same results. The drawback of the approach based on transformed coordinates is the extensive modifications introduced in the underlying gyrokinetic equations. We note here, that the problem is not specific to gyrokinetics.

The goal of the aforementioned method is to perfectly align the coordinate lines (surfaces) with the desired domain boundary, see, for instance, Figure 13. However, in gyrokinetic simulations, the exact shape of the velocity space boundary does not play an important role, as long as the velocity ranges are chosen sufficiently large. Therefore, in this chapter we introduce block-structured grids, which are focused on how to subdivide, in an optimal way, the mixed radial distance and velocity space domain into blocks, and adjust the range and resolution of each block mesh depending on the local temperature. An example of a block-structured grid outline with six blocks is shown in Figure 17. This outline was obtained by applying the blocking technique

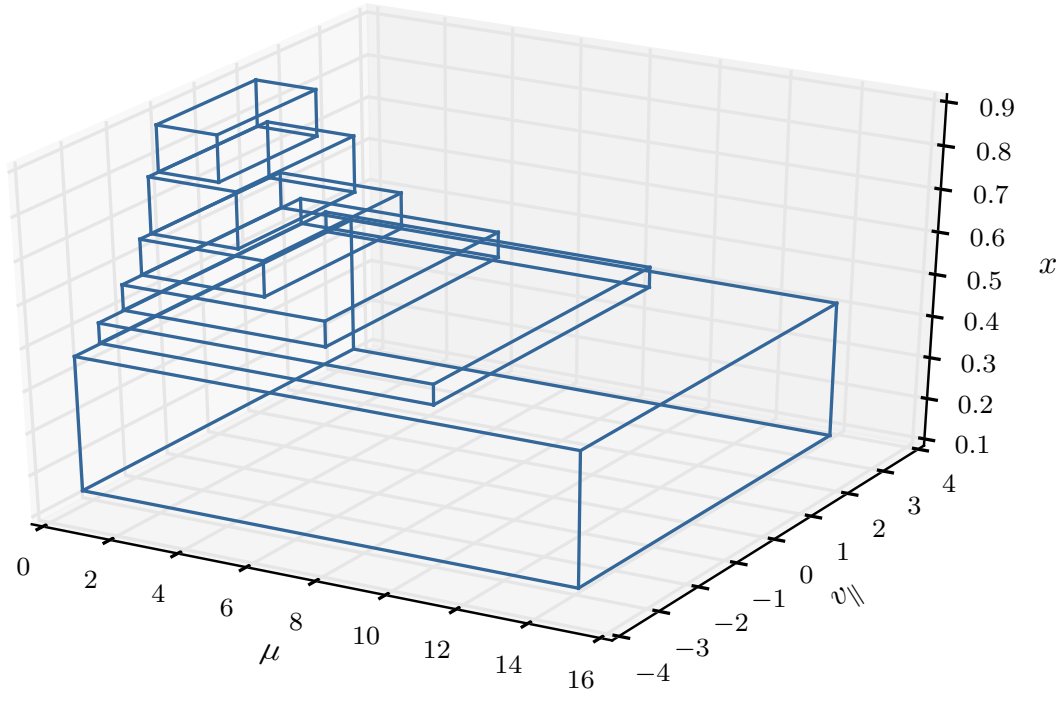


Figure 17: An example of a block-structured grid with six blocks in $x - v_\parallel - \mu$ subspace.

to the domain shown previously in Figure 13. Unlike the grids introduced in Subsection 3.3.3, block-structured grids are not boundary conforming, but approximate the desired boundary with a step-like function. Relaxing the requirement of aligning coor-

dinate lines to boundaries allows us to avoid transforming the underlying gyrokinetic equation.

The proposed block-structured grids are related to overlapping or overset grids [121, 122], since both types of grids use interpolation during data-exchange between blocks. The latter grids are typically applied to represent complex geometries and discretize areas with different physical models. In gyrokinetic simulations, however, an accurate representation of the shape of the simulation domain in the velocity space is not important. The focus here is on a careful choice of the range and resolution of the velocity grid, which depend on the temperature at a given radial distance.

Block-structured grids are also frequently used for an adaptive mesh refinement in the context of plasma simulations. Such grids have been employed for PIC simulations in [123], and for Vlasov two-dimensional (a position and a velocity coordinate) simulations in [124]. Locally refined block-structured grids are also used for fluid plasma simulations, see [125, 126]. Then, for edge plasma simulations, block-structured grids correspond to multiple connected grids in the logical computational domain, where each block-grid discretizes a simulation subdomain with different physical properties, see [127].

The rest of the chapter is structured as follows. We first introduce the concept of block-structured grids for the parallel velocity direction and compare the results with those of the transformation approach. Then we explain a blocking technique for the magnetic moment coordinate, which is conceptually different, because, unlike in the parallel velocity direction, the focus is shifted on computing quadrature-like operations instead of derivatives. Special attention is dedicated to the gyrophase averaging operation, because this is where the new technique requires the most extensive modification. At the end of the chapter, we estimate the theoretical reduction in the number of grid points in the block-structured grids compared to the equivalent regular grids.

4.1 PARALLEL VELOCITY DIRECTION

The proposed block-structured grids can be constructed independently in the $x - v_{\parallel}$ and $x - \mu$ subspaces. It is easier to demonstrate the concept behind the construction for the parallel velocity coordinate, because we use an equidistant meshing for this direction in GENE. After introducing the block-structured grid in the $x - v_{\parallel}$ subspace, the extension to include the magnetic moment direction is straightforward. Therefore, in this section we discuss constructing the grids in the $x - v_{\parallel}$ subspace only.

Constructing a block-structured grid consists of two main steps: first, the domain contour is approximated by a step-like curve, which gives the positions of the block boundaries, and then a number of $v_{\parallel}$ grid points is chosen in each block. The result of the first step is demonstrated in Figure 18, which is computed for the TCV temperature profile, see Figure 9 (bottom).

Approximating the domain contour can be regarded as finding the positions of block boundaries in the radial direction. When these positions are given, we know the height (radial extent) of each block and are thus able to choose a minimum width (parallel velocity extent) of each block, so that the desired domain is inside of the obtained step-shaped contour. The radial positions of the block boundaries can be determined in several ways. For example, good boundary locations (exhibiting a reasonably small temperature variation within one block) can be obtained by minimizing the area dif-

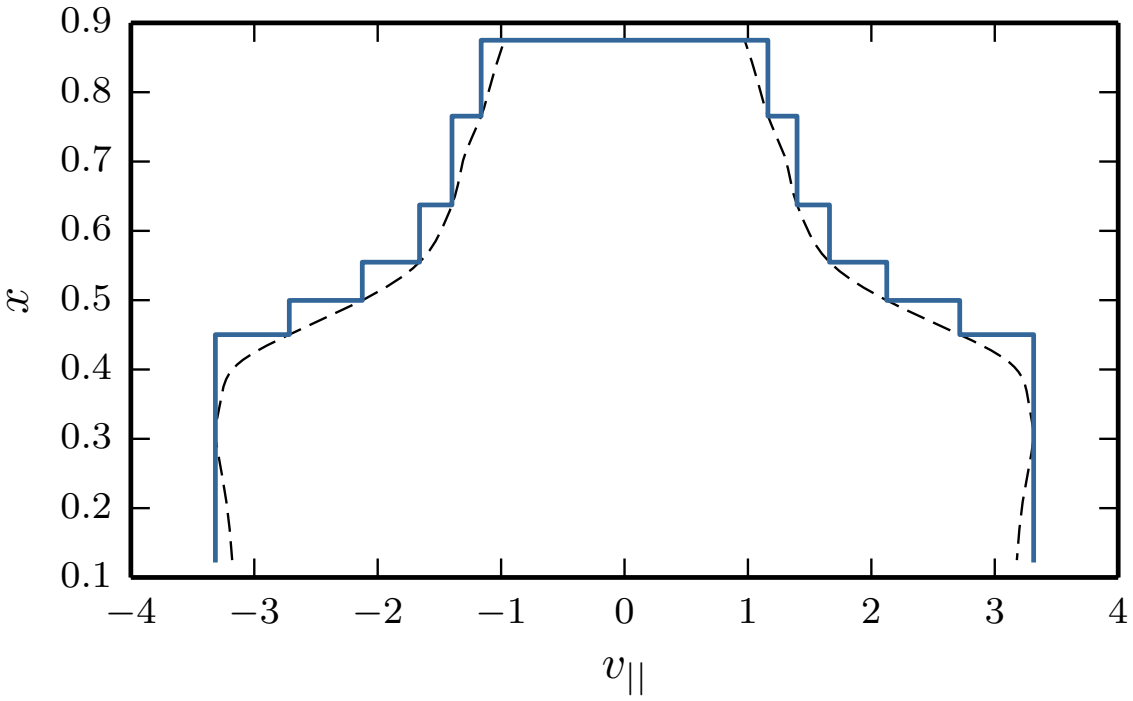


Figure 18: An approximation of the $x - v_{||}$ subdomain contour by a step-like curve. The dashed line corresponds to the contour and the solid blue line to the blocked approximation.

ference between the desired and blocked domains. This is the default way to obtain block-structured grids, for more details we refer to the description of the grid generation tool in Appendix Section A.1.

After the shape and position of the blocks are computed, we have to decide upon the number of grid points in the parallel velocity direction. There are two natural choices. The first choice is to fix the resolution in each block, so that grid lines are aligned on the block boundaries, see Figure 19 (top). We refer to these grids as first type. The block-structured grids of the first type require a minimum modification of the original code written for the regular grids. We can obtain these grids by simply disregarding those areas of the regular grid that are outside the blocked domain. However, the first type grids do not result in a significant reduction in the number of grid points, for details see Section 4.4.

The second type block-structured grids are obtained by following the approach of constructing a $x - v_{||}$ grid for radially global simulations based on the resolution that would be employed in local simulations. According to this approach, we have to use the same number of grid points in each block. Consequently, we obtain the second type grid demonstrated in Figure 19 (bottom). This grid leads to a significantly higher reduction in the discretization points, but it is also more difficult to implement, due to the misalignment of the coordinate lines at the block boundaries. We address this misalignment by applying interpolation on the block boundaries, for explanations see Section 5.2. Formally, the block-structured grids of the first and second type can be considered as two successive steps towards the normalized grid presented in Subsection 3.3.3, which also preserve the governing equation.

Next, we provide a qualitative comparison between four types of introduced grids: regular, normalized, and the block-structured grids of first and second type. All these

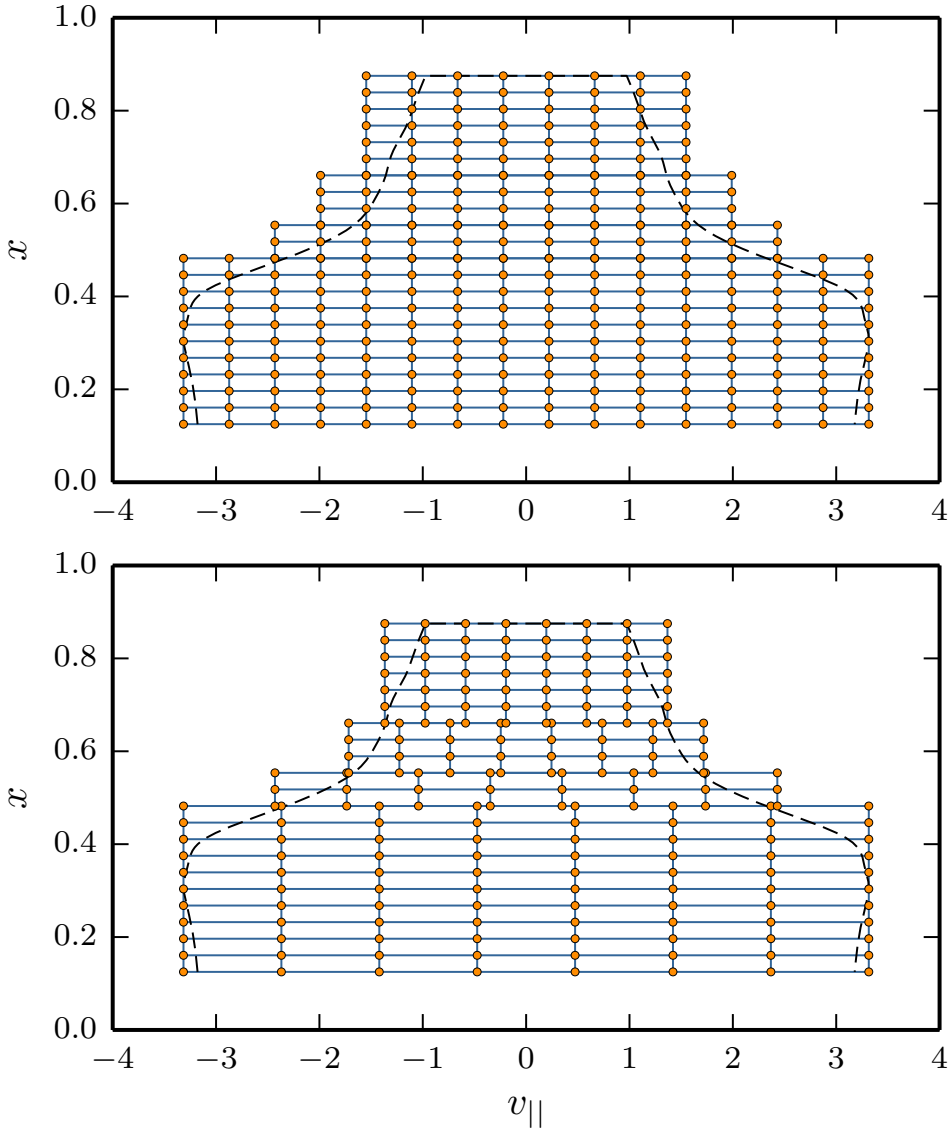


Figure 19: Examples of the first (top) and the second (bottom) types of block-structured grids. For the sake of visibility, the resolutions of the grids are significantly coarser than in practice.

four types of grids are schematically presented in Figure 20, where the four top subplots (**1, 2, 3, 4**) represent the $v_{||}$ coordinate lines in the $x - v_{||}$ subspace, whereas the four bottom subplots (**5, 6, 7, 8**) represent the coordinate lines in the $x - v_{||}/v_T$ subspace. The grid with normalized velocity space is labeled (**1**) and (**5**) in the two different coordinate systems, the default regular grid (**2, 6**), the block-structured grid of the first type (**3, 7**), and the block-structured grid of the second type (**4, 8**).

As it was discussed in Subsection 3.3.2, the default regular grid requires a sufficiently wide range and a fine enough resolution to satisfy both high and low temperature regions. Consequently, it necessitates more grid points than the grid with the normalized parallel velocity. A comparison of the normalized and regular grids in the $x - v_{||}/v_T$ subspace ((**5**) and (**6**)) reveals both the overestimated number of $v_{||}$ nodes in the high temperature region and overly wide range in the low temperature region. The block-structured grid of the first type solves the problem of the unnecessary wide range in the

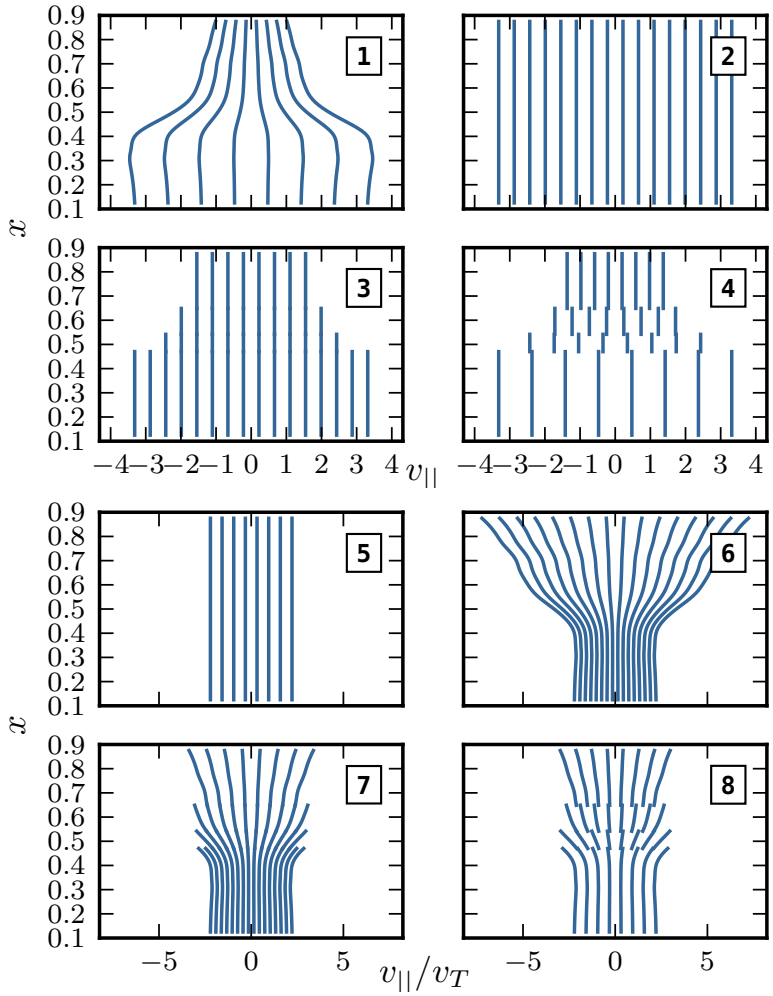


Figure 20: Examples of $v_{||}$ coordinate lines in different coordinate systems: in the top subplots (**1, 2, 3, 4**) the horizontal axis is the physical parallel velocity coordinate, while in the bottom subplots (**5, 6, 7, 8**) the horizontal axis is the normalized parallel velocity coordinate. The normalized grid is represented in (**1, 5**), the default regular grid in (**2, 6**), the block-structured grid of the first type in (**3, 7**), and the block-structured grid of the second type in (**4, 8**).

low temperature region by cutting away the insignificant grid points, see plots (**3**) and (**7**). However, by comparing (**5**) with (**7**), we observe that the high temperature region is still overresolved. The next step, which leads to appropriate ranges and resolutions, see plots (**4, 8**), is achieved by the block-structured grid of the second type.

Visually, the second type of block-structured grids is the closest match to the grid with the normalized velocity coordinates. Theoretically, for infinitely many blocks, these grids should correspond graphically to each other. Practically, however, the number of blocks necessary for the gyrokinetic simulations in the block-structured grids of the second type is quite small. To determine an optimal number of the blocks, we look at how the number of grid points depends on the number of blocks.

For example, Figure 21 illustrates such a dependence for the previously introduced TCV temperature profile (see Figure 9 (bottom)). In this figure, the corresponding block-structured grids of the second type are constructed based on a regular reference

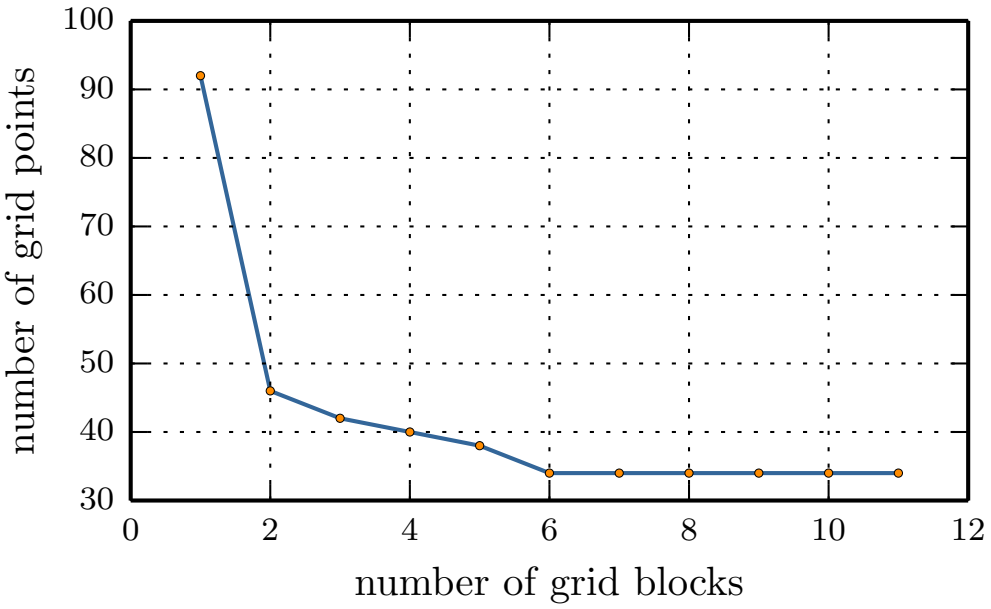


Figure 21: Dependence of the number of grid points on the number of blocks in the block-structured grid of the second type. The TCV temperature profile was used for the grid construction; the reference regular grid has $\texttt{nx0} = 512$ and $\texttt{nv0} = 92$.

grid with $\texttt{nx0} = 512$ grid points in the radial direction and $\texttt{nv0} = 92$ parallel velocity grid points. The construction procedure ensures that the parallel velocity range of the widest block is equal to the range of the reference grid and the parallel velocity resolution in the narrowest block is equal to the resolution of the reference grid. From Figure 21, we observe that six blocks is an optimal number; taking more than six blocks would not lead to a reduction in the number of grid points.

In the following section, we extend the block-structured grids to include the magnetic moment direction μ.

4.2 MAGNETIC MOMENT DIRECTION

The technique of the first type block-structured grids can be directly applied to the $x - \mu$ subspace. This procedure for the magnetic moment direction is introduced for reasons similar to those in the parallel velocity direction: the background distribution outside the step approximated contour is negligibly small; the same holds true for the fluctuating part, which, according to the gyrokinetic ordering, is even smaller. Therefore, without loss of accuracy, we can ignore the areas outside the desired contour of the simulation domain. This is achieved by introducing a block-structured grid of the first type in the radial distance – magnetic moment plane, as shown in Figure 22. This grid is constructed based on the TCV temperature profile shown in Figure 9 (bottom).

The step-like contour approximation is obtained in the same way as for the $x - v_{\parallel}$ space. Theoretically, we have to construct the step-shaped approximations for both planes $x - v_{\parallel}$ and $x - \mu$; then, the procedure has to be extended to the $x - v_{\parallel} - \mu$ volume, as demonstrated in Figure 17. In practice, however, it is sufficient to determine

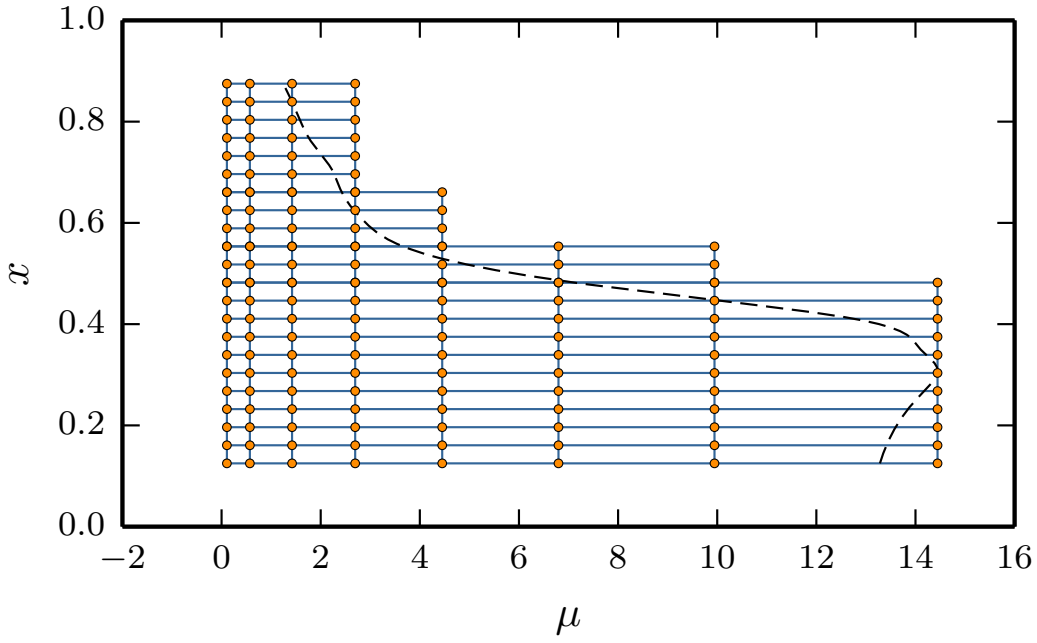


Figure 22: An example of the first type of block-structured grids in the $x - \mu$ plane. For the sake of visibility, the resolutions of the grid are significantly coarser than in practice.

the radial extents and positions of the blocks for one plane only, e. g., $x - v_{\parallel}$, and then use this information to determine the step-like contour on both planes.

Integration and gyrophase averaging are the main operations performed in the magnetic moment direction. Therefore, we employ the Gauss quadrature rule to compute integrals efficiently by using a minimum number of nodes. Due to the exponential shape of the background distribution function in the $x - \mu$ plane, the Gauss-Laguerre rule is an optimal choice for our case. This rule is expressed by

$$\int_0^\infty e^{-\mu} p(\mu)\, d\mu = \sum_{m=1}^{\texttt{nw0}} w_m p(\mu_m)\,, \tag{4.1}$$

where μ_m and w_m are the nodes and weights of the quadrature rule. In the previous expression, we denoted by `nw0` the number of μ nodes and by $p(\mu)$ a polynomial of maximum degree $2 \cdot \texttt{nw0} - 1$.

In the case of the regular grid and the block-structured grid of the first type, we use the same set of nodes μ_m and weights w_m to integrate functions at all radial positions. In the block-structured grid of the first type, the number of μ nodes (`nw0`) is different in each block, e. g., in Figure 22, the first block contains eight nodes and the last only four. The block-structured grids of the first type remove those nodes from the quadrature rule with a negligible contribution to the sum in (4.1).

In the Gauss-Laguerre rule (4.1), we assume that the integrated function decays exponentially $\exp(-\lambda\mu)$ with a factor $\lambda = 1$. However, the background distribution function (3.39) decays with an exponential factor given by $\lambda(x) = T_{\text{ref}}/T(x)$. Therefore, for a strong temperature variation in the radial direction, the quadrature rule (4.1) is precise (optimal) only at the reference radial distance ($x_0 = 0.5$), where the exponential factor is $\lambda = 1$. At radial positions close to the boundaries, the precision of this quadrature

rule deteriorates drastically, as shown in Figure 23 by the green dash-dotted line. In this

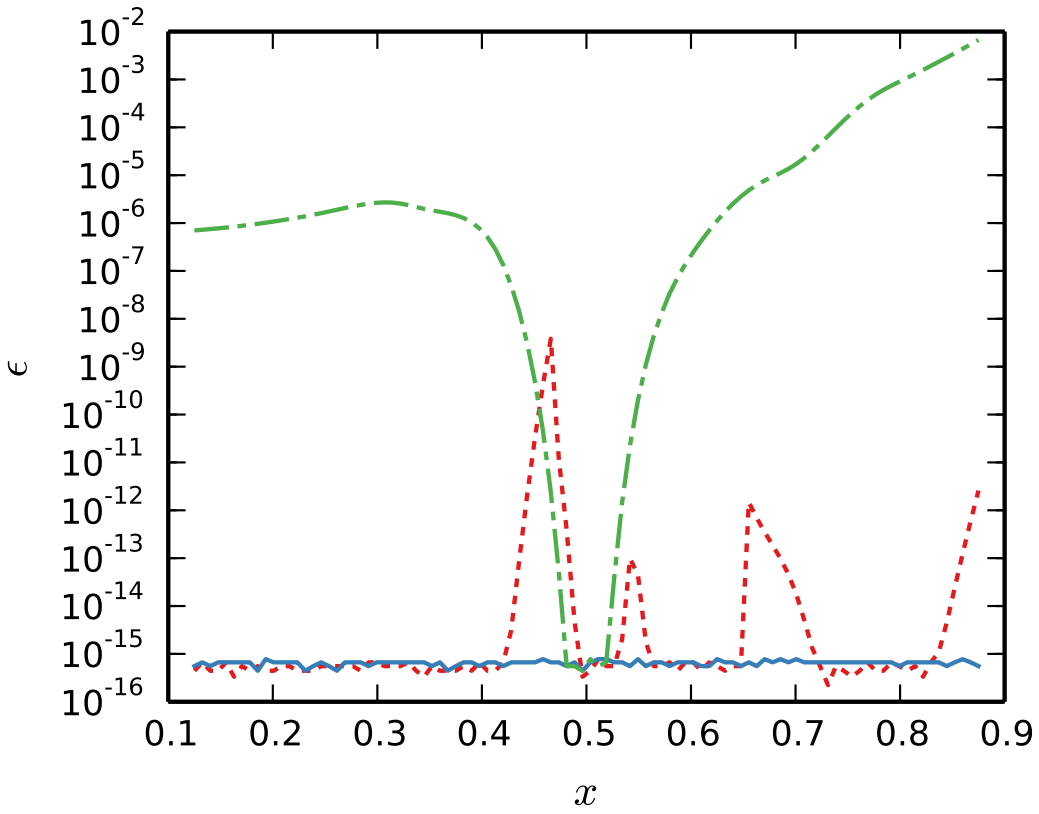


Figure 23: Quadrature error ϵ dependence on radial position x for three types of Gauss-Laguerre rules (with `nw0` $= 8$): the blue solid line — ideally adjusted quadrature rule at each radial point, the dash-dotted line — the quadrature rule of the regular grid, which is optimal at the reference point, the red dotted line — the quadrature rule is chosen separately for each block in the radial direction (four blocks in total).

figure, we demonstrate the error of the Gauss-Laguerre rules (with `nw0` $= 8$) applied to the background distribution function with the TCV temperature radial profile.

To reduce the error of the quadrature rule for the regular grid in the vicinity of the radial boundaries, we have to take more grid points than it is necessary for the case of radially local simulations or global simulations with a constant temperature profile. Alternatively, we can adjust the rule given by (4.1) at each radial position. This is achieved by scaling the node positions and weights by the exponential factors $\mu_m/\lambda(x)$ and $w_m/\lambda(x)$. In this case, the integration error of the background distribution is naturally at the machine precision level for all x (see the blue solid line in Figure 23). However, such adjustments displace the positions of the μ grid points, so they do not appear on the straight $\mu = \text{const}$ coordinate lines in the $x - \mu$ plane. Consequently, we cannot directly apply a finite difference scheme, and have instead to interpolate the distribution function in order to compute radial derivatives at each time step. Such an approach would, however, increase the computational costs tremendously.

Therefore, we introduce the block-structured grids of the second type in the $x - \mu$ plane. The underlying idea is to adjust the Gauss-Laguerre quadrature rule in each

grid-block. For example, we can make the rule optimal at the middle of the radial extent of each block. This rule is accurate for the integration of the background distribution function, as shown in Figure 23 by the red dotted line. In this plot, the block-structured grid of the second type does not achieve machine precision in areas close to the block boundaries. However, achieving machine accuracy is not strictly necessary, because, during simulations, we integrate the fluctuating part of the distribution function, which retains the shape of the background only approximately. Furthermore, a lot of computations in GENE use approximations such as those given by finite difference schemes.

An example of a block-structured grid of the second type is shown in Figure 24. Like in the case of the second type block-structured grids in the $x - v_{\parallel}$ subspace (Sec-

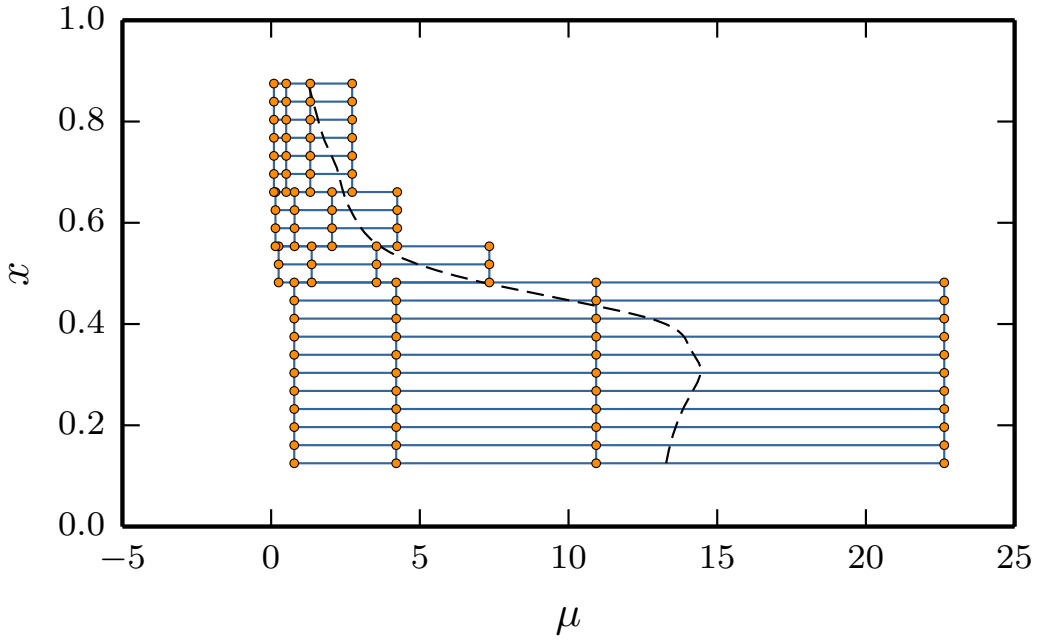


Figure 24: An example of the second type of block-structured grids in the $x - \mu$ plane. For the sake of visibility, the resolutions of the grid are significantly coarser than in practice.

tion 4.1), the computations of the radial derivative do not change inside the grid-blocks and require modifications only close to the block boundaries, due to misaligned coordinate lines ($\mu = \mathrm{const}$). Therefore, like in the $x - v_{\parallel}$ subspace, we solve the misalignment issue by interpolating the distribution function before applying a finite difference scheme; for the implementation details we refer to Section 5.2.

The oscillatory nature of the fluctuating part of the distribution function and the complex gyrophase averaging operation require a bigger number of μ nodes than is necessary for a mere integration of an analytic function. Even if the high number of nodes in the magnetic moment direction helps resolve the perturbed part of the distribution function better, it also results in the last μ_m being far away from the $\mu = 0$ axis. For example, in Figure 24, the quadrature rule with just four μ nodes yields the last grid point outside the prescribed domain denoted by the dash line.

This is an undesirable effect, because the background distribution function in the areas far from $\mu = 0$ is very close to zero and the fluctuating part, due to the gyroki-

netic ordering, even smaller. Thus, the precision of any floating point operation will be insufficient for consistent computations of the distribution function at particularly large μ values. If the problem is not addressed, one might get spurious results in the grid points positioned far away from $\mu = 0$.

One way to tackle this issue is to rescale the given nodes μ_m and weights w_m to ensure that all nodes fit in the prescribed range and the Gauss-Laguerre quadrature rule is still valid. The rescaling is the default procedure used for the regular grid, e.g, the construction of the first type block-structured grid shown in Figure 22 was based on such a rescaled regular grid, by effectively removing grid points outside the domain contour. An example of the block-structured grid of the second type with rescaled μ nodes is shown in Figure 25.

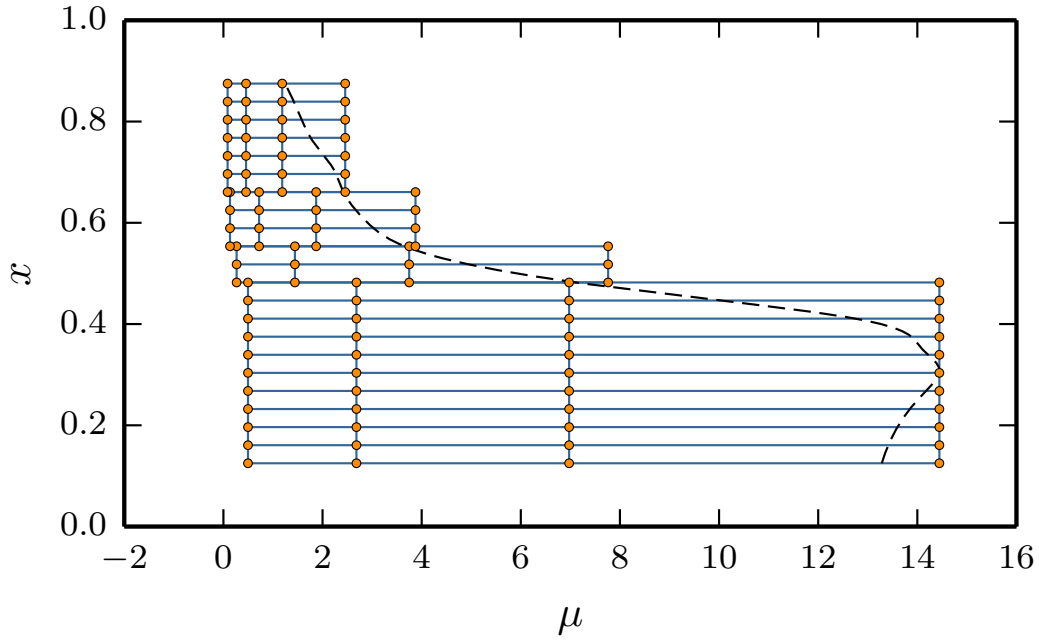


Figure 25: An example of the second type of block-structured grids in the $x - \mu$ plane, with rescaled μ nodes to fit inside a prescribed domain. For the sake of visibility, the resolutions of the grid are significantly coarser than in practice.

A side effect of the rescaling operation is the modification of the quadrature rule, which was adjusted for the background distribution at the given radial point. In spite of the deviation from the initially adjusted quadrature rule, in practice, the rescaling technique results in block-structured grids that still require a small number of μ nodes, for example, see Section 6.1.

For the sake of completeness, we describe an alternative, albeit more complex, approach for the case when the simple rescaling operation deviates too much from the initial quadrature rule. The technique preserves the optimal quadrature rule for the given background and keeps the grid nodes within the given magnetic moment range.

Instead of integrating from zero to infinity, as done in (4.1), we take a finite range integral

$$\int_0^{\mu_{max}} e^{-\mu} p(\mu)\, d\mu = \sum_{m=1}^{nw\theta} w_m p(\mu_m)\,, \tag{4.2}$$

where μ_{max} is sufficiently large; this limit is usually computed by $\mu_{max} = -\ln(1-p)$, where p is a given confidence level. In the following, we mark by superscripts $^{\lambda}$ the nodes and weights of the quadrature rule with an arbitrary exponential factor $\lambda \neq 1$. The Gauss-Laguerre rule generalized for λ is then given by

$$\int_0^{-\ln(1-p)/\lambda} e^{-\lambda\mu} p(\mu)\, d\mu = \sum_{m=1}^{nw\theta} w_m^{(\lambda)} p(\mu_m^{(\lambda)})\,. \tag{4.3}$$

There is a certain relation between the generalized quadrature rule and the previously introduced Gauss-Laguerre rule (4.2) for $\lambda = 1$. Namely, by changing the variable $x = \mu/\lambda$ in the integral in (4.3), we obtain

$$\int_0^{-\ln(1-p)/\lambda} e^{-\lambda\mu} p(\mu)\, d\mu = \frac{1}{\lambda} \int_0^{\ln(1-p)} e^{-x} p(x)\, dx = \sum_{m=1}^{nw\theta} \frac{w_m}{\lambda} p\left(\frac{x_i}{\lambda}\right)\,. \tag{4.4}$$

Hence, we conclude that $w_m^{(\lambda)} = w_m/\lambda$ and $\mu_m^{(\lambda)} = \mu_m/\lambda$. Efficient computations of the generalized Gauss quadrature are provided in [128], see also Chapter 18 in [129]. The same scaling rule is valid for integrals from zero to infinity.

The discretization of the magnetic moment coordinate affects not only the numerical integration but also the gyrophase averaging. In the following section, we provide details on how the corresponding numerical operation is realized for block-structured grids.

4.3 GYROPHASE AVERAGING

In GENE, two types of gyrophase averaging are employed. The first type is applied on the electromagnetic field potentials and marked by an overbar. This kind of gyrophase averaged potentials appear explicitly in the gyrokinetic Vlasov equation (2.26). The corresponding mathematical operation is given by taking the integral over the gyrophase angle θ

$$\overline{\phi}(\mathbf{X}) = \frac{1}{2\pi} \int_0^{2\pi} \phi(\mathbf{X} + \boldsymbol{\rho}(\mathbf{X}))d\theta\,, \tag{4.5}$$

where $\mathbf{X} = (X, Y, Z)$ is the gyro-center position and $\boldsymbol{\rho}$ the gyro-radius, whose length is given by (2.36).

Physically, this corresponds to projecting the particle's gyro-trajectory on the plane perpendicular to the magnetic field line and taking the average of the fields on the projection. A circular example $|\boldsymbol{\rho}(\mathbf{X})| = \text{const}$ of this projection is shown in Figure 26.

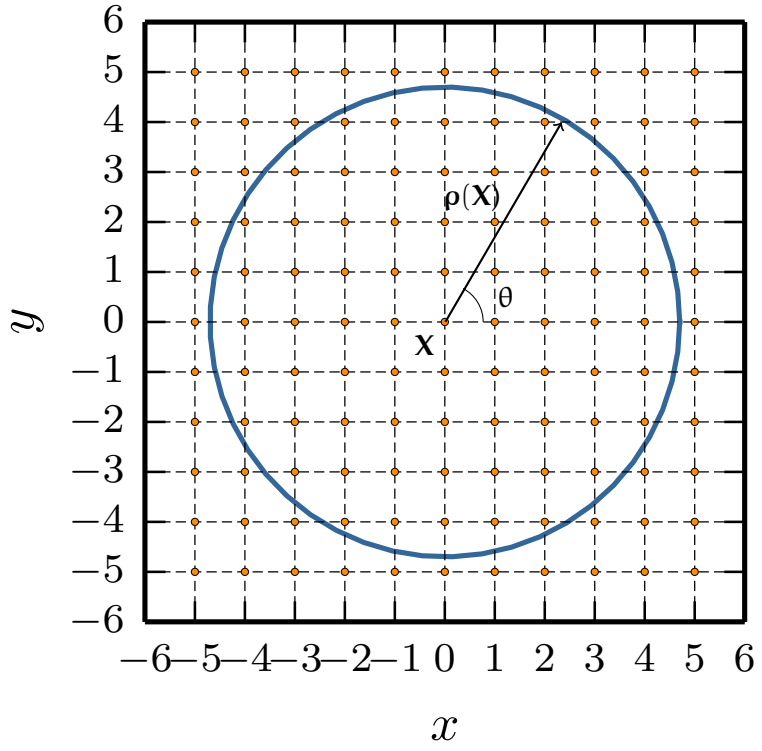


Figure 26: An example of a projection of the circular gyro-trajectory on the $x - y$ plane perpendicular to the magnetic field. For the sake of simplicity, the (x, y) coordinates system is orthogonal. Values of the potential ϕ are known only at the mesh nodes, an example of such a mesh is marked by dashed lines and circular nodes.

The second type of gyro-averaging is used when computing the moments of the distribution functions and the gyro-averaged potentials. This type of the gyro-averaging is defined by

$$\langle f \rangle (\mathbf{x}) = \frac{1}{2\pi} \int_0^{2\pi} \int \delta(\mathbf{X} + \boldsymbol{\rho}(\mathbf{X}) - \mathbf{x}) f(\mathbf{X}) d^3X d\theta \,. \tag{4.6}$$

This operation can be interpreted physically as taking the average over the gyrophase angle of $f(\mathbf{X})$, where $\mathbf{X}$ are the gyro-centers of trajectories passing through $\mathbf{x}$.

As shown in [43, 44], the discretized versions of both types of gyrophase averaging are related. Therefore, in the following, we consider only the first type of gyrophase averaging for fields.

Equation (4.5), introduced previously for the gyrophase averaging for fields, has to be rewritten to take into consideration that, in GENE, the binormal component Y of the gyro-center position $\mathbf{X}$ is discretized in the Fourier space. Therefore, we use $\phi(X, k_y, Z)$ values instead of $\phi(X, Y, Z)$, where k_y is the wave length of the toroidal mode. As a result, the potential ϕ in the configuration space (the position space, in this particular case) is computed by

$$\phi(\mathbf{X} + \boldsymbol{\rho}(\mathbf{X})) = \frac{1}{2\pi} \sum_{k_y} \phi(X + \rho_x(\mathbf{X}), k_y, Z + \rho_z(\mathbf{X})) e^{ik_y (Y + \rho_y(\mathbf{X}))} \,, \tag{4.7}$$

where, due to the field-aligned position space coordinate system, $\rho_z = 0$. Consequently, the gyrophase averaging takes the following form

$$\overline{\phi}(\mathbf{X}) = \frac{1}{2\pi} \sum_{k_y} \int_0^{2\pi} \phi(X + \rho_x(\mathbf{X}), k_y, Z)) e^{ik_y(Y+\rho_y(\mathbf{X}))} d\theta \,. \tag{4.8}$$

To be able to carry out the integration in (4.8), a continuous representation of ϕ is necessary. However, the function ϕ is known only at the computational grid points (X_i, k_y). The grid in Figure 26 serves as a discretization example in the $X - Y$ plane, where both coordinates are discretized in the configuration space (X_i, Y_j).

In the global GENE simulations, only X is discretized in the configuration space, while Y is discretized in the Fourier space. Therefore, on the one hand, we have a continuous representation of the potential ϕ in the Y direction given by the trigonometric interpolation (4.7). On the other hand, we know this function only at discrete X grid points and have to interpolate. Hence, we approximate the potential by using finite-element-like base functions with local support in the X direction. As a result, ϕ is expressed by a linear combination of the base functions

$$\phi(X, k_y, Z) = \sum_i \phi(X_i, k_y, Z) \Lambda_i(X) = \boldsymbol{\Lambda} \cdot \boldsymbol{\phi} \,. \tag{4.9}$$

After substituting (4.9) into (4.8), we obtain the gyro-averaged function at the gyro-center position $\mathbf{X}_k = (X_k, Y, Z)$

$$\begin{aligned} \overline{\phi}(\mathbf{X}_k) = \frac{1}{2\pi} \sum_{i,k_y} \int_0^{2\pi} \phi(X_i, k_y, Z) \Lambda_i(X_k + \rho_x(\mathbf{X}_k)) e^{ik_y(Y+\rho_y(\mathbf{X}_k))} d\theta \\ = \sum_{i,k_y} e^{ik_y Y} \mathcal{G}_{k,i} \phi_i = \sum_{k_y} e^{ik_y Y} \mathcal{G} \cdot \boldsymbol{\phi} \,. \end{aligned} \tag{4.10}$$

The matrix $\mathcal{G}$ is called the gyro-averaging matrix and is expressed by

$$\mathcal{G}_{ki}(k_y, z, \mu) = \frac{1}{2\pi} \int_0^{2\pi} \Lambda_i(X_k + \rho_x(\mathbf{X}_k)) e^{ik_y \rho_y(\mathbf{X}_k)} d\theta \,. \tag{4.11}$$

The gyro-radius projections on x and y axis in the field-aligned curvilinear coordinate system are given by

$$\begin{aligned} \rho_x(\mathbf{X}_k) &= \rho(\mathbf{X}_k) \sqrt{g^{xx}(\mathbf{X}_k)} \cos\theta \,, \\ \rho_y(\mathbf{X}_k) &= \frac{\rho(\mathbf{X}_k)}{\sqrt{g^{xx}(\mathbf{X}_k)}} \left(g^{xy}(\mathbf{X}_k) \cos\theta - \sqrt{\gamma_1} \sin\theta \right) \,. \end{aligned} \tag{4.12}$$

In this expression, we use metric coefficients taken from a metric tensor, which is defined by

$$\begin{aligned} g &= \begin{pmatrix} g^{xx} & g^{xy} & g^{xz} \\ g^{yx} & g^{yy} & g^{yz} \\ g^{zx} & g^{zy} & g^{zz} \end{pmatrix} \,. \\ g^{ij} &= \boldsymbol{\nabla} u^i \cdot \boldsymbol{\nabla} u^j \,, \end{aligned} \tag{4.13}$$

where by $(u^{(1)}, u^{(2)}, u^{(3)}) = (x, y, z)$ we denote the curvilinear field aligned coordinates.

Then, γ_1 is given by

$$\gamma_1 = g^{xx} g^{yy} - g^{xy} g^{yx} . \tag{4.14}$$

In GENE (see [43]), the choice of the base functions relies on an expansion with polynomials of odd degree p

$$\phi(x) = \sum_{n=1}^{nx0} \sum_{m=0}^{(p-1)/2} \left. \frac{\partial^m \phi(x)}{\partial x^m} \right|_{x=x_n} P_{m,n}(x) , \tag{4.15}$$

where `nx0` is the number of grid points in the radial direction. The local support polynomials are defined by

$$\left. \frac{\partial^u P_{m,n}}{\partial x^u} \right|_{x=x_j} = \delta_{jn} \delta_{um} , \qquad j = n, n+1 , \qquad u = 0, \ldots, \frac{p-1}{2} . \tag{4.16}$$

According to (4.15), the derivatives of ϕ are necessary in order to perform interpolation. These derivatives can be computed only numerically by applying a finite difference scheme. The stencils for derivative computations correspond to those used to discretize the governing equations. As a result, the polynomial expansion is given by

$$\phi(x) = \sum_{m=0}^{(p-1)/2} \mathbf{P}_m \mathcal{D}^m \boldsymbol{\phi} , \tag{4.17}$$

where $\mathbf{P}_m = (P_{m,1}, \ldots, P_{m,nx0})$ is a vector of polynomials, $\mathcal{D}^m$ is the m-order derivative band matrix (rows correspond to stencil), and $\boldsymbol{\phi} = (\phi_1, \ldots, \phi_{nx0})^T$ is a vector with discretized function values.

Comparing expressions (4.9) and (4.17), we observe that the base functions are given by

$$\boldsymbol{\Lambda} = \sum_{m=0}^{(p-1)/2} \mathbf{P}_m \mathcal{D}^m . \tag{4.18}$$

A substitution of this base function into (4.11) yields the gyro-averaging matrix, with rows of the form

$$\mathcal{G}_{k*}(k_y, z, \mu) = \frac{1}{2\pi} \sum_{m=0}^{(p-1)/2} \left(\int_0^{2\pi} \mathbf{P}_m(X_k + \rho_x(\mathbf{X}_k)) e^{i k_y \rho_y(\mathbf{X}_k)} d\theta \right) \mathcal{D}^m . \tag{4.19}$$

To simplify the notation, we introduce a θ-integral-matrix $\mathcal{Q}^m$, with elements given by

$$\mathcal{Q}^m_{kn} = \frac{1}{2\pi} \int_0^{2\pi} P_{m,n}(X_k + \rho_x(\mathbf{X}_k)) e^{i k_y \rho_y(\mathbf{X}_k)} d\theta . \tag{4.20}$$

With this notation, the gyro-averaging matrix is expressed by

$$\mathcal{G}(k_y, z, \mu) = \sum_{m=0}^{(p-1)/2} \mathcal{Q}^m \mathcal{D}^m . \tag{4.21}$$

The gyro-averaging matrix $\mathcal{G}$ is a band matrix, because it is a sum of products of the band-structured matrices $\mathcal{Q}$ and $\mathcal{D}$.

In GENE, the gyro-averaging matrix corresponding to the gyrophase averaging defined by (4.6) is by default approximated by $\mathcal{G}^\dagger$, e. g., see [44].

During the gyrokinetic simulations in GENE, we have to compute the gyrophase averages for the given set of magnetic moment nodes. In Figure 27 (left), we show a set of gyro-trajectories for a given discretization of the magnetic moment μ in the default regular grid. In the case of block-structured grids, see Figure 27 (right), due to different μ nodes at different radial positions, the corresponding gyro-trajectories are not closed. As a result, we cannot compute the gyrophase averaging (4.5) in the same way as for the regular grid. Nevertheless, this type of gyro-averaging is carried out only for the

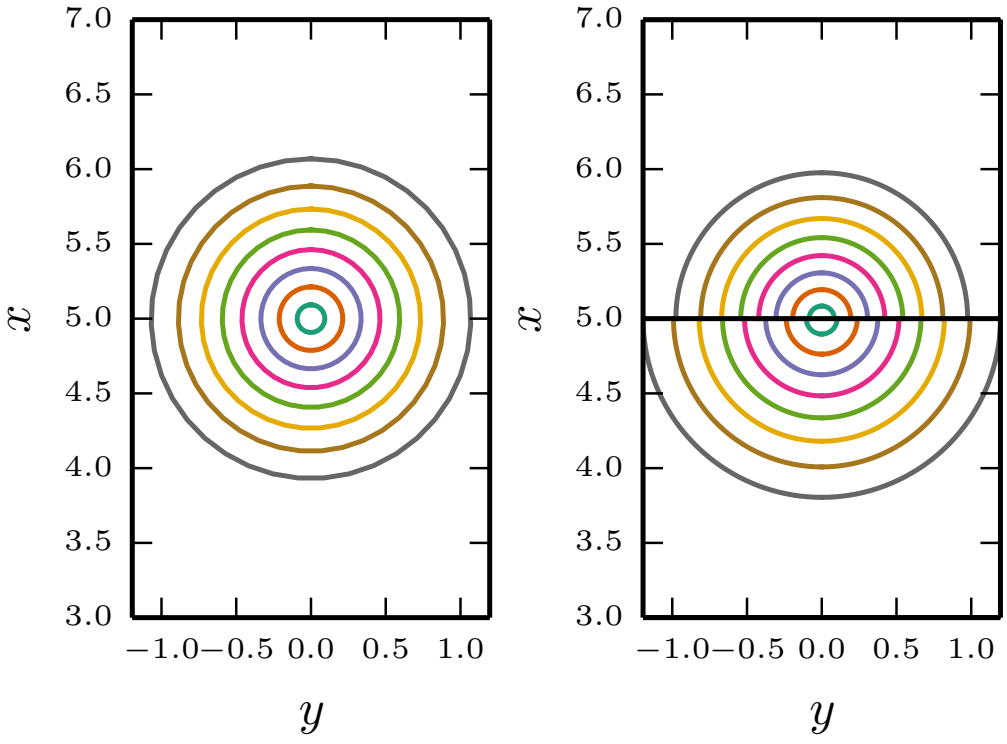


Figure 27: Examples of available gyro-trajectories at gyro-center position $(x, y) = (5, 0)$ for a regular grid (left plot) and a two-block grid (right plot) with the block boundary at $x = 5$.

electromagnetic fields, which are given in the position space and do not depend on μ. Therefore, one can compute gyro-averaged fields at any magnetic moment coordinate, irrespective of the magnetic moment nodes chosen for the discretization of the distribution function.

This issue can, however, not be avoided for the gyrophase averaging (4.6), which is applied on the distribution function. Fortunately, this type of gyrophase averaging is used when computing moments (2.34), which also involves the integration over the

magnetic moment. Therefore, we do not need the standalone gyrophase averaging of the distribution functions and can combine this operation with the μ-quadrature.

For regular grids, the combined operation can be expressed in a schematic way

$$\int \langle f \rangle (\mathbf{x}) \, d\mu = \sum_{m=1}^{nw0} w_m \mathcal{G}^{\dagger}(\mu_m) \mathbf{f}(\mu_m) \,, \tag{4.22}$$

where $\mathbf{f} = (f_1, \ldots, f_{nx0})^T$. This expression is valid as long as all elements of the vector $\mathbf{f}(\mu_m)$ are computed at the same magnetic moment μ_m .

For block-structured grids, however, fixing the index m does not lead to the same μ_m at all radial positions; thus, (4.22) does not hold anymore. For the sake of simplicity, we illustrate how we obtain a correct mathematical expression of the combined operation for the block-structured grids for a two-block grid. In this case, the vector $\mathbf{f}$ is given by

$$\mathbf{f}_m = \left(\begin{array}{ccc|ccc} f_1^{(1)} & \cdots & f_k^{(1)} & f_{k+1}^{(2)} & \cdots & f_{nx0}^{(2)} \end{array} \right)^T , \tag{4.23}$$

where the superscript index refers to the corresponding block.

The two parts of vector $\mathbf{f}_m^{(1,2)}$ are computed for two different values of the magnetic moment $\mu_m^{(1,2)}$. Therefore, we can compute the combined operation by subdividing the columns of the gyrophase matrix $\mathcal{G}^{\dagger}$ into two submatrices $\mathcal{G}^{\dagger(1,2)}$, which are computed for the corresponding $\mu_m^{(1,2)}$.

Then for the two-block grid, the operation (4.22) is given by

$$\int \langle f \rangle (\mathbf{x}) \, d\mu = \sum_{m=1}^{nw0} \left(\begin{array}{c|c} w_m^{(1)} \cdot \mathcal{G}^{\dagger(1)}_{*,1\ldots k}(\mu_m^{(1)}) & w_m^{(2)} \cdot \mathcal{G}^{\dagger(2)}_{*,k+1\ldots nx0}(\mu_m^{(2)}) \end{array} \right) \cdot \left(\begin{array}{ccc|ccc} f_1^{(1)} & \cdots & f_k^{(1)} & f_{k+1}^{(2)} & \cdots & f_{nx0}^{(2)} \end{array} \right)^T . \tag{4.24}$$

In the latter expression, we denote by $\mathcal{G}^{\dagger(b)}_{*,l\ldots k}(\mu^{(b)})$ the submatrix corresponding to the columns of $\mathcal{G}^{\dagger}$ with indices from l to k. This submatrix is computed for the set $\mu^{(b)}$, which belongs to the b-block.

To summarize, in the case of the block-structured grids, we do not compute a standalone gyrophase averaging of the distribution function, but a combined operation of the averaging and the integration over the magnetic moment. To achieve this, for each grid-block, we compute the columns of the gyrophase averaging matrix $\mathcal{G}^{\dagger}$ with a corresponding set of μ nodes and then apply (4.22). In the case of two blocks, this expression is explicitly given by (4.24).

For block-structured grids, the matrix $\mathcal{G}$ for the gyrophase averaging of the fields is subdivided into rows corresponding to different blocks. For example, the expression of the gyrophase averaging of the fields for the two-block grid is given by

$$\overline{\phi}(\mathbf{X}, \mu_m) = \left(\frac{\mathcal{G}^{(1)}_{1\ldots k,*}(\mu_m^{(1)})}{\mathcal{G}^{(2)}_{k+1\ldots nx0,*}(\mu_m^{(2)})} \right) \cdot \boldsymbol{\phi} \,, \tag{4.25}$$

where $\mathcal{G}^{(b)}_{l\ldots k,*}(\mu^{(b)})$ is a submatrix corresponding to the rows of $\mathcal{G}$ with indices from l to k and computed with the set $\mu^{(b)}$ of the b-block.

Considering the relation $\mathcal{G}^{(b)}_{l\ldots k,*}(\mu^{(b)}) = \mathcal{G}^{\dagger(b)}_{*,l\ldots k}(\mu^{(b)})$, we do not have to compute the $\mathcal{G}$ and $\mathcal{G}^{\dagger}$ matrices separately for the block-structured grids. As in the case of the regular grid, $\mathcal{G}^{\dagger}$ is the complex conjugate transpose of $\mathcal{G}$.

4.4 QUALITATIVE COMPARISON OF BLOCK-STRUCTURED GRIDS WITH THEIR REGULAR COUNTERPARTS

In GENE and many other grid-based gyrokinetic simulations, the execution time of one time step is roughly proportional to the number of grid points in the velocity space. Therefore, the reduction in the number of computational nodes gives approximately an estimate of the speedup. Other enhancements associated with the reduced number of the grid points are an improved parallel efficiency (less ghost grid points) and oftentimes a larger time step size (less time iterations are necessary).

In Table 2, we provide estimates of the number of grid points for three types of grids: regular, "BS 1" — block-structured of the first type, and "BS 2" — block-structured of the second type.

Table 2: Number of computational nodes in three types of grids constructed for the TCV temperature radial profile. The block-structured grid of the first type is denoted by "BS 1" and of the second type by "BS 2".

space type	grid type		
	regular	BS 1 6 blocks	BS 2 6 blocks
radial direction `nx0`	512	223, 33, 38, 56, 87, 75	223, 33, 38, 56, 87, 75
binormal direction `nky0`	1	1	1
parallel direction `nz0`	16	16	16
parallel velocity `nv0`	92	92, 76, 60, 48, 40, 34	34
magnetic moment `nw0`	64	64, 58, 48, 38, 33, 28	7
$x - v_{\parallel}$ subgrid	47104	34022 (72%)	17408 (37%)
$x - \mu$ subgrid	32768	25109 (77%)	3584 (10%)
$x - v_{\parallel} - \mu$ subgrid	3014656	1856312 (62%)	121856 (4%)
full grid	48234496	29700992 (62%)	1949696 (4%)

For the sake of consistency, we use the TCV electron temperature profile shown in Figure 9 to construct the block-structured grids with six blocks (this number of blocks is optimal according to Figure 21). To estimate which number of computational nodes in the resulting block-structured grids yields results with the same accuracy as the provided regular grid, we assumed that the regular grid is chosen according to the procedure described in Subsection 3.3.2, i.e., the regular grid has the widest range and the finest resolution required in the velocity directions.

The provided example of the block-structured grid of the first type is obtained by cutting away the areas of the regular grid, which are outside the step-like approximation of the desired domain contour, see Figure 18. The resulting step-structure is reflected in the number of radial grid points `nx0` in each block, which is shown in the first row of Table 2 for the columns of the block-structured grids. We used the same step-like approximation for both of the block-structured grids.

From the table, we observe that, in this particular example, the remainder of the grid points after the procedure of removing some parts of the simulation domain is 72% in

the $x - v_{\parallel}$ subspace, 77% in the $x - \mu$ subspace; the resulting full block-structured grid is 62% (1.6 times less) of the regular counterpart in terms of computational nodes. At a first glance, the reduction in the number of grid points in the $x - \mu$ subspace seems too small, considering that we reduce the simulation area in this subspace approximately by half, as shown in Figure 22. The reason lies in the fact that most of the nodes from the Gauss-Laguerre quadrature are located close to the axis $\mu = 0$. Therefore, removing a big area with high μ values does not bring a significant reduction in the number of grid points.

The block-structured grid of the second type yields significantly less grid points than of the first type. In the example provided in Table 2, only 4% (25 times less) grid points of the original regular grid are kept. The theoretical estimates are computed by assuming that the velocity space in the regular grid is resolved with the finest resolution, which is necessary for the lowest temperature regions (high x values). Therefore, to find the number grid points in the velocity space for the block-structured grid of the second type, we apply the following formula

$$\mathtt{nv0}_{BS_2} = \mathtt{nv0}\frac{\mathtt{lv}_{min}}{\mathtt{lv}_{max}}, \tag{4.26}$$

$$\mathtt{nw0}_{BS_2} = \mathtt{nw0}\frac{\mathtt{lw}_{min}}{\mathtt{lw}_{max}}, \tag{4.27}$$

where $\mathtt{lv}_{min}$ and $\mathtt{lw}_{min}$ are the minimum $v_{\parallel}$ and μ ranges of the grid-block corresponding to the area with the lowest temperature, $\mathtt{lv}_{max}$ and $\mathtt{lw}_{max}$ the velocity space ranges of the regular grid or the grid-block corresponding to the area with the highest temperature. By applying the presented formulas, we ensure that the velocity space resolution in the grid-block containing areas of the lowest temperature is preserved (the same as in the regular grid) and the resolutions of all other blocks are coarsened.

The introduced formulas for $\mathtt{nv0}_{BS_2}$ and $\mathtt{nw0}_{BS_2}$ can be written in terms of temperature ratios by taking into account that $\mathtt{lv}$ is proportional to the standard deviation σ and $\mathtt{lw}$ is inversely proportional to the rate parameter λ of the corresponding projections of the background distribution function, for details see Subsection 3.3.2. Then, from (3.42,3.47) follows

$$\mathtt{nv0}_{BS_2} = \mathtt{nv0}\frac{\sigma_{min}}{\sigma_{max}} = \mathtt{nv0}\sqrt{\frac{T_{min}}{T_{max}}}, \tag{4.28}$$

$$\mathtt{nw0}_{BS_2} = \mathtt{nw0}\frac{\lambda_{max}}{\lambda_{min}} = \mathtt{nw0}\frac{T_{min}}{T_{max}}, \tag{4.29}$$

where σ_{min} and λ_{min} are the standard deviation and the rate parameter used to choose the range and the resolution of the velocity grids in the grid-block corresponding to the area with the lowest temperature, whereas σ_{max} and λ_{max} correspond to the highest temperature. The total reduction in the number of grid points is given by $(T_{max}/T_{min})^{3/2}$. In the example of $T(x)$ shown in Figure 9, the ratio of the highest and lowest temperature is approximately ten and, thus, the achievable theoretical reduction in the number of grid points is about 32.

In practice, the difference in the number of computational nodes in the regular grid and block-structured grid of the second type is not as high as the theoretical estimate. For example, the nonlinear tests (see Section 6.2) with the TCV electron temperature and density profiles demonstrate that the reduction in the number of grid points is

around ten instead of the estimated 25. The main reason is that the fluctuating part of the distribution function does not retain the shape of the background exactly and is not equally strong at all radial distances. For example, if the fluctuating part is not excited at the radial distances with low temperature, then the regular grid can be coarsened without a loss of accuracy. However, this does not hold for the block-structured grid of the second type, because the resolution of the velocity grid is more or less already adjusted to the radial temperature profiles to match the velocity space scales of the possibly excited fluctuating distribution function.

4.5 SUMMARY

In this chapter, we presented block-structured grids of two types, which addressed the problem of strong radial temperature variation and presented an alternative approach to the thermal speed normalization. A significant advantage of the introduced grids in comparison to the normalized grid is that they are minimally invasive and do not cause modifications in the gyrokinetic set of equations.

The first type of block-structured grids approximates the shape of the desired domain by a step-like contour and keeps the resolution of the velocity space the same in each block. This type of grids requires minimum modifications to the implementation of the original regular grids. However, these grids do not usually yield a significant reduction in the number of grid points. In the demonstrated example, the reduction in the number of computational nodes for the first type of block-structured grids was estimated to 38% fewer as for the original grid.

Compared to the first type, the second type of block-structured grids yields a much higher reduction in the number of grid points. This is achieved by reducing the area of the simulation domain and adjusting the resolution of the grid in each block. The most consistent choice for the resolution is obtained by fixing the number of velocity grid points in each block. This approach leads to block-structured grids that closely match the normalized grids (see Figure 20) regarding the simulation domain contour and the grid resolution.

When constructing these block-structured grids, the focus differs depending on the subspace: $x - v_{\parallel}$ or $x - \mu$. In the magnetic moment direction, only quadrature-like operations, such as a gyrophase averaging, are performed and the main goal is an optimal choice of the quadrature rule in each block. In the parallel velocity direction, a finite difference scheme is employed to compute derivatives and the main focus is to set a correct mesh size in each block matching the scale-lengths of the fluctuating function.

The block-structured grids of the second type are more challenging to realize than the first type, because of the misalignment of the grid nodes close to the boundaries of the grid-blocks. However, they are also more promising, due to the higher reduction in the number of the computational nodes. In the provided example, the theoretical estimate of the reduction in the number of grid points in the second type of block-structured grids was around 96% fewer grid points compared to the regular grid.

In the next chapter, we detail the implementation of the block structured grids and explain aspects such as the treatment of block boundaries and parallelization.

5 IMPLEMENTATION ASPECTS

GENE is a highly parallel code with a large and growing user and developer base from different institutions, for more information see [37]. In this context, it is especially important to elaborate a seamless integration of the block-structured grids into the existing GENE code. The idea of the block-structured grids, as it is presented in Chapter 4, stems from using the blocks of regular grids with different ranges and resolutions in the velocity space. In this way, the grid is adjusted to the possible velocity-scales, which depend on the radial temperature profiles of the fluctuating part of the distribution function. This means that all the code written originally for the regular grids can be reused for computations inside the grid-blocks. Otherwise, block-structured grids necessitate only minor modifications, which can be roughly classified into three categories: prefactors, boundaries of the grid-blocks, and parallelization.

The simplest type of modifications is given by prefactors, which correspond to different coefficients in the set of gyrokinetic equations. These coefficients are computed during the initialization phase of the simulation and depend on the nodes and weights of the Gaussian quadrature rule (μ_m and w_m) in the magnetic moment direction and on the parallel velocity mesh spacing $dv_\parallel$. In the block-structured grids of the first type, the quadrature rule and the mesh spacing are the same at all radial positions as in the original regular grids. Therefore, this type of block-structured grids does not require modifications of prefactors. In the block-structured grids of the second type, however, these quantities (μ_m, w_m, and $dv_\parallel$) differ in each grid-block. Therefore, the prefactors have to be adjusted correspondingly.

Adjusting the prefactors is a straightforward procedure: if a prefactor is an array with one of the dimensions associated with the radial distance, then only computations have to be changed, to involve additional radial dependencies. When no dimension is associated with the radial distance, e. g., the prefactor is a scalar, the prefactor has to be extended by introducing another dimension enumerating grid-blocks. Then, the original computations with different μ_m, w_m, and $dv_\parallel$ have to be carried out for each block separately. After the block-structured grid is initialized, we have to ensure that all prefactors with an additionally introduced grid-block dependence are applied correctly, i. e., during computations in a particular block, consistent prefactor values are used.

In this chapter, we offer insight into implementation aspects of the block-structured grids only. Details on more complex modifications of prefactors, i. e., computing matrices for gyrophase averaging in the case of block-structured grids, are provided in Section 5.1. The other two modification categories (boundaries of the grid-blocks and parallelization) require more attention than the prefactors and are discussed in Section 5.2 and Section 5.3, respectively. To obtain a full picture of the manner in which GENE is implemented, please refer to [32, 43, 44, 96, 103–106], as well as the code manual [130].

5.1 IMPLEMENTATION OF GYROPHASE AVERAGING

In Section 4.3, we introduced the gyro-averaging matrices $\mathcal{G}$ and $\mathcal{G}^\dagger$ for the block-structured grids. Formally, the changes in the computations of the gyro-matrices belong to the category of modifications in prefactors. In this section, we discuss the details on how to set these matrices correctly for the particular case of a block-structured grid of the second type with three blocks.

The gyro-averaging matrix $\mathcal{G}$, which is applied on the electromagnetic field potentials, is subdivided in a row-wise manner (according to (4.25)). Each resulting group of rows is computed with a different $\mu_m^{(b)}$ set associated to a b-indexed grid-block. For the block-structured grid instance with three blocks, the gyro-averaging matrix $\mathcal{G}$ has the following structure

$$\bar{\phi}(\mathbf{X}, \mu_m) = \left(\bar{\phi}(\mathbf{X}, \mu_m^{(1)}) \,\middle|\, \bar{\phi}(\mathbf{X}, \mu_m^{(2)}) \,\middle|\, \bar{\phi}(\mathbf{X}, \mu_m^{(3)}) \right)^\intercal = \mathcal{G}(\mu_m)\phi(\mathbf{x})$$

$$= \begin{bmatrix} * & * & * & * & & & & & & & \\ * & * & * & * & * & & & & & & \\ * & * & * & * & * & * & & & & & \\ * & * & * & * & * & * & * & & & & \\ & * & * & * & * & * & * & * & & & \\ & & * & * & * & * & * & * & * & & \\ & & & * & * & * & * & * & * & * & \\ & & & & * & * & * & * & * & * & * \\ & & & & & * & * & * & * & * & * \\ & & & & & & * & * & * & * & * \\ & & & & & & & * & * & * & * \end{bmatrix} \cdot \begin{bmatrix} * \\ * \\ * \\ * \\ * \\ * \\ * \\ * \\ * \\ * \\ * \end{bmatrix} = \begin{bmatrix} * \\ * \\ * \\ * \\ * \\ * \\ * \\ * \\ * \\ * \\ * \end{bmatrix} . \tag{5.1}$$

In this equation, we used colored frames to enclose matrix and vector elements belonging to the same grid-block (computed with the same set of the magnetic moment nodes).

The illustrated gyrophase averaging matrix is just a simplified representation of the matrices used in practice. In real life simulations, gyro-averaging matrices have significantly more elements and their bandwidth is often much narrower relative to the matrix size. This makes them difficult to visualize. The same holds for all similar expressions in the following discussion.

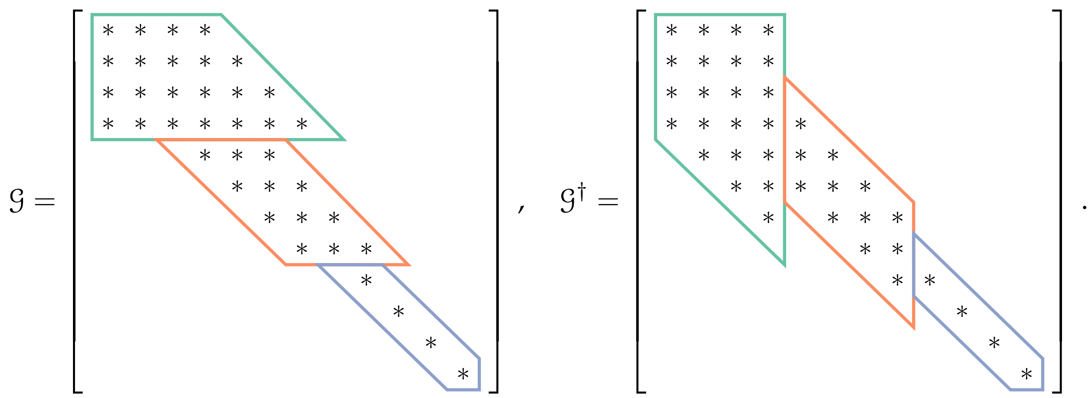
The gyro-averaging matrix $\mathcal{G}^\dagger$ is subdivided in a column-wise manner, according to Expression (4.24). This matrix is multiplied by vectors of the distribution function or already gyro-averaged fields. The schematic representation of the respective matrix-vector product for the same three grid-block example is given by

$$\langle f \rangle (\mathbf{x}) = \mathcal{G}^\dagger f(\mathbf{X}) = \begin{bmatrix} * & * & * & * & & & & & & & \\ * & * & * & * & * & & & & & & \\ * & * & * & * & * & * & & & & & \\ & * & * & * & * & * & * & & & & \\ & & * & * & * & * & * & * & & & \\ & & & * & * & * & * & * & * & & \\ & & & & * & * & * & * & * & * & \\ & & & & & * & * & * & * & * & * \\ & & & & & & * & * & * & * & * \\ & & & & & & & * & * & * & * \end{bmatrix} \cdot \begin{bmatrix} * \\ * \\ * \\ * \\ * \\ * \\ * \\ * \\ * \\ * \\ * \end{bmatrix} = \begin{bmatrix} * \\ * \\ * \\ * \\ * \\ * \\ * \\ * \\ * \\ * \\ * \end{bmatrix} . \tag{5.2}$$

As it was mentioned in Section 4.3 (Expression (4.21)) and also shown in Expressions (5.1) and (5.2), the gyro-averaging matrices have a band structure. Only non-zero elements inside the diagonal band are saved in the data structures developed specially for GENE. For more details on the implementation of the band matrices, see to the developer's manual [131].

In the default implementation of the second type block-structured grid, we first compute the matrices $\mathcal{G}$ and $\mathcal{G}^{\dagger}$, which are saved in the same band matrix data structures as the gyro-averaging matrices for the regular grids. Then, we reuse the implementation of the original regular grids to carry out computations involving the gyro-averaging matrices. While this approach is minimally invasive, it does not reduce the size (bandwidth) of the resulting gyro-averaging matrices and, thus, does not lead to additional memory savings. Consequently, there are also no additional savings in the number of floating point operations in the matrix-vector product. The savings come only from the fact that we need less μ grid nodes for the block-structured grids of the second type and, therefore, we have to compute less matrix-vector products for the gyrophase averaging operation.

The bandwidth of the gyro-averaging matrices is determined by the gyro-radius to grid-space ratio and is, thus, strongly influenced by the maximum magnetic moment value in the set of the Gauss-Laguerre quadrature nodes μ_{max}. This leads to relatively larger μ_m values and, consequently, larger gyro-radii. In the block-structured grids of the second type, the maximum magnetic moment $\mu_{max}^{(b)}$ might differ considerably in the smallest and biggest grid-blocks. The following examples demonstrate the structures of the resulting gyro-averaging matrices for the three grid-blocks



(5.3)

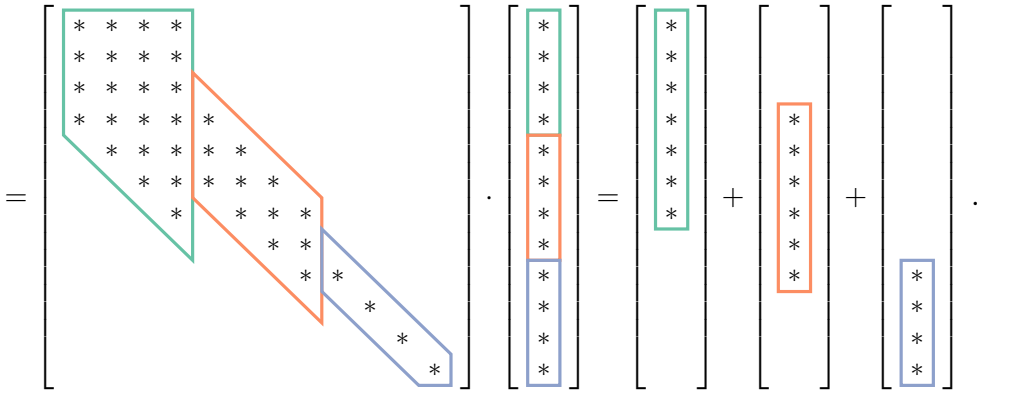
From these matrix schemes, we observe that the initial gyrophase averaging matrices are segmented into parts with different bandwidths, where each part corresponds to a separate grid-block. Considering both the size `nx0` $\times$ `nx0` of the square matrices and that `nky0` $\times$ `nz0` $\times$ `nw0` $\times$ `n_spec` gyro-averaging matrices[1] are allocated during GENE simulations, significant memory savings can be obtained by keeping only non-zero matrix elements of the gyro-averaging matrices for the block-structured grids of the second type. For this purpose, however, we have to replace the original band matrix format, because the amount of memory allocated in this data structure is determined by the maximum bandwidth in a whole gyro-averaging matrix.

1 `n_spec` is the number of species, all other notations are explained in Table 2

Therefore, in addition to the aforementioned default approach, we implemented an alternative data structure to store the gyro-averaging matrices for the block-structured grids of the second type. In this data structure, every matrix part with a different bandwidth is saved in a separate array according to a widely used (e. g., in LAPACK) band storage format: the band diagonals of the original matrix are saved in array rows, so that the elements in one column of the array correspond to the elements in the same column of the matrix, for details see [132]. This format is supported by many numerical linear algebra libraries, which allow us to perform efficiently matrix-vector products for each submatrix of the gyro-averaging matrices.

With the band storage format, the matrix-vector product demonstrated in (5.1) is replaced by three corresponding products for each submatrix. The resulting subvectors are merged together to form the whole vector of the gyro-averaged field potential. The matrix-vector product provided in (5.2) becomes more complicated and is performed as in the following example

$$\langle f \rangle (\mathbf{x}) = \mathcal{G}^{\dagger} f(\mathbf{X})$$



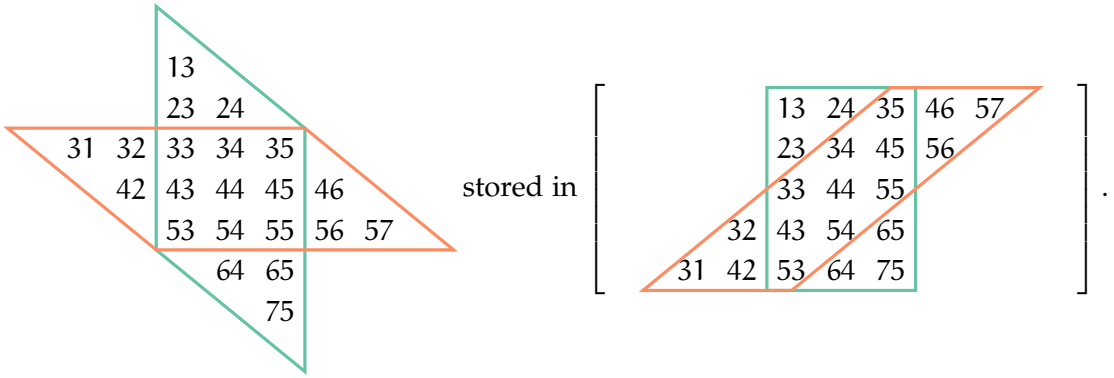
(5.4)

where each submatrix is multiplied by a corresponding subvector and the partial results with preserved global indices (of the whole original vector) are summed up to form the final gyro-averaged vector.

The aforementioned implementations of the gyrophase averaging are provided for the case when the gyro-averaging matrix (4.6) for the distribution function and already gyro-averaged fields is approximated by $\mathcal{G}^{\dagger}$. In some cases (e. g., when the Neumann boundary conditions are used in radial direction), however, this gyro-phase matrix is taken to be equal to $\mathcal{G}$ for the original regular grids in GENE. For the block-structured grids of the second type, this leads to a different matrix in the gyrophase averaging, as introduced in (4.6). The corresponding gyro-averaging matrix has the same structure as $\mathcal{G}^{\dagger}$ in expression (5.3), but with all elements taken from $\mathcal{G}$. In other words, we have to subdivide the gyro-averaging matrix $\mathcal{G}$ and compute elements with different $\mu_m^{(b)}$ sets not only row-wisely, but also column-wisely.

At a first glance, we have to save two types of matrices and allocate twice as much memory. In fact, we actually manage to save the row-wise and column-wise versions

of each submatrix efficiently in a single array, and preserve the standard band storage format. The idea is illustrated by the following expression



(5.5)

For the sake of visibility, the number of elements in this example is very small and the bandwidth of the submatrix is relatively wide.

With the introduced combined storage we can access required submatrices (row-wise or column-wise) and, thus, still use the standard band matrix-vector product routines. For example, to access the row-wise submatrix (orange frame), we take the whole storage array and specify the dimension of the row-wise submatrix (three rows, seven columns, one superdiagonal, and one subdiagonal). To access the column-wise submatrix (green frame) we first take the corresponding subarray of the storage array (green frame) by specifying the memory address of the first subarray element denoted by 13. All other elements (in the case of column-wise storage) are stored sequentially. Then, we provide the dimensions of the column-wise submatrix (seven rows, three columns, one superdiagonal, and one subdiagonal).

5.2 TREATMENT OF BLOCK BOUNDARIES

We start this section with a discussion about the boundary treatment for the block-structured grids of the first type, because they require less modifications than the block-structured grids of the second type do. This type of grids is very similar to the original regular grid. The procedure is thus trivial and does not go beyond how to correctly set the Dirichlet boundary condition for the distribution function to zero on the domain step-like contour. An example of a fragment of a block-structured grid of the fist type in the $x - v_{||}$ subspace is provided in Figure 28. Here, the zero Dirichlet boundary condition is shown by black dashed lines.

The simplest way to develop block-structured grids of the first type is to reuse the data structures (arrays) from the original regular grid code and ignore the parts of the arrays corresponding to the areas outside the blocked contour. An example of the block-shaped domain is demonstrated in Figure 17. The ignored parts of the array are set to zero and are never updated. This way, we ensure the correct Dirichlet zero boundary condition. The ranges of all update-loops inside the simulation code have to be restrained so that only the values at the nodes inside the blocked domain are updated. The presented approach leads to a fast realization of the block-structured grids, but it results in a higher rate of cache misses, due to a non-continuous memory access in the update-loops.

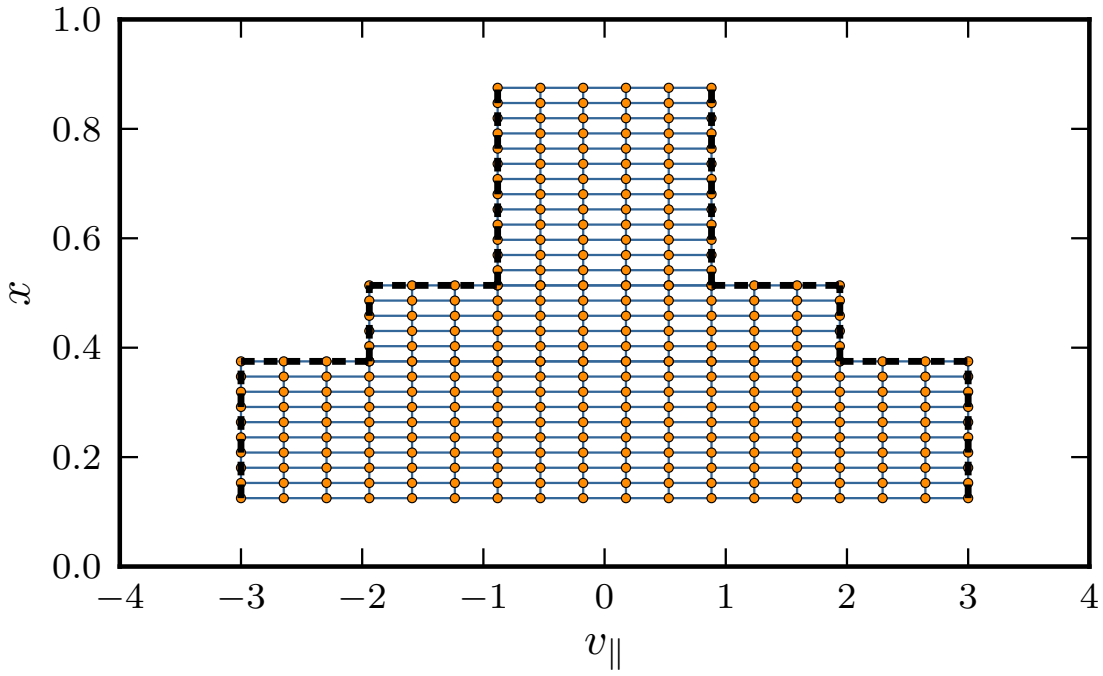


Figure 28: A fragment of a block-structured grid of the first type in the $x - v_{||}$ subspace, with Dirichlet boundary condition set to zero on the contour marked by dashed lines.

The impact of the non-continuous memory access on the run-time of the linear simulation is demonstrated in Figure 29. From this figure, we observe that, despite a total reduction of 62% (green long-dashed line) in the number of grid points for the seven-block grid in the $x - v_{||} - \mu$ subspace, the resulting execution time of one time-step iteration is lowered only to 83% (orange short-dashed line).

In the experiments with the regular grids, the execution time of one time-step iteration corresponds almost ideally to the reduction in the number of computational nodes in a coarsened grid. Thus, we conclude that the gap between the runtime and the total reduction in the number of computational nodes in the block-structured grid of the first type is due to the cache misses (non-continuous memory access). To circumvent this problem of non-continuous memory access, special data structures have to be developed for the block-structured grids. Because the block-structured grids of the first type are just a transitional stage to the block-structured grids of the second type, we only implement this type of data structures for the block-structured grids of the second type. For more details on the resulting speedup versus grid point reduction, see end of Section 6.1 and Section 6.2.

The treatment of the block boundaries in the second type of block-structured grids requires not only setting to zero the Dirichlet boundary condition for the computational nodes at the step-like-shaped boundary, but also changing the finite difference scheme close to the block boundary to address the problem of misaligned coordinate lines. One way to solve the latter problem is by interpolating the distribution function at locations aligned with the adjacent block points, so that derivatives can be easily computed on the boundaries of the neighboring block grid.

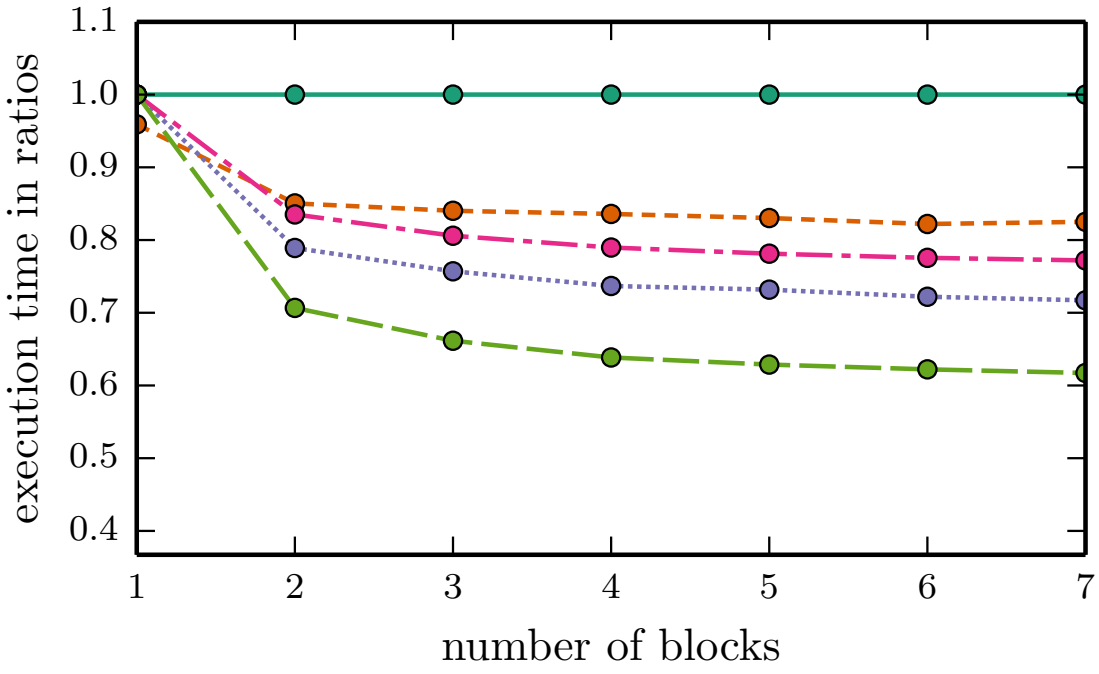


Figure 29: Illustration of how the non-continuous memory access affects the execution time, for one simulation time-step in the case of a block-structured grid of the first type with different number of blocks. The green solid line corresponds to the run-time of the reference regular grid and the orange short-dashed line to the block-structured grid of the first type in the full velocity space. The other lines correspond to the reduction in the number of computational nodes in different subspaces: magenta dash-dotted line — reduction in the $x - v_{\parallel}$ subspace, violet dotted line — reduction in the $x - \mu$ subspace, and green long-dashed line — reduction for the whole space.

An example of the modified scheme for the case of a block-structured grid in the $x - v_{\parallel}$ subspace is provided in Figure 30. This is a schematic representation of nodes used in the default GENE fourth order finite difference scheme for the first order derivative computations at the inner and boundary nodes of a grid-block.

The stencil of the default finite difference scheme for the first order derivative computations in GENE is provided in Table 3, where the "inner" row corresponds to the inner domain nodes and the "boundary 1 or 2" rows to the first and second boundary nodes where the interpolation approach is applied. In this table, the stencil values for the boundary nodes which are multiplied with the interpolated values are marked by gray. Applying the default central difference directly on the interpolated values of the distribution function on the $x - v_{\parallel}$ plane, leads to errors of order $\mathcal{O}(\Delta v_{\parallel}^{m}/\Delta x)$, where Δx and $\Delta v_{\parallel}$ are the mesh sizes in the radial and parallel velocity directions, and m is the order of the applied polynomial interpolation — the inaccuracy of the interpolation is $\mathcal{O}(v_{\parallel}^{m})$. For the block-structured grid in the full velocity space, an additional term $\mathcal{O}(\Delta\mu^{m}/\Delta x)$[2] has to be added to the error of the numerical computations of the derivative at the nodes close to the boundary. The inaccuracy of the finite difference scheme used at the boundary nodes can be considered as an $m-1$ order error. For example, the linear interpolation is of second order; this means that the error is $\mathcal{O}(\Delta v_{\parallel}^{2}/\Delta x)$,

2 $\Delta\mu$ is the representative mesh size at the point of interpolation

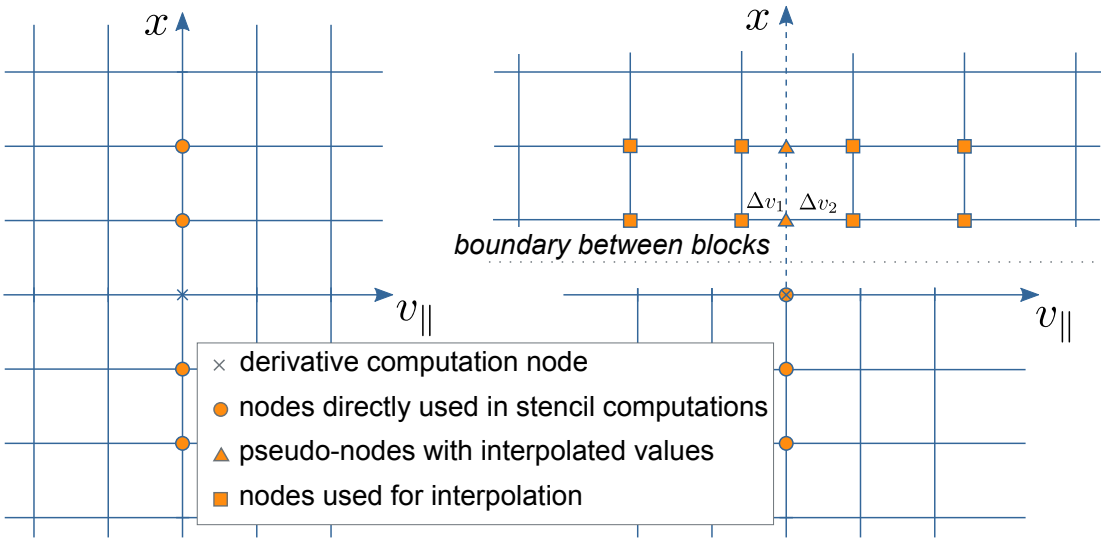


Figure 30: Default first order derivative finite difference stencil nodes in GENE for grid-block inner nodes (left) and block boundary nodes (right), for a block-structured grid of the second type in the $x - v_\parallel$ subspace.

which corresponds to a first order approximation of the radial derivative computed on a boundary of a grid-block. The results provided in Chapter 6 were obtained by using the fourth order $m = 4$ polynomial interpolation scheme for both directions $v_\parallel$ and μ. In Table 4, we provide the interpolation coefficients for the equidistant mesh in the $x - v_\parallel$ plane. For the $x - \mu$ subspace, we use a Lagrangian interpolation technique to obtain the interpolation coefficients.

We tested different order interpolation schemes for the scenarios involving block-structured grids of the second type and observed that the accuracy of the simulation results is essentially not influenced by the order of the interpolation at the block boundaries. Therefore, in the test cases at hand, the error term $\mathcal{O}(\Delta v_\parallel^m/\Delta x)$ due to the interpolation on the block boundaries was insignificant.

While not necessary, the $\mathcal{O}(\Delta v_\parallel^m/\Delta x)$ inaccuracies could be eliminated, for instance, by modifying the main stencil and increasing the number of points used to compute the derivative for the boundary points. For the sake of completeness, we tested two alternative schemes (see Table 3) for each of the two nodes close to the block boundary. The first scheme uses two interpolated values outside the block and has an error of order $\mathcal{O}(\Delta x^4) + \mathcal{O}(\Delta v_\parallel^m)$, while the second scheme needs three interpolated values and is of order $\mathcal{O}(\Delta x^4) + \mathcal{O}(\Delta x \Delta v_\parallel^m)$. In both cases, we used the fourth order $m = 4$ polynomial interpolation scheme provided in Table 4. Unfortunately, both modified finite difference schemes lead to numerical instabilities during the gyrokinetic simulations.

Another potential solution to eliminate these inaccuracies stems from computational fluid dynamics and is described in [133]. However, this method requires running simulations on the additionally introduced ghost grids and causes severe modifications to the original implementation of the regular grids. For this reason, we did not test it with the grid-based gyrokinetic simulations. It is, nevertheless, a possibly interesting direction to be explored in the future.

Table 3: Fourth order finite difference schemes for the first order derivatives.

node type	derivative $f_x(x_i)\Delta x \approx \sum a_k f(x_k)$								accuracy
	a_{i-3}	a_{i-2}	a_{i-1}	a_i	a_{i+1}	a_{i+2}	a_{i+3}	a_{i+4}	
inner		$\frac{1}{12}$	$-\frac{8}{12}$		$\frac{8}{12}$	$-\frac{1}{12}$			$\mathcal{O}(\Delta x^4)$
boundary 1		$\frac{1}{12}$	$-\frac{8}{12}$		$\frac{8}{12}$	$-\frac{1}{12}$			$\mathcal{O}(\Delta x^4) + \mathcal{O}\left(\frac{\Delta v_\parallel^m}{\Delta x}\right)$
boundary 2		$\frac{1}{12}$	$-\frac{8}{12}$		$\frac{8}{12}$	$-\frac{1}{12}$			
alternative schemes for block boundaries									
first alternative scheme									
boundary 1		$\frac{5}{48}$	$-\frac{37}{48}$	$\frac{5}{24}$	$\frac{11}{24}$	$\frac{1}{48}$	$-\frac{1}{48}$		$\mathcal{O}(\Delta x^4) + \mathcal{O}(\Delta v_\parallel^4)$
boundary 2	$-\frac{7}{48}$	$\frac{13}{16}$	$-\frac{17}{8}$	$\frac{35}{24}$	$-\frac{1}{16}$	$\frac{1}{16}$			
second alternative scheme									
boundary 1		$\frac{1}{8}$	$-\frac{31}{36}$	$\frac{25}{72}$	$\frac{7}{18}$	$-\frac{1}{72}$	$\frac{1}{36}$	$-\frac{1}{72}$	$\mathcal{O}(\Delta x^4) + \mathcal{O}(\Delta x \Delta v_\parallel^4)$
boundary 2	$-\frac{5}{24}$	$\frac{13}{12}$	$-\frac{61}{24}$	$\frac{5}{3}$	$\frac{1}{24}$	$-\frac{1}{12}$	$\frac{1}{24}$		

Table 4: Fourth order polynomial interpolation coefficients for nodes outside the block.

function value $f(v_i + \Delta v_1)$ or $f(v_{i+1} - \Delta v_2)\Delta v_\parallel \approx \sum b_k f(v_k)$ $\alpha = \Delta v_1/\Delta v_\parallel$ and $\beta = \Delta v_2/\Delta v_\parallel$			
b_{i-1}	b_i	b_{i+1}	b_{i+2}
$-\frac{1}{6}\alpha\beta(1+\beta)$	$\beta\left(1+\frac{\alpha\beta}{2}\right)$	$\alpha\left(1+\frac{\alpha\beta}{2}\right)$	$-\frac{1}{6}\alpha\beta(1+\alpha)$

When the block-structured grid is applied to the full velocity space, the misalignment is no longer on the line (as has been shown in Figure 30), but on the $v_\parallel - \mu$ plane. An example of two misaligned grids in the $v_\parallel - \mu$ plane from neighboring grid-blocks is illustrated in Figure 31. Like in the case of the misalignment on a line, we address this problem by interpolating values at the missing locations and using the interpolated values in our finite difference scheme. We apply local polynomial interpolation on a plane — the polynomials are defined only in vicinity of the corresponding interpolation point.

In Figure 31, we provide a schematic example of the interpolation procedure: to interpolate at the node surrounded by a circle on the grid with dashed coordinate lines, a fourth order interpolation scheme requires spanning a polynomial through those points enclosed by squares. If this interpolation node is outside the grid with dashed coordinate lines, then we distinguish between two cases

- when the location of the node is greater than μ_{max}, we set the interpolation value to zero

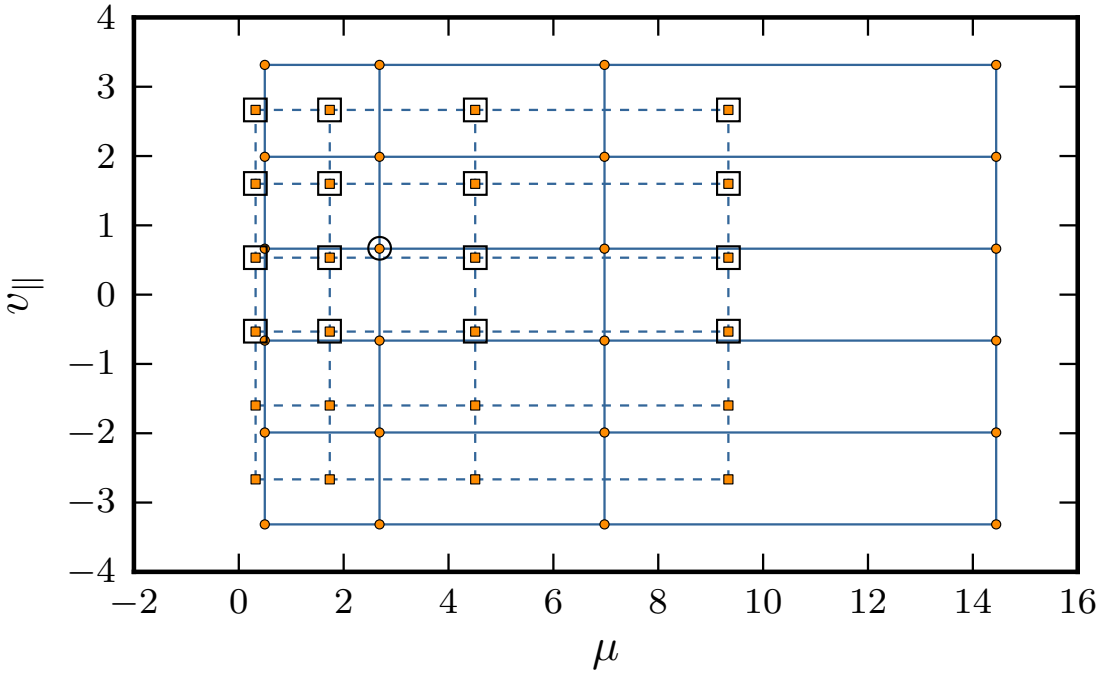


Figure 31: An example of two misaligned grids in the $v_\parallel - \mu$ subspace, which are taken from adjacent grid-blocks. The two grids are marked by either solid or dashed coordinate lines. To interpolate a value at the node surrounded by a circle, a polynomial is spanned through the points enclosed by squares. The resolution of the grid is much finer in real life simulations, making the nodes used for interpolation more localized.

- when the location of the node is less than μ_{min}, we extrapolate the interpolation value at the node

The difference between the minimum magnetic moments μ_{min} of adjacent grid-blocks is very small. Therefore, the extrapolation procedure required in the latter case yields precise results.

The finite difference derivative computations are performed at each time step, along with the interpolation on the block boundaries. Thus, we have to ensure that it is performed as efficiently as possible. For this purpose, we introduce ghost grids in the $x - v_\parallel - \mu$ subspace to keep the interpolated values. The ghost grid is a prolongation of a grid-block to a neighboring block. The extent of the ghost grid in the x direction corresponds to the order of the finite difference scheme used for computing radial derivatives, e. g., a fourth order scheme requires an extent of two grid points. Examples of block-structured grids with and without ghost grids are shown in Figure 32 for the $x - v_\parallel$ subspace and in Figure 33 for the $x - \mu$ subspace.

The values at the nodes of the ghost grid are interpolated as soon as the values at the grid-block overlapping with the ghost grid are updated. No other computations are performed on the nodes of the ghost grid. The number of nodes necessary for the ghost grids is usually about three percent of the total number of grid points. Therefore, the memory consumption is negligibly small in comparison to the total reduction in the number of grid points due to the block-structured grids of the second type. To handle the block-structured grid with the ghost grids, we use a specifically developed

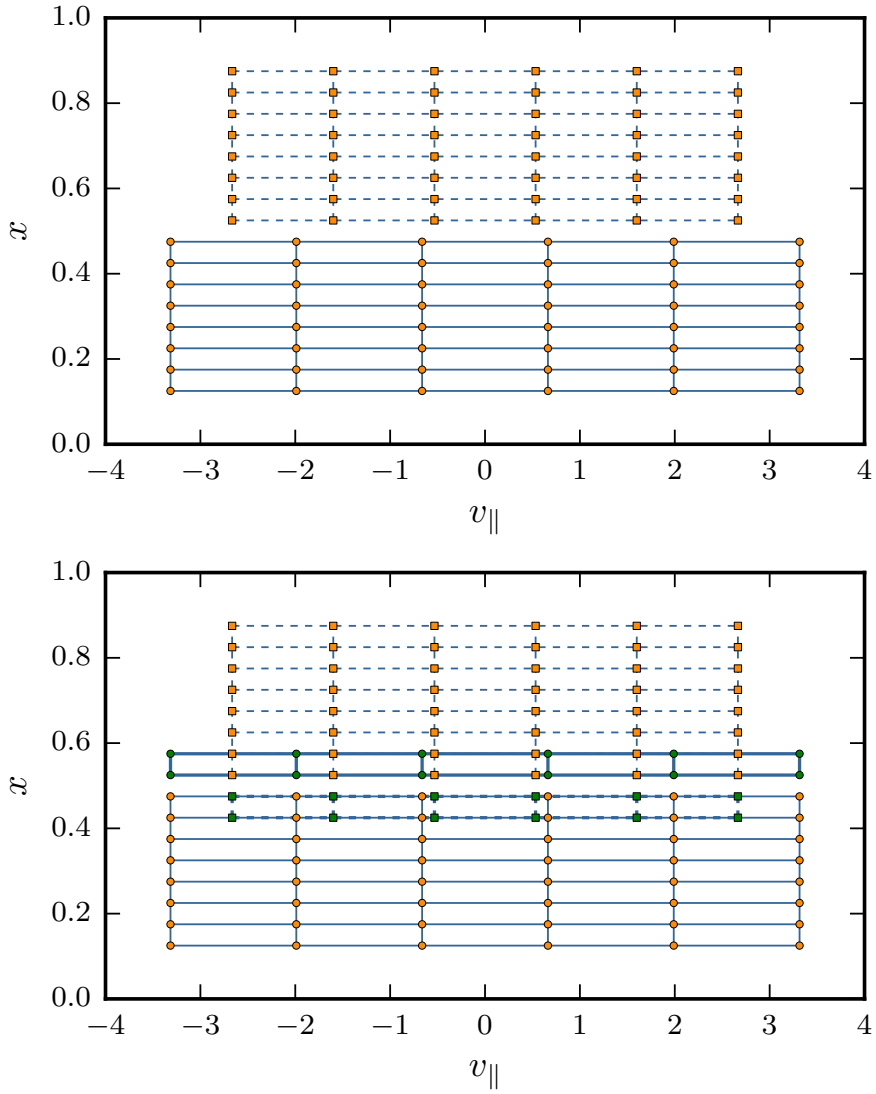


Figure 32: An example of two block-grids in the $x - v_{\parallel}$ subspace without (top) and with (bottom) ghost grids.

data structure. The discussion on this data structure can be found in the Section 5.3 dealing with parallelization.

5.3 PARALLELIZATION ASPECTS

A vast number of the exchange routines originally developed for the regular grids for distributed memory parallelization can be reused for the block-structured grids. A simple example of the domain decomposition in the $x - v_{\parallel}$ subspace for a regular grid is schematically shown in Figure 34. The $x - v_{\parallel}$ subspace is subdivided into process subdomains with approximately the same number of grid points. This is achieved by dividing the radial direction and parallel velocity ranges (`lx` and `lv`) by a specified number of processes in these directions. The same decomposition procedure is applied to all other dimensions.

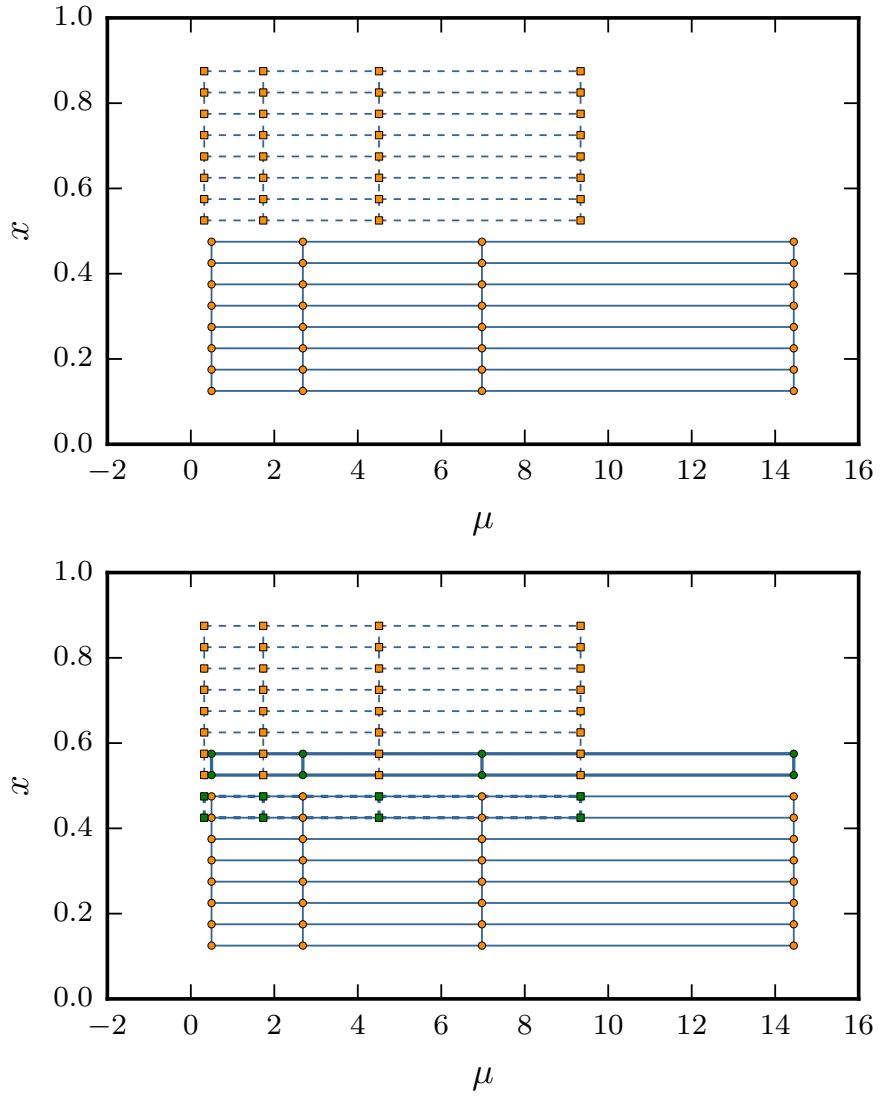


Figure 33: An example of two block-grids in the $x - \mu$ subspace without (top) and with (bottom) ghost grids.

In the case of the block-structured grids of the first type, the subdivision of the radial range (`lx`) is no longer uniform. These grids require different locations for the process grid boundaries in the radial direction in order to balance the number of grid points and, thus, of computational workload in each process subdomain. An example of the domain decomposition for the block-structured grids of the first type in the $x - v_{\parallel}$ subspace is shown in Figure 35.

In this example, only the number of grid points in the $x - v_{\parallel}$ process subplanes is balanced. For block-structured grids in the full velocity space, the number of grid points in the $x - v_{\parallel} - \mu$ subvolumes has to be balanced. The boundaries of the process grid in the radial direction can be found by solving a simple geometrical problem: we fix the boundaries of the process grid in the $v_{\parallel}$ (and μ) directions and write equations for the radial boundary (unknowns), with the condition that all areas (volumes) of the process subdomains are equal. The radial boundaries of the process grid shown in Figure 35

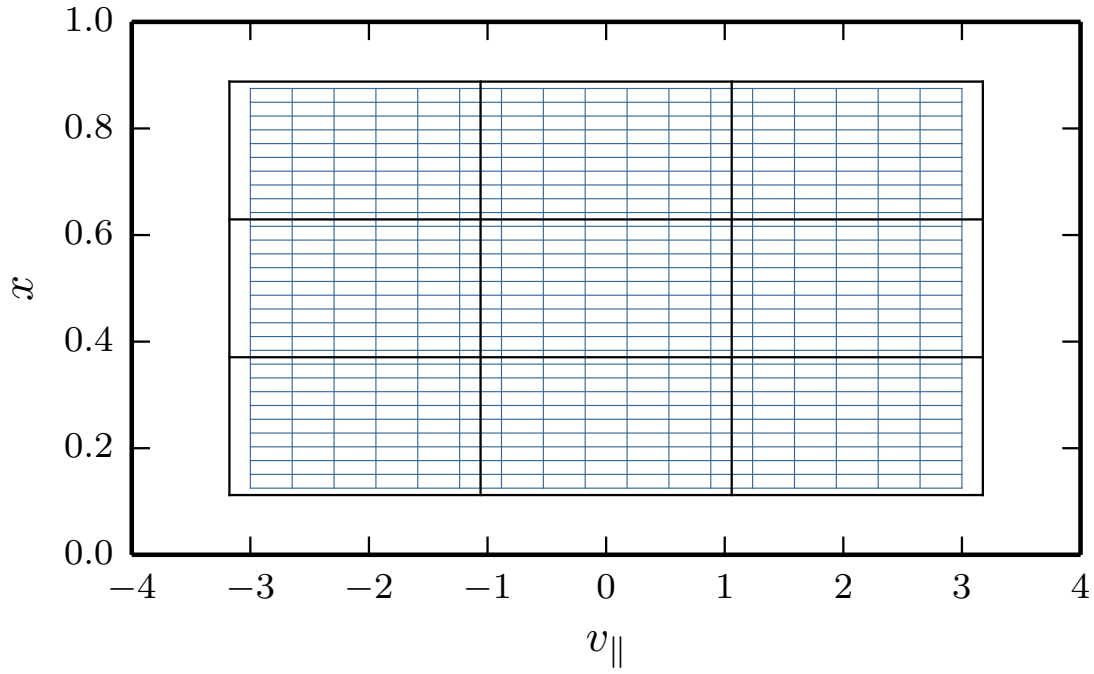


Figure 34: An example of the domain decomposition for a regular grid. Black lines denote the process grid (3×3) in the $x - v_{||}$ subspace, the blue lines correspond to the computational grid.

were determined in this manner. Here, we have `nx0` $= 30$ computational nodes (significantly less than in real life simulations). As a result, we obtain 36 computational nodes in the radially first and second process subdomains and 34 computational nodes in the radially third subdomain. The more radial grid nodes we have in the block-structured grid of the first type, the better the load balancing is.

The readjustment of the process boundary locations in the radial direction is not necessary for the block-structured grids of the second type, because this type of grids has, like the regular grids, the same number of computational nodes in each block. A simple example of the domain decomposition in the $x - v_{||}$ subspace for the block-structured grids of the second type is demonstrated in Figure 36. In a similar way we can obtain the domain decomposition for the $x - \mu$ subspace.

The parallelization schemes in the $v_{||}$ and μ directions, developed originally for the regular grids, remain the same for the block-structured counterparts. Only the communication routines in the radial direction, which are necessary to compute radial derivatives, have to be modified. Examples of the radial exchange for a regular ($x - v_{||}$ subdomain), a block-structured grid of the first type ($x - v_{||}$ subdomain), and a block-structured grid of the second type ($x - v_{||}$ and $x - \mu$ subdomains) are schematically demonstrated in Figure 37.

The implementation of the radial exchange routine in the regular grid is standard: we first send data to the top neighbor and receive from the bottom neighbor, and then send to the bottom and receive from the top. This type of communication is marked by vertical arrows (vertical exchange) in Figure 37. The block-structured grids inherit the vertical exchange from the regular grids. However, there is an additional

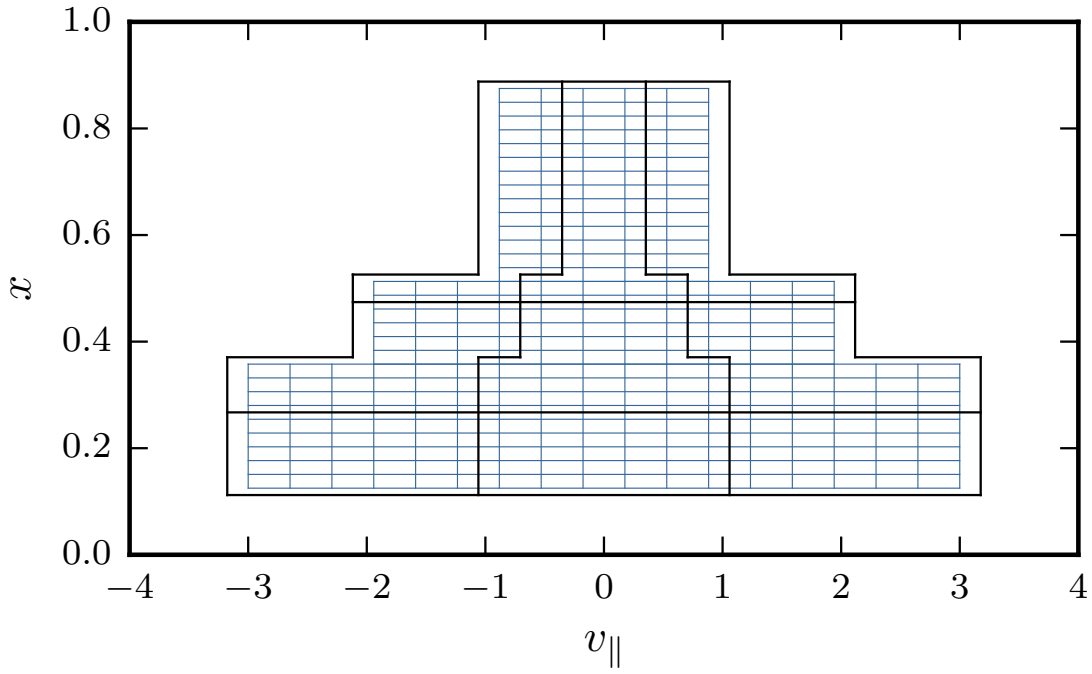


Figure 35: A simplified example of the domain decomposition for the block-structured grid of the first type in the $x - v_{||}$ subspace. The process grid (3×3) is shown by the black lines, the computational grid by the blue lines. The positions of the x process grid boundaries are chosen to balance the number of grid points in each process subdomain.

communication introduced in the block-structured grids, which we denote by slanted arrows (side exchange) in Figure 37. The side exchange becomes necessary due to the step shape of the velocity space boundaries in the process grid. While this type of communication typically requires considerably less data to be sent than the vertical exchange, it is more difficult to implement.

There are several ways of arranging the radial exchange in the block-structured grids. For instance, we can efficiently combine the side exchanges with the standard exchanges in the velocity space directions. The vertical exchange is then performed in the same way as in the original regular grid implementation.

The only exception occurs when the radial boundary of the grid-block coincides with the radial boundary of the process grid. In this case, we have to take care that the interpolated values from the ghost grids are sent during the vertical exchange. The advantage of the aforementioned approach is that the radial exchange in the block-structured grids is, formally, the same as in the regular ones. Furthermore, the side and velocity exchanges can be potentially organized so that the total number of messages being sent is the same as in the regular grids. This approach, however, does not keep the radial exchange decoupled from the communications in the velocity space, nor does it hide the introduced complexity of the radial derivative computations (they can be computed only after the exchange in the velocity space direction is done).

Therefore, we developed a special data structure, which handles the vertical and side exchange in the radial direction and hides the complexity from users. With this approach, we start the computations of the radial derivative and initialize the side ex-

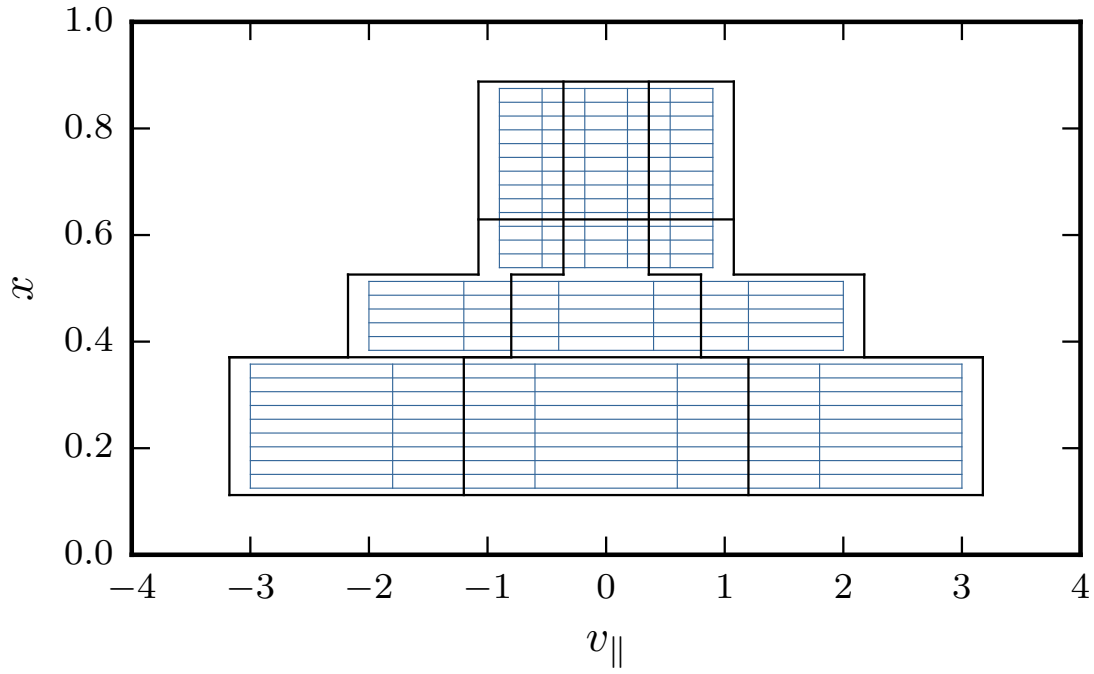


Figure 36: An example of the domain decomposition in the $x - v_{||}$ subspace for the block-structured grids of the second type. The process grid (3×3) is shown by the black lines, the computational grid by the blue ones. In this case, the locations of the radial boundaries of the process grid correspond to the locations of the regular grid.

changes after the vertical exchanges are done. After the side exchanges are completed, the derivatives close to the steps of the process grid are computed.

The values sent during the side exchanges are those from the ghost grids. The processes participating in the communication are not necessarily direct neighbors (top, bottom, right, or left). To provide insight into the communication happening during the side exchange, we show in Figure 38 two overlapping velocity space grids from adjacent grid-blocks, with the radial direction orthogonal to the plane of the plot. When radial derivatives are computed on the x boundary of the smaller grid (marked by dashed lines), the values from the ghost grids on the bigger grid (marked by solid lines) have to be used. The nodes of the ghost grids might be distributed among different processes. For instance, the processes denoted by 1.1, 1.2, 2.1, and 2.2 in Figure 38 keep the parts of the ghost grid (prolonged from the smaller grid-block) that overlap with their local domains. Therefore, if we want to compute the radial derivative at the location marked with a circle, we have to communicate with process 1.1. Similarly, we communicate with process 1.2 for the location marked by a cross, with process 2.1 for the location marked by a square, and with process 2.2 for the location marked by diamond. The values on the nodes of the ghost grid are interpolated as soon as the values on the real computational grids are updated, so that they are available for the radial derivative computations.

At locations close to the boundary of the process grids, a corresponding process might not have all real grid nodes to perform the interpolation of a given order. In such cases, the process computes the interpolation sum only partly, by using the values from

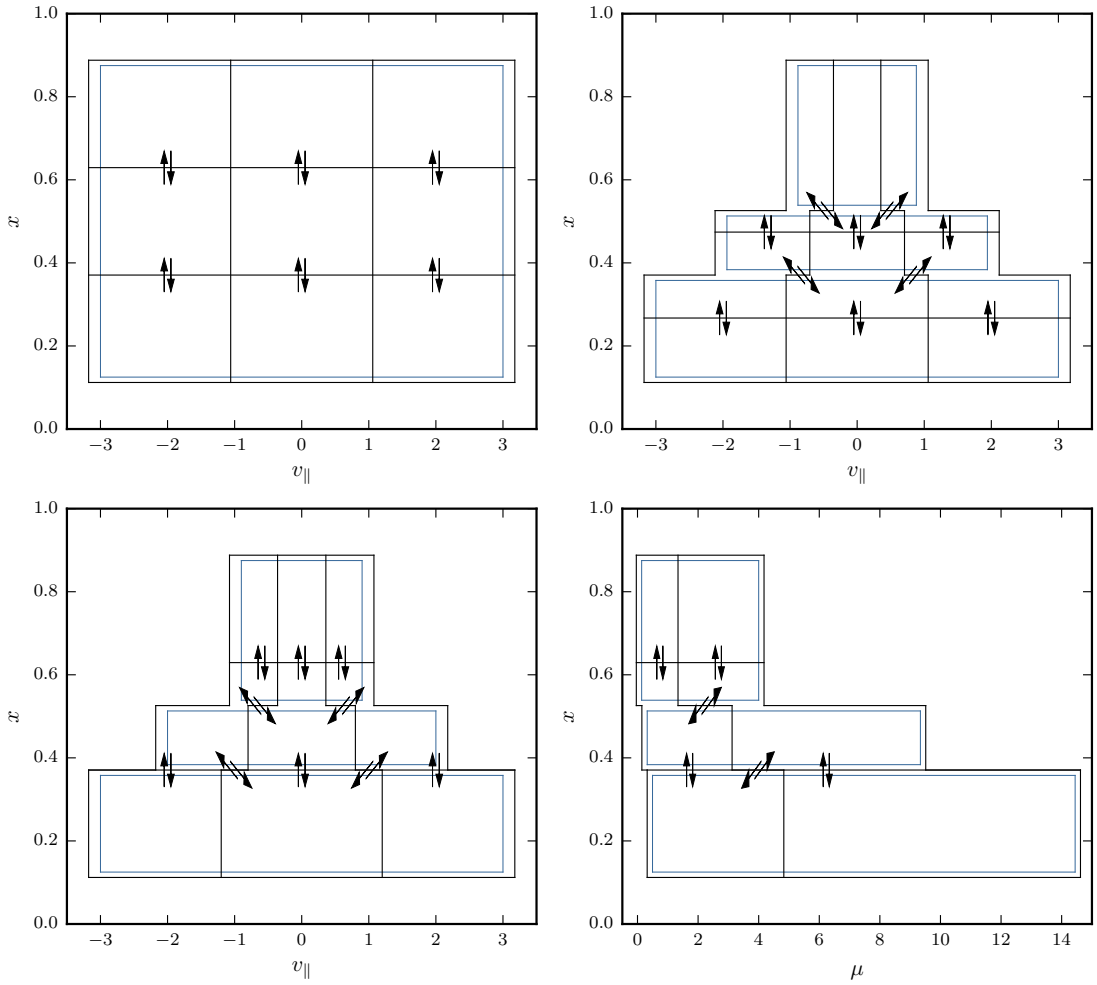


Figure 37: Schematic examples of the radial exchanges in different types of grids: a regular grid (top left), a block-structured grid of the first type (top right), and a block-structured grid of the second type in the $x - v_\parallel$ subspace (bottom left) and in the $x - \mu$ subspace (bottom right). The blue lines denote the outline of the computational grids, the black lines correspond to the process grids. The arrows symbolize the radial exchange.

the available grid nodes. Then, the process computing the radial derivative collects all the partial sums, to sum up and obtain the correct interpolated value. For example, for the computations close to the intersection of the process grid lines, the communication might be necessary with all four processes (1.1, 1.2, 2.1, 2.2).

Both the complex derivative computations and the arising communication are hidden from users in two types of specially developed data structures. The arrangement of the blocks of the computational grids, as well as the process grid on top of it, is reflected in the design of the first data structure type. The facade of the data structure is the "Portion" class, representing grid-related data saved on one process. This is shown schematically using the orange dashed lines in Figure 39. In this figure, we illustrate a fragment of a block-structured grid of the second type in the $x - v_\parallel$ subspace, where grid-blocks are represented by rectangles. In the case of a block-structured grid in the $x - v_\parallel - \mu$ subspace, the grid-blocks are rectangular cuboids, which hinders illustrating the grid arrangement.

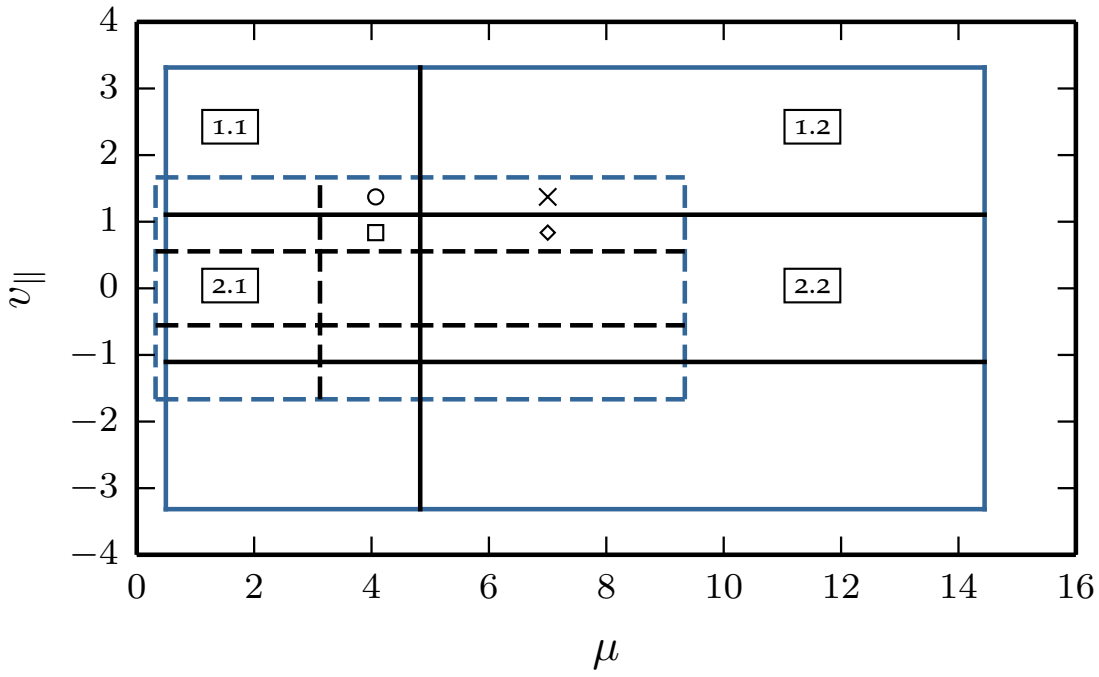


Figure 38: Examples of grid outlines and boundaries of process grids of two velocity space grids from adjacent blocks. The grid outline of the smaller grid is plotted with blue dashed lines and the boundaries of the process grid with black dashed lines. The grid outline and the boundaries of the process grid of the larger grid are plotted with solid lines.

In the provided example, the "Portion" object encompasses three grid-blocks, which are saved in separate data structures — represented by the "Slab" class. A "Slab" object keeps data associated to one block-grid with extended boundaries, which are necessary for saving values received from neighboring processes (for computations of the radial derivatives by a finite difference method). We symbolize this one grid-block related data by the blue rectangles in Figure 39. Furthermore, the "Slab" data structures involve also the ghost grids, which are represented by grid lines inside the blue rectangles in Figure 39. It is the responsibility of the "Slab" objects to interpolate the data on the ghost grids and make the side-exchanges possible.

All data exchanges in the radial direction are arranged by "Portion" objects, which save all information necessary for communication, for example, the order and ranks of the processes communicating with the current process. An example of radial exchanges occurring with one "Portion" object is illustrated in Figure 39 by arrows.

As aforementioned, there are several ways to organize the radial exchange. The default way is to perform first the classic vertical exchange (vertical arrows) and then, gradually, every side exchange (slanted arrows) for each "Slab" object.

5.4 SUMMARY

In this chapter, we explained implementations of three types of modifications (prefactors, block-grid boundaries, and parallelization) induced by introducing block-structu-

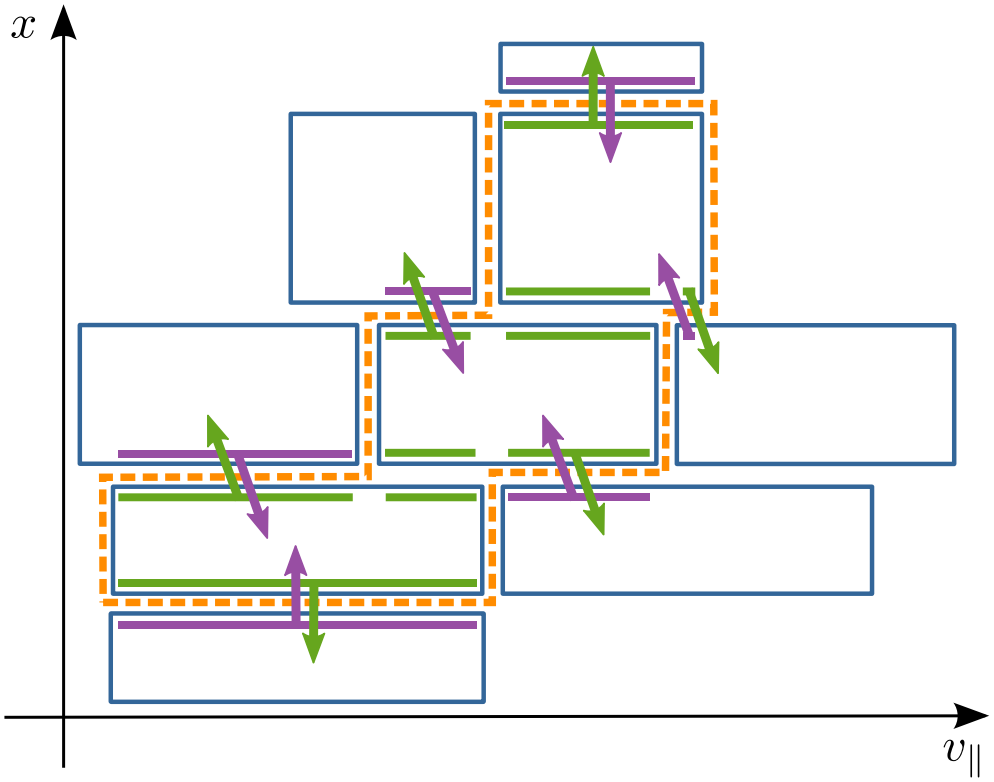


Figure 39: A schematic representation of computational grid data saved and managed by specially developed data structures for the block-structured grids of the second type. The orange dashed lines enclose the part of the grid belonging to one process — the corresponding grid data is managed by a "Portion" class object, which coordinates the radial exchange. The blue rectangles illustrate grid-blocks, while the green lines inside these rectangles represent ghost grids. "Slab" class objects save the data of the grid-blocks with associated ghost grids and perform interpolation routines.

red grids to GENE. It is important to note that these modifications can also be used for other grid-based gyrokinetic codes.

Except for the gyrophase averaging, the changes necessary in the prefactors are straightforward. For the changes in the gyrophase averaging matrices (and respective matrix-vector products), two methods were presented. The default and least invasive method is to reuse the already existing band matrix storage method implemented for the original code with regular grids. This approach requires rewriting only the initialization of the gyrophase averaging matrices. All other routines involved in the averaging remain unchanged. The sizes of the arrays allocated for these matrices are also approximately the same as for regular grid simulations used with the same physical scenario. The number of gyrophase averaging matrices is proportional to the number of grid nodes in the magnetic moment direction and, thus, it is much smaller for the block-structured grids of the second type than for the regular grids.

To capitalize further on the block-structured grids, we also implemented an alternative storage method for the gyrophase averaging matrices. The introduced data structures take into consideration the fact that the bandwidth of each submatrix corresponding to a grid-block depends on the μ for which the matrix is computed. There-

fore, submatrices for small grid-blocks have narrower bandwidths than those for big grid-blocks. We store the non-zero elements of the gyrophase-averaging submatrix in a separate array and have, thus, reduced memory requirements. However, this alternative storage causes changes in the original well-established implementation of the matrix-vector products.

The treatment of the block-boundaries in the block-structured grids of the first type is trivial: the zero amplitude Dirichlet boundary condition is set on the step-shaped contour of the block-structured grid. The original data structures storing grid-associated data can be reused for this type of grids by restraining the ranges of all update-loops. However, this approach increases the rate of cache misses and leads to smaller speedups than expected. This issue can be addressed by enabling a continuous memory access with new data structures. This was realized for the block-structured grids of the second type.

To this purpose, specially developed data structures address not only storage, but also the treatment of boundaries of grid-blocks and parallelization. At a higher level, the data structure stores a portion of grid-related data assigned to one process. On this level, we manage the radial exchanges of two types: vertical exchanges (inherited from the original regular grids) and side exchanges (arising due to step-like boundaries of the process grid). Grid-block data with associated ghost grids is stored in a lower level data structure. This data structure is responsible for interpolation operations at the nodes of the ghost grid, as soon as the values on the real computational grid are updated.

In the following chapter, we provide results of the linear and nonlinear gyrokinetic simulations obtained with the block-structured grids and the implementation outlined above.

6 NUMERICAL RESULTS

Gyrokinetic simulations δf-codes like GENE usually support two main operation modes: linear — including linear terms only — and nonlinear — including nonlinear terms as well. Linear studies are computationally significantly cheaper relative to nonlinear simulations, but are insufficient to investigate turbulence. They offer, nevertheless, a suitable way to explore the underlying microinstabilities. It is relatively easy to compare linear runs stemming from different grids, as the microinstabilities are characterized mainly by two parameters: a linear growth rate and a real frequency.

Nonlinear gyrokinetic simulations are an in-depth way to study plasma turbulence, but they are computationally very expensive (e. g., grids for this type of simulations require much more computational nodes than for linear runs). Furthermore, it is also challenging to verify nonlinear results obtained with different computational grids, because these results are of chaotic nature and, therefore, sensitive to the choice of initial conditions. To obtain a meaningful comparison we investigate time averaged results characteristic to a quasi-stationary state. Moreover, there are many observables characterizing nonlinear runs, which can be compared.

To test our block-structured grids on both types of gyrokinetic simulations, interesting test cases require radial profiles with a considerable temperature variation along the radial direction. Therefore, corresponding simulations necessitate a lot of grid points in the $x - v_{\parallel} - \mu$ subspace. This leads to computationally expensive test scenarios. To solve these issues and make these test cases feasible, we simplify the physical scenarios. To this purpose, for example, we simulate only collisionless and electrostatic plasmas. Furthermore, we choose a circular magnetic geometry with concentric flux surfaces, as described in Section 3.2. This analytic magnetic geometry demands less discretization nodes in the positional space than realistic geometries based on arbitrary magnetohydrodynamics equilibria. The safety factor profile in all our simulations is given by

$$q(r/a) = 2.2(r/a)^2 + 0.868\,, \tag{6.1}$$

where a is the minor radius. The aspect ratio at the last flux surface is set to $a/R = 0.35$ (R is the major radius) and $\rho^* = \rho_{\text{ref}}/a = 1/80$ (ρ_{ref} is the reference gyro-radius). The rest of the physical input depends on a particular test and is provided in the following sections.

This chapter has two sections, each discussing results obtained from linear and nonlinear simulations, respectively. First, as the linear simulations are a prime verification tool for the block-structured grids, we present results of linear runs in Section 6.1. Then, we present results for nonlinear simulations in Section 6.2. This kind of gyrokinetic simulations is the ultimate target of the proposed grids, in view of their enormous computational demands.

6.1 LINEAR SIMULATIONS

As mentioned before, linear microinstabilities are characterized by a linear growth rate (γ) and a real frequency (ω). These observables for the dominant microinstability are obtained either by finding the largest eigenvalue (the real part is γ, the imaginary part is ω) of the linearized right hand side of the gyrokinetic equation or by simulating the initial value problem. These two approaches for investigating the linear microinstabilities are, for instance, compared and discussed in more details in [54].

The gyrokinetic equations solved by the nonlinear initial solver comprise the governing equations of the linear problem and several additional nonlinear terms. Therefore, by developing and verifying the block-structured grids with the linear gyrokinetic simulations, we already partially address the target nonlinear case. For this reason, we consider in this section the initial value problem instead of the eigenvalue problem.

The eigenvalue solver can also benefit from the block-structured grids, due to the reduced size of the matrix of the linear operator for which the eigenvalues are found. For more details on the implementation and performance of the eigenvalue solver in GENE, we refer to [134].

In the case of linear simulations, we not only compare γ and ω obtained from the block-structured grids and the regular counterpart, but also provide speedups achieved due to the introduced grids. In this way, we check the correctness and evaluate the quality of the implementation of the code relevant for block-structured grids.

This section consists of two parts. For the sake of simplicity and clarity, in Subsection 6.1.1, we compare regular grids to block-structured grids that differ in the $x - v_{\parallel}$ subspace only. Then, in Subsection 6.1.2, the comparison with the regular grids is performed with respect to block-structured grids in the $x - v_{\parallel} - \mu$ subspace, including all velocity directions.

6.1.1 *Linear Tests for Grids in $x - v_{\parallel}$ Subspace*

We start by providing the simplest way to compare block-structured grids with their regular counterparts. We assume that the reference regular grid has a sufficiently wide range and a resolution fine enough at all radial positions in the computational grids of the velocity space. Based on this assumption, we construct the block-structured grids of the first and second types with five blocks (including more blocks do not lead to a significant reduction in the number of computational nodes, for the discussion on the number of blocks see Section 4.1). Then, we simulate a linear initial value problem with the same physical scenario on all the three grids (regular, first and second type block-structured grids).

Examples of results of such linear simulations with the TCV electron profiles (shown in Figure 9) are demonstrated in Table 5. The number of the $v_{\parallel}$ computational nodes (`nv0`) differs in each one of the three grids. The number of grid points in all other directions is kept the same for all types of grids and is given by `(nx0,nky0,nz0,nw0)` $= (128, 1, 16, 64)$. The toroidal mode wave number is fixed to $k_y = 0.1773$. The accuracy of the block-structured grids is characterized by the deviation of the growth rate (γ) and frequency (ω) of the dominant fluctuation mode from the results obtained with the regular grid. Other rows of the table (STEPS, Δt, TIME, and SPEEDUP) outline the performance the grids achieved in the provided example.

Table 5: Results of linear simulations with the TCV electron temperature profile and three different types of grids. Each grid type (regular, block-structured of first ("BS 1") and second type ("BS 2")) is shown in a different column. The rows of the table are as follows: `nv0` is the number of $v_{\parallel}$ grid points (and, in the "BS 1" column, the number of grid points in each grid-block), γ is the growth rate, ω is the frequency, STEPS is the number of steps, Δt is the time step cost, TIME is the total simulation time, and SPEEDUP is the speedup relative to the regular grid.

	grid type		
	regular	BS 1 with 5 blocks	BS 2 with 5 blocks
`nv0`	96	96, 78, 50, 40, 36	36
γ	0.371	0.371	0.357
ω	-0.056	-0.056	-0.050
STEPS	35359	35359	7779
Δt / s	0.808	0.521	0.390
TIME / s	28577	18415	3035
SPEEDUP	-	1.6	9.4

In the tests performed for linear simulations, the results stemming from the simulations with the reference regular grid ("regular" column) and the block-structured grid of the first type ("BS 1" column) are identical. The precision of the computed growth rate and frequency is at least three digits after the decimal point. The obtained results confirm our assumption that the contribution of the simulation domain located outside the step-like counter approximation of the desirable domain is negligible. This observation holds for all other test performed for linear simulations as well. The speedup achieved by the corresponding block-structured grid is 1.6 in total computational time.

The block-structured grid of the second type attains a much higher total speed of 9.4, but with some small penalty in accuracy. According to column "BS 2" in Table 5, the relative error value of the grow rate is around 3.8% only. A deviation of this size can be explained by the fact that, in this particular scenario, the fluctuating part of the distribution function is not excited at all radial distances. This can be observed in Figure 40[1]. Therefore, the precision of the results is dependent mostly on the resolution and range of the parts of the computational grid where the perturbed distribution function is localized. The chosen number of parallel velocity grid points (`nv0` $= 64$) was just sufficient to resolve this fluctuation in the vicinity of $x = 0.5$ in the regular grid. However, this number of grid points in the $v_{\parallel}$ direction would have to be increased in case of excited fluctuations in the lower temperature regions (higher x values).

Constructing the block-structured grid of the second type is based on the assumption that the regular reference grid is well resolved at all radial positions. This, however, does not hold for the current example. Consequently, the resolution of the grid-block,

1 In this and similar following plots, we use (unless otherwise specified) adapted color-maps, to combine the benefits of both linear and logarithmic color-maps, without suffering from their individual drawbacks. Similarly to linear color-maps, the adapted color-maps visualize accurately the fluctuation near $v_{\parallel} = 0$ and, like logarithmic color-maps, show well the shape of the fluctuating part (without smearing the details close to $v_{\parallel} = 0$).

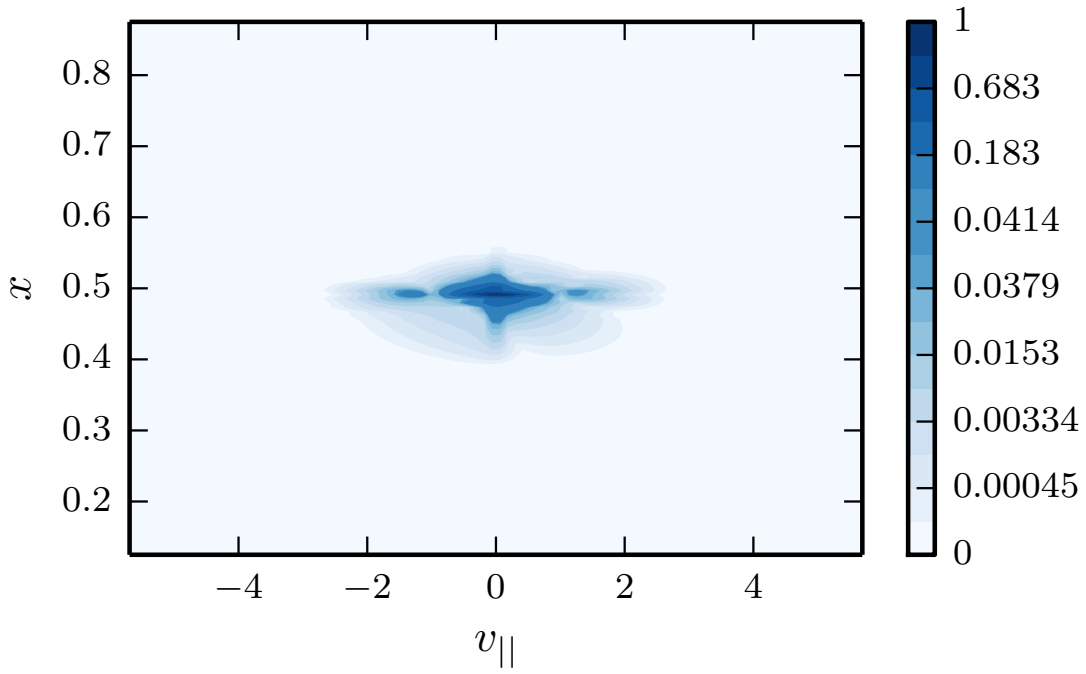


Figure 40: The absolute value of the fluctuating distribution function in the $x - v_{||}$ subspace, stemming from the linear simulation with the regular grid. The distribution function is taken at the middle of the z interval (outboard midplane position) and at the smallest magnetic moment μ_{min}, where the fluctuating part is the most prominent. The TCV density and temperature radial profiles for electrons were used in the corresponding simulation.

where the fluctuating part of the distribution function is located, is too coarse. Therefore, the gained result is not as precise as in the case of the block-structured grids of the first type, which preserves the resolution of the regular grid.

The most comprehensive comparison of the block-structured grids with the regular counterparts is given by the convergence tests shown in Figure 41. In these tests, we verify whether, for increasing numbers of $v_{||}$ grid points (`nv0`), the growth rate and frequency computed with the block-structured grids of the second type approach the values obtained with the reference regular grid. Furthermore, from the convergence plots, we judge upon the quality of the grid by looking at how fast (minimum `nv0`) the reference values of γ and ω are attained. All plots shown in Figure 41 are scaled by the converged values of γ and ω (with an accuracy of three digits after the decimal point). These values were obtained from the linear simulations with the reference grid, so that the plots can be easily associated with the relative error. The reference regular grid which leads to these converged results requires `nv0` $= 96$ grid points in the parallel velocity direction. The converged frequency ω in our linear numerical experiments is quite close to zero. As a consequence, the relative error of ω in this case might look worse than it is.

The convergence plots in Figure 41 reveal that the block-structured grid of the second type is, on the whole, more accurate than its regular counterpart, as the relative error of the block-structured grid is smaller for the same fixed `nv0`. The growth rate and frequency converge at `nv0` ≈ 70 for the block-structured grid. This does not agree with

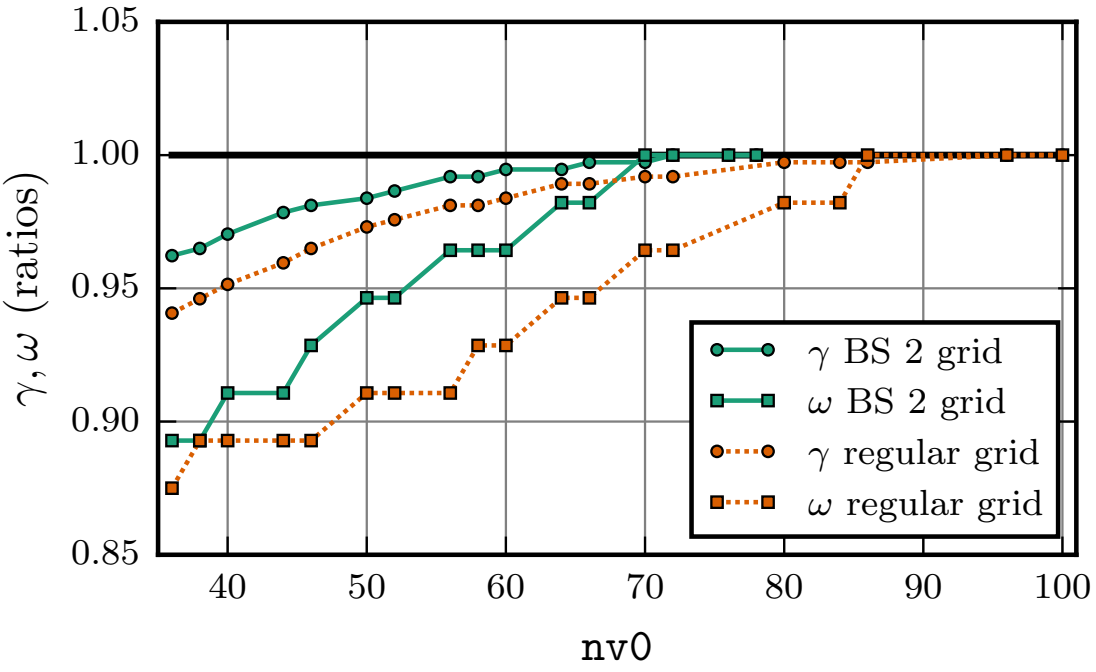


Figure 41: Convergence plots of the growth rate (γ — circle markers) and frequency (ω – square markers) of the dominant fluctuation mode obtained by linear simulations with the TCV profiles for electrons. The green solid lines correspond to simulations using block-structured grids and the orange dotted lines correspond to simulations using regular grids.

our initial assumption for the number of the $v_{\parallel}$ grid points sufficient for the block-structured grid of the second type: `nv0` $= 36$, as presented in Table 5. The reason for this is that the dominant mode has a localized pattern in the $x - v_{\parallel}$ subspace for the discussed linear run, as it was shown in Figure 40. The localization of the fluctuating part of the distribution function does not comply with our assumption that the structure of the fluctuation is similar to the one of the background distribution function.

Despite of the localization of the distribution function, block-structured grids of the second type are still very useful even for the presented type of linear simulations. We can determine the localization pattern of the dominant mode only after performing linear gyrokinetic simulations. A block-structured grid of the second type with a relatively small number of $v_{\parallel}$ grid points can be used to perform computationally inexpensive linear runs to determine the shape of the fluctuation in the $x - v_{\parallel}$ subspace. Then, to improve the results obtained with the block-structured grid, which could already be very precise, the linear runs can be repeated with a carefully adjusted regular grid. The localization pattern of the fluctuation helps us decide upon the grid ranges in the radial and parallel velocity directions, as well as upon the number of the $v_{\parallel}$ grid points in the tuned regular grid.

For the linear simulations leading to a radially localized fluctuating part of the distribution function, the block-structured grids of the second type do not reveal all their beneficial properties. This happens primarily because the simulation results are sensitive mostly to the range and resolution of the grid corresponding to the localization

area, which can be located, i. e., in one grid-block only. Therefore, in the following numerical experiments, we use the linear simulations resulting in a broader radial pattern of the dominant mode in the radial direction.

6.1.2 *Linear Tests for Grids in* $x - v_{\parallel} - \mu$ *Subspace*

Next, we present numerical results and verify the block-structured grids in the full velocity space. The easiest way to influence the resulting structure of the fluctuation in the $x - v_{\parallel} - \mu$ subspace is by specifying the radial temperature and density profiles. We observe that the radial function shown in Figure 42, chosen for the radial temperature and density dependencies in the linear runs, leads to a non-localized perturbed part of the distribution function. For the sake of having a simpler control over the resulting

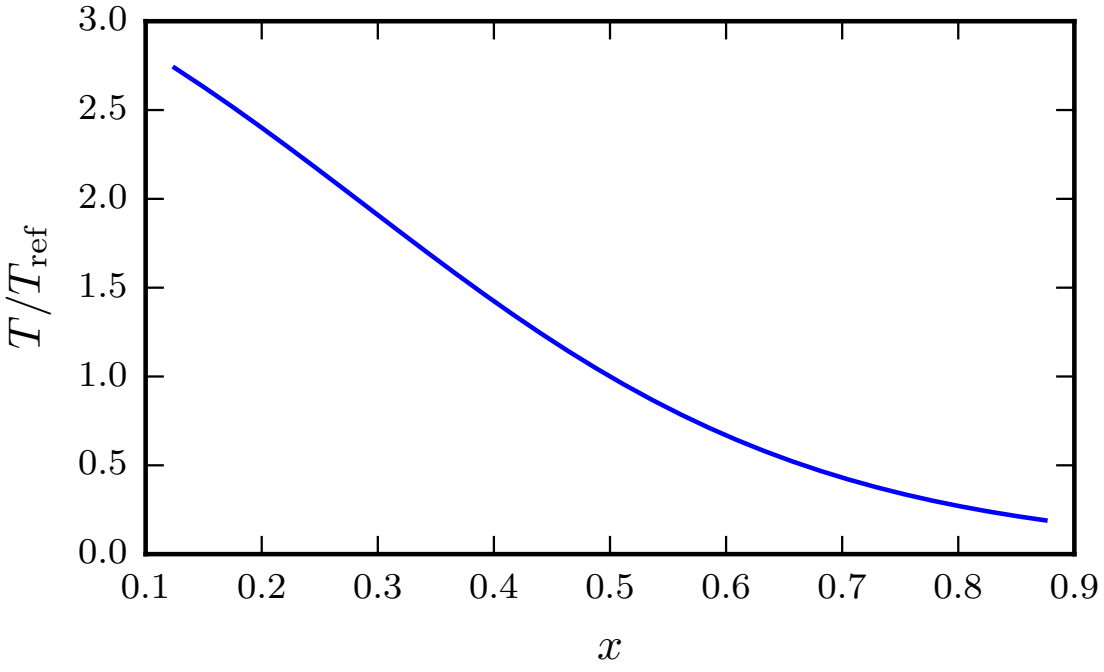


Figure 42: An example of radial temperature dependence leading to a broad structure of the distribution function in the radial direction for linear simulations with electrons and protons. The same radial function is used for the radial density profile.

shape of the distribution function, we set the radial density and temperature profiles to be the same. Furthermore, we run linear simulations with two species: electrons and protons, as the two species simulations with our radial profiles appear to be more stable numerically than simulations with one species only. Moreover, the multiple species simulations are going to be more relevant in the future. The temperature and density profiles of electrons and protons are chosen to be identical. This enables us to apply the same block-structured grid for the distribution function of each species involved in the simulations.

The described settings for the initial value problem yield a wide structure of the fluctuating part of the distribution function, as demonstrated in Figure 43 for electrons and in Figure 44 for protons. In these figures, the absolute values of the distribution

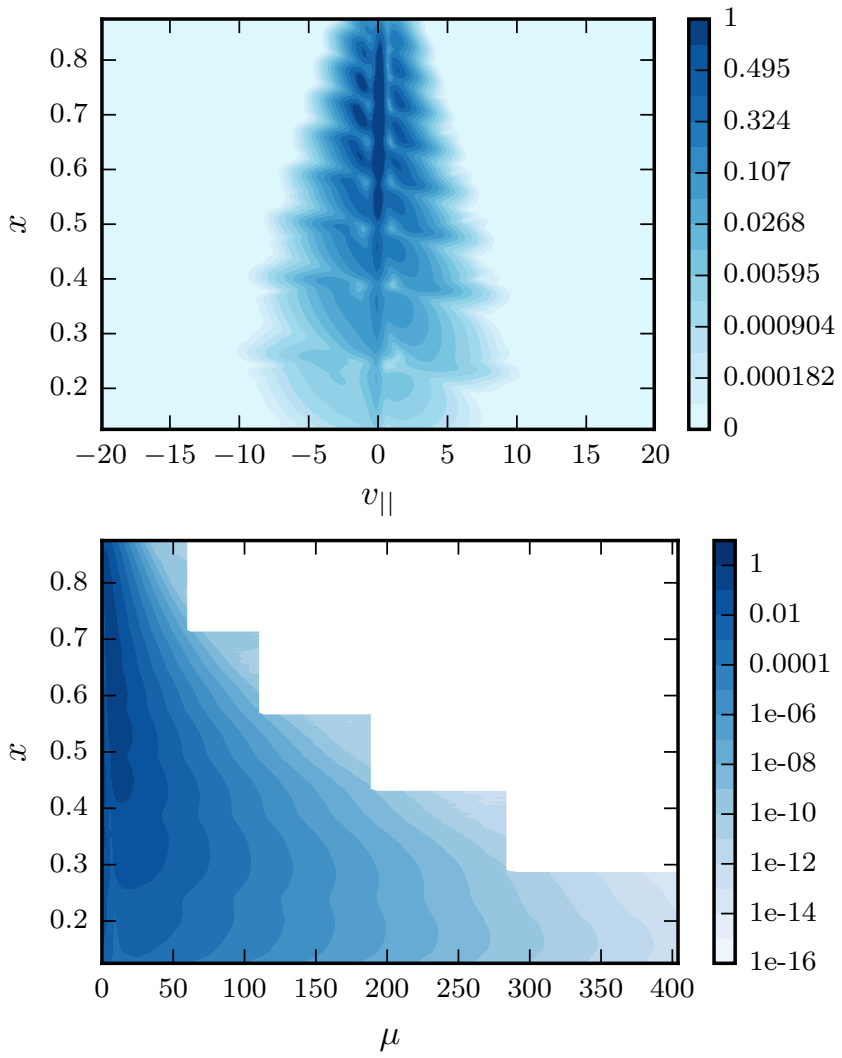


Figure 43: Projection of the absolute value of the perturbed distribution function of the electrons on the $x - v_{||}$ (top plot) and $x - \mu$ (bottom plot) planes. The parallel coordinate z is fixed at the middle of its range (outboard midplane location). The projection on the $x - v_{||}$ plane is taken at μ_{min}, and the projection on the $x - \mu$ plane at $v_{||} = 0$.

function are normalized by their respective maximum values. All the projections are taken at the middle of the parallel direction (z) interval (outboard midplane position), where the fluctuations are typically most pronounced (ballooning). The $x - v_{||}$ projections are taken at the smallest magnetic moment μ_{min}, which is determined by the Gauss-Laguerre quadrature rule, and the $x - \mu$ projections are taken at $v_{||} = 0$. At the described locations, the perturbed distribution function is usually most prominent, which makes it easier to be visualized.

We used the specially adjusted color map for the plots on the $x - v_{||}$ plane and the logarithmic color map for the plots on the $x - \mu$ plane, to make the shape of the distribution function clearly visible. As a result of the logarithmic color map, the step-like contour on the $x - \mu$ plane also becomes manifest.

In a further test involving numerical experiments with the presented radial temperature and density profiles, we compare results of second type block-structured grids

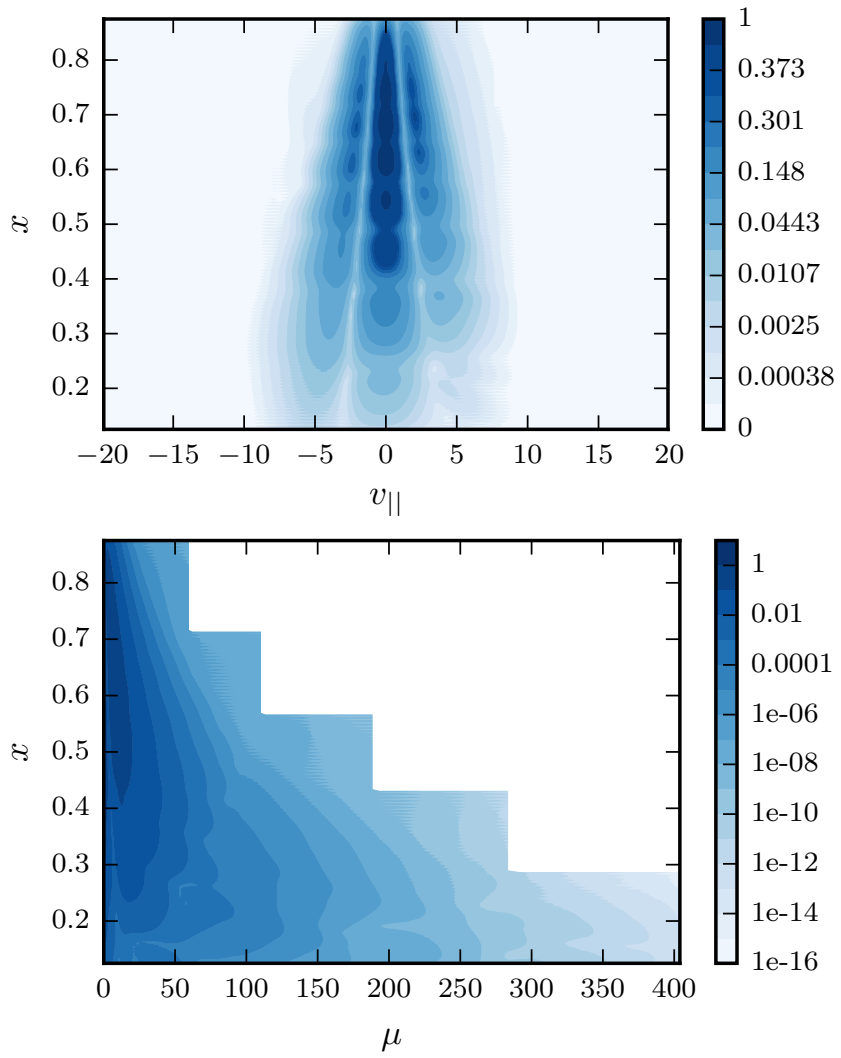


Figure 44: Projection of the absolute value of the perturbed distribution function of the protons on the $x - v_{||}$ (top plot) and $x - \mu$ (bottom plot) planes. The parallel coordinate z is fixed at the middle of its range (outboard midplane location). The projection on the $x - v_{||}$ plane is taken at μ_{min} and the projection on the $x - \mu$ plane at $v_{||} = 0$.

(with five blocks) with reference results obtained with regular grids. In this test, we again look at the convergence of γ and ω with respect to the number of grid points in the parallel velocity direction (`nv0`) and in the magnetic moment direction (`nw0`). For all other coordinates, we fix the number of grid points to relatively high values (`nx0,nz0`) $= (256, 16)$, in order to minimize numerical errors stemming from the discretization. Furthermore, in all linear runs, the toroidal mode wave number is set to $k_y = 0.1773$.

To minimize the number of linear simulations the convergence plots are obtained in two stages. First, we check how fast the results of the investigated grids converge, relative to `nv0`. For this purpose, we keep the grid in the $x - \mu$ subspace regular and fix the number of grid points in the magnetic moment direction to a relatively high value `nw0` $= 96$.

In the first stage, the simulations yield the convergence plots for nv0 demonstrated in Figure 45. The growth rate and frequency converged to the same value for both regular

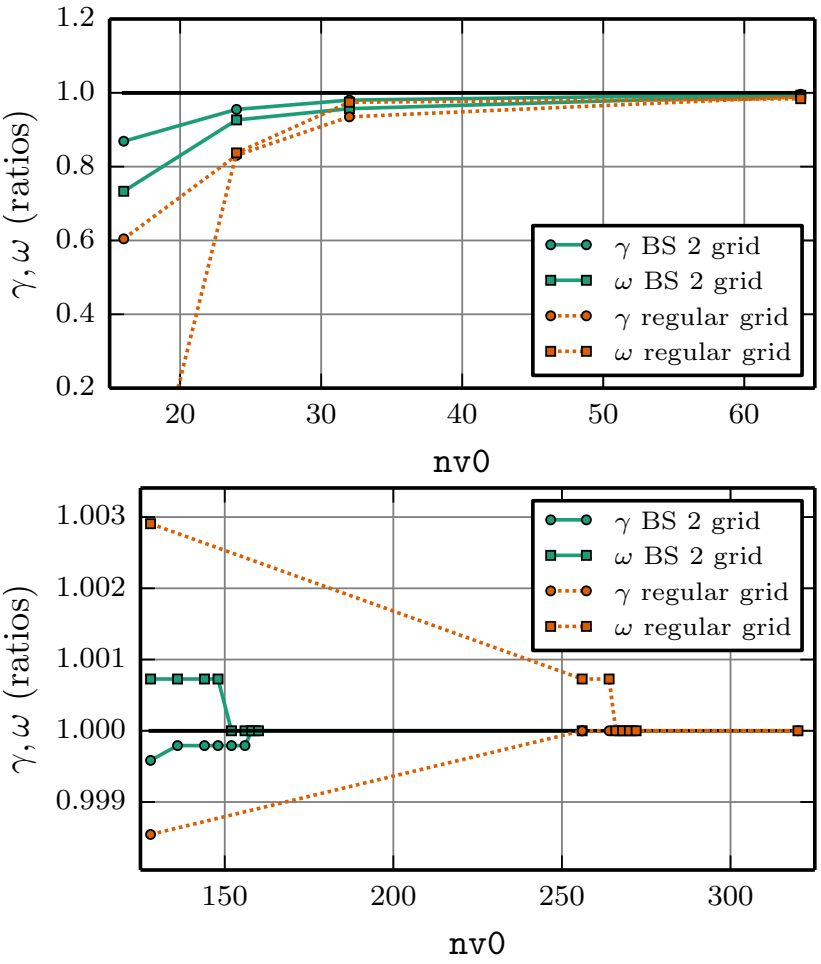


Figure 45: Convergence plots of the growth rate (γ — circle markers) and frequency (ω – square markers), for nv0 number of grid points in the parallel velocity direction. The results are obtained by linear simulations with two species: electrons and protons. The green solid lines correspond to simulations using block-structured grids in the $x - v_{\parallel}$ subspace only, whereas the orange dotted lines correspond to simulations using regular grids. The top plot shows the convergence curves for small nv0 and the bottom plot shows for large nv0.

and block-structured grids, and are given by $\gamma_{\text{conv}} = 4.810$ and $\omega_{\text{conv}} = 1.376$, with a precision of three digits after the decimal point. All results presented in Figure 45 are scaled by these converged values. From the obtained results, it becomes apparent that the block-structured grids of the second type converge already at $\texttt{nv0} = 158$, whereas the regular grids necessitate $\texttt{nv0} = 266$ grid points to achieve the same accuracy. This shows that, with the help of the second type block-structured grid in the $x - v_{\parallel}$ subspace, we can remove already around 41% of the grid points without a loss of accuracy.

In the second stage, in accordance to the results from the first convergence test, we fix the number of grid points to $(\texttt{nx0},\texttt{nz0},\texttt{nv0}) = (256, 16, 158)$ for the block-structured grids and to $(\texttt{nx0},\texttt{nz0},\texttt{nv0}) = (256, 16, 268)$ for the regular grids. Next, we run linear simulations to reveal the convergence curves of the growth rate and frequency depend-

ing on the number of computational nodes (nw0) in the magnetic moment direction. We note that, for the second stage linear runs, we use the block-structured grid in the full velocity space.

The convergence plots for the second numerical experiment are shown in Figure 46. In this experiment, the converged growth rate ($\gamma = 4.809$) and frequency ($\omega = 1.375$)

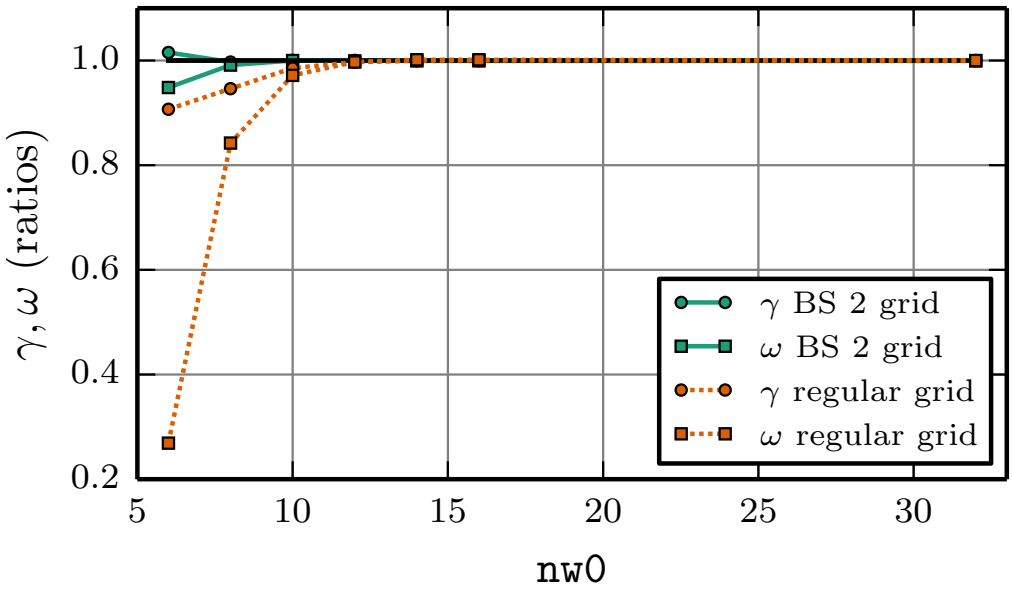


Figure 46: Convergence plots of the growth rate (γ — circle markers) and frequency (ω – square markers), for nw0 grid points in the magnetic moment direction. The results are obtained by linear simulations with two species: electrons and protons. The green solid lines correspond to simulations using block-structured grids in the full velocity space with nv0 = 158, while the orange dotted lines correspond to simulations using regular grids with nv0 = 268.

obtained with the block-structured grids of the second type differ slightly from the the reference γ_{ref} and ω_{ref}: $\Delta\gamma = 0.02\%$ and $\Delta\omega = 0.07\%$. Considering the differences in the results occurring in the linear simulations with the regular grid when starting the initial value solver with different initial perturbed distribution functions, the current differences are of a similar order and thus negligible. A potential cause for the deviations in the case of block-structured grids in the full velocity space is the removal of a large area of the $x - \mu$ subspace from the computational domain (i.e, see Figure 43 (bottom) and Figure 44 (bottom)). Furthermore, the block-structured grids in the $x - \mu$ subspace require more significant modifications in the numerical implementation than the grids in the $x - v_{||}$ subspace.

From Figure 46, we observe that both the block-structured grid of the second type and the regular grid converge very fast: at a number of grid points nw0 around 8 – 10 for the block-structured grid, and nw0 around 12 – 14 for the reference grid. Nevertheless, the important effect revealed by the convergence plots is that the block-structured grids are much more accurate at a coarse resolution than their regular counterparts. Furthermore, in this numerical experiment, the regular grids are also helped by the fact that they have more parallel velocity grid points than the block-structured grids do.

For the sake of a fair comparison between block-structured and regular grids, we also provide the convergence plots obtained for these two grids with same number of $v_{\parallel}$ grid points fixed to `nv0` $= 24$, see Figure 47. In this figure, we observe that

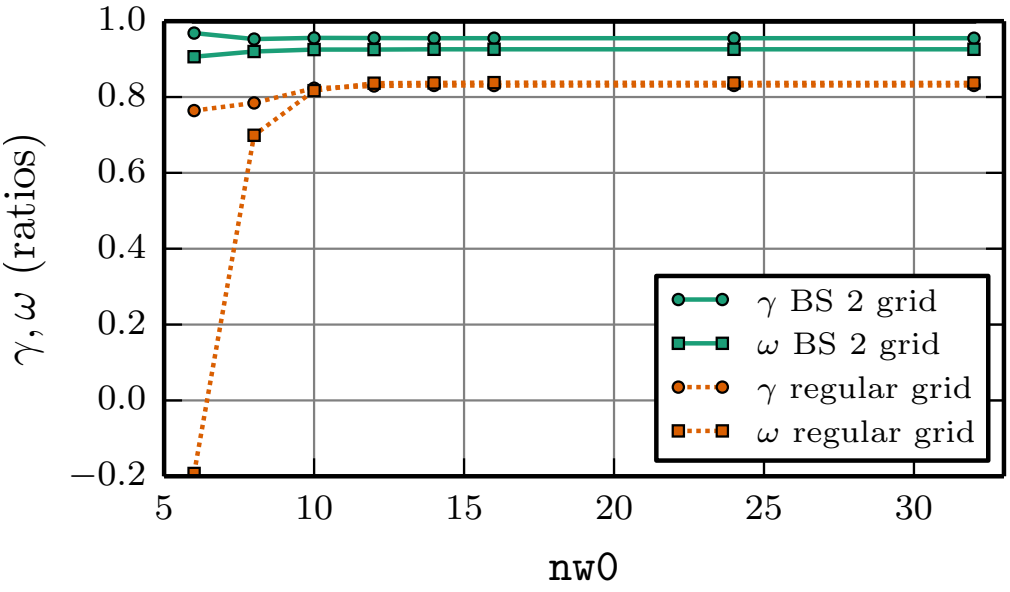


Figure 47: Convergence plots of the growth rate (γ — circle markers) and frequency (ω – square markers) for `nw0` grid points in the magnetic moment direction. The results are obtained by linear simulations with two species: electrons and protons. The green solid lines correspond to simulations using block-structured grids in the full velocity space and the orange dotted lines correspond to simulations using regular grids. Both types of grids have coarse resolution in the $v_{\parallel}$ direction with `nv0` $= 24$.

the accuracy gap between the block-structured and regular grids becomes even more evident. The ability of the block-structured grids to preserve the accurate results with even coarse resolutions in the velocity space is useful for methods that combine results stemming from linear simulations on coarse grids to get a final accurate estimate. For details on such methods applied in GENE, we refer to [107, 135–137].

Finally, we compare the performance of the linear simulations with the block-structured grids of the second type to the performance of their regular counterparts. For a fair comparison, we examine the block-structured grid with `(nv0,nw0)` $= (158, 10)$ computational nodes in the velocity space and the regular grid with `(nv0,nw0)` $= (268, 14)$. According to the convergence plots in Figure 45 and Figure 46, these two grids attain converged results with the same precision. To obtain the performance estimates comparable to real life situations, we apply the same compiler optimization for these measurements like for GENE production runs. In the linear runs of the initial value solver with the block-structured and regular grids, we use 32 CPUs.

With these settings, the block-structured grid of the second type converged in 19 429 s and the regular grid in 39 346 s. The achieved speedup is 2.03 and thus somewhat smaller than the reduction in the number of the grid points, which is 2.36. The difference between these two numbers is explained by the computation and communication overhead on the boundaries of the grid-blocks. As explained in Chapter 5, additional interpolations are performed due to the misalignment of the radial coordinate lines in

each grid-block. Moreover, additional side-exchanges are carried out due to the step-like process grid boundaries in the case of the block-structured grids of the second type. Nevertheless, the overheads observed in the linear runs become less apparent in the nonlinear simulations, as will be shown in the following section.

6.2 NONLINEAR SIMULATIONS

In this section, we provide results from nonlinear gyrokinetic simulations. In this case, the block-structured grids of the first type yield precise results, just like in the case of linear simulations (see Section 6.1). However, this type of block-structured grids do not lead to a high reduction in the number of the grid points in the velocity space. Therefore, in this section, we compare the regular grids with the block-structured grids of the second type only, as they are more promising.

As described in Subsection 6.1.1, it is possible that linear gyrokinetic simulations result in a dominant mode localized in the radial direction. This is usually not a case for nonlinear gyrokinetic runs, where several toroidal modes are excited simultaneously. For example, the perturbed part of the distribution function obtained from the linear runs with the TCV radial temperature and density profiles is localized, as shown in Figure 40. However, nonlinear runs with the same settings lead to a broad shape of the distribution function, as shown in Figure 48. Consequently, to demonstrate the benefits of the block-structured grids for nonlinear simulations, it is not necessary to tune the temperature and density profiles, like in the case of linear simulations. In the following numerical experiments, all results are obtained with the TCV temperature and density profiles, see Figure 9.

There are several reasons why it is more difficult to compare results from nonlinear gyrokinetic simulations than from linear simulations:

- The analysis of a nonlinear run requires numerous observables, for example, heat, particle, momentum fluxes, etc. These observables are obtained by postprocessing, which involves integration in the five dimensional phase space. As a result, a lot of details are lost and do not influence the observables at hand.

- The fluctuating part of the distribution function is sensitive to initial conditions. Consequently, a small modification in the computational grid leads to very different values of the distribution function after the initial value solver performs several time steps. Therefore, only quasi-stationary features are of interest, which are obtained by time averaging the perturbed part of the distribution function on a sufficiently long time interval that excludes the initialization phase of the nonlinear simulation.

- There are no clear criteria when to stop a nonlinear run. The simplest but not mathematically strict way to decide when to terminate nonlinear runs checks whether the time averaged simulation results are no longer changing over increasing time intervals.

To address the described challenges, in this section, we provide the heat fluxes obtained from different computational grids and compare the time averaged fluctuating parts of the distribution function.

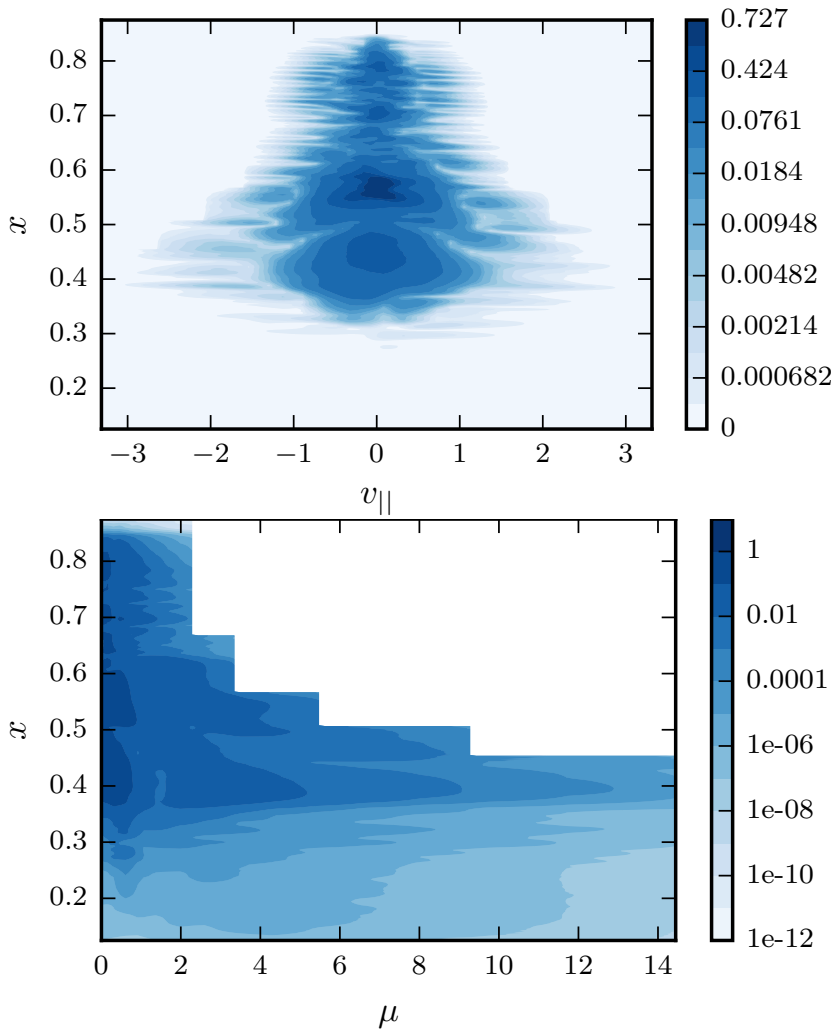


Figure 48: Time-averaged fluctuating part of the distribution function (in absolute value) on the $x - v_{||}$ (top plot) and $x - \mu$ (bottom plot) planes for nonlinear simulations with TCV radial profiles for electrons. The parallel coordinate z is fixed at the middle of its range (outboard midplane location). The projection on the $x — v_{||}$ plane is taken at μ_{min} and the projection on the $x - \mu$ plane at $v_{||} = 0$.

The nonlinear gyro-kinetic simulations are much more computationally expensive than the linear simulations. Consequently, obtaining convergence plots like in the linear case is almost infeasible. Therefore, we verify the proposed block-structured grids in a different manner. Specifically, we compare the results obtained using block-structured grid with those obtained using three regular grids:

- The first regular grid has fine resolutions in all five dimensions and yields accurate results according to the experience gained before the comparison runs. In the following, we refer to this grid as a reference regular grid. By comparing results obtained using the block-structured with results from reference grids, we check whether our proposed grids achieve the goal of removing computational nodes from the velocity space grid without loss of accuracy.

- The second regular grid — the first alternative grid — has the same coordinate ranges as the reference grid, but a coarse resolution in the velocity space. The number of grid points in the velocity directions is the same as for the block-structured grid. The simulation results obtained using this regular grid help us check whether a simple coarsening of the velocity space grid could lead to results of the same accuracy as the reference grid.

- The third regular grid — the second alternative grid — has the same ranges of the velocity directions as the smallest grid-block of the block-structured grid. Furthermore, the resolution of this grid is the same as for the reference regular grid (or the number of grid points is the same as in the block-structured grid). The results obtained with the second alternative grid show us whether a simple reducing of the velocity space ranges of the reference grid could potentially attain the accuracy of the reference grids.

To summarize, by performing comparisons between the block-structured grid and the aforementioned regular grids, we check whether the former grid is as accurate as the reference counterpart with the largest number of computational nodes. We also check whether alternative regular grids with the same number of grid points as the block-structured grid can achieve this goal.

The block-structured grids of the second type were initially implemented for the $x - v_{\parallel}$ subspace. After a successful verification, this type of grids was extended to include the magnetic moment direction. In this section, therefore, we first present results for the proposed grids in the $x - v_{\parallel}$ subspace (Subsection 6.2.1), and then for the grids in the $x - v_{\parallel} - \mu$ subspace (Subsection 6.2.2).

6.2.1 *Nonlinear Tests for Grids in $x - v_{\parallel}$ Subspace*

In the tests performed on the block-structured grids in the $x - v_{\parallel}$ subspace, the reference grid has the following numbers of computational nodes $(\texttt{nx0}, \texttt{nky0}, \texttt{nz0}, \texttt{nv0}, \texttt{nw0}) = (512, 16, 16, 40, 64)$. This holds also for the block-structured and two alternative regular grids, except that these grids have $\texttt{nv0} = 18$ computational nodes in the parallel velocity direction.

Four examples of these grids on the $x - v_{\parallel}$ plane are schematically illustrated in Figure 49. In this figure, the reference regular grid is represented by the plot with label (**RR**), the block-structured grid of the second type with five blocks (as in the case of linear tests) by the plot with label (**BS**), the first alternative regular grid by the plot with label (**1A**), and the second alternative grid by the plot with label (**2A**). For the sake of visibility, the resolutions of the grids were coarsened by a factor of four in the x direction and by a factor of two in the $v_{\parallel}$ direction (compared to the actual grids used in the nonlinear simulations).

The easiest way to compare the time-averaged perturbed parts of the distribution function stemming from different grids is the plot over line method. The line position is chosen in so as to traverse the areas with the strongest fluctuation and span over all radial positions. This location is scenario dependent. For the simulations with the TCV electrons temperature and density profiles, the fluctuating part of the distribution function is strongest at $v_{\parallel} = 0$ and $\mu_{\min}$ in the velocity space, at the outboard midpoint position (at the middle of parallel direction z interval) in the position space with the

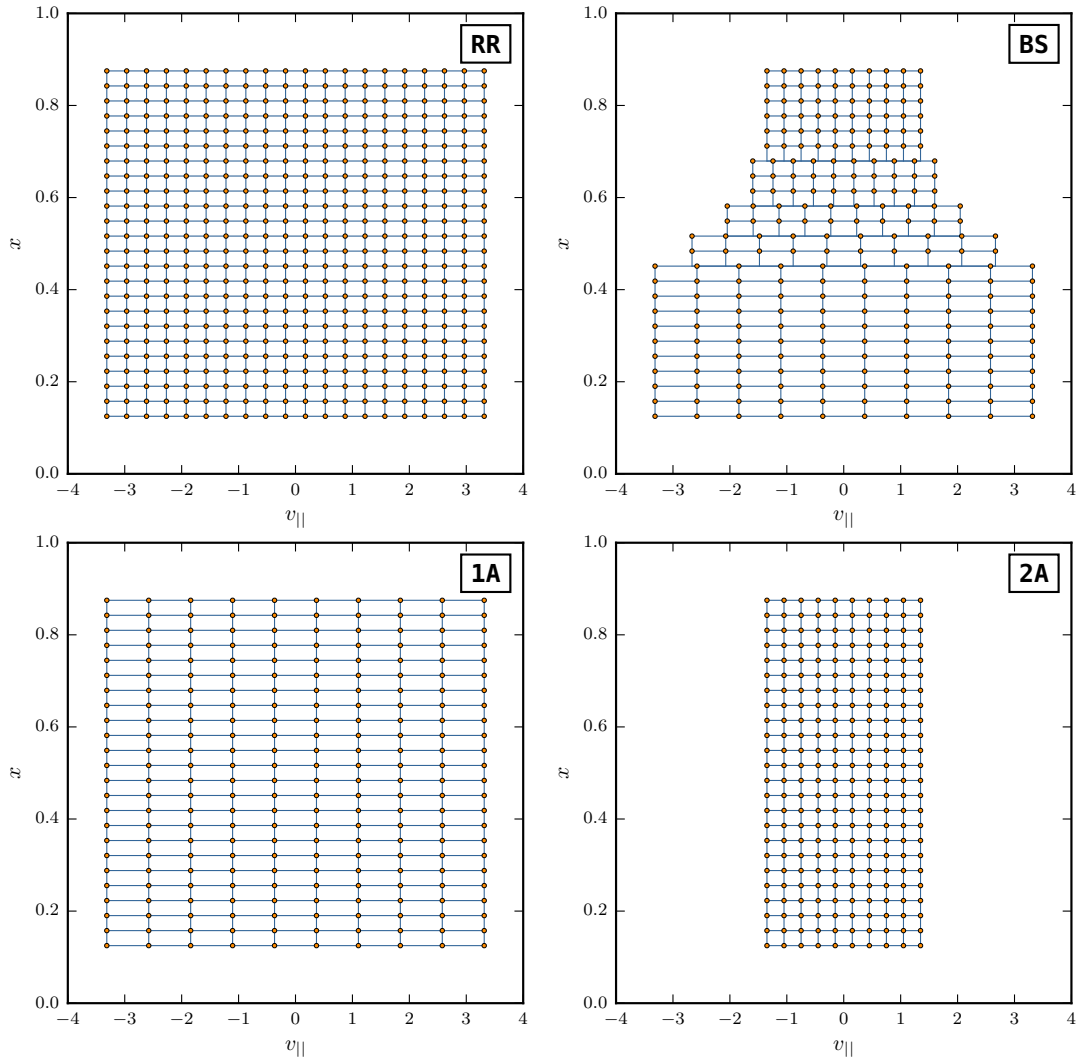


Figure 49: Examples of four types of grids: the reference regular grid marked by (**RR**), the block-structured grid marked by (**BS**), the first alternative grid marked by (**1A**), and the second alternative grid marked by (**2A**). Actual grids used in nonlinear simulations have a finer resolution.

toroidal wave number $k_y = 0$. An example of the time-averaged fluctuation (for nonlinear runs with the TCV radial profiles) in the $x - v_{||}$ subspace at the outboard midpoint position with fixed μ_{min} and $k_y = 0$ is shown in Figure 48 (top). In our comparisons, therefore, we take the time-averaged absolute values of this fluctuation at the vertical line in the middle of the plot ($v_{||} = 0$), which covers the whole range of x.

Corresponding plots over line for the four previously described grids are shown in Figure 50. These plots were produced by using the time-averaged distribution function in the $18\,R/c_s$ time interval[2], which is three time longer than the initialization phase ($6\,R/c_s$). The time-averages were computed with around 30 – 40 samples at different time steps.

2 R is major radius, c_s is ion sound speed.

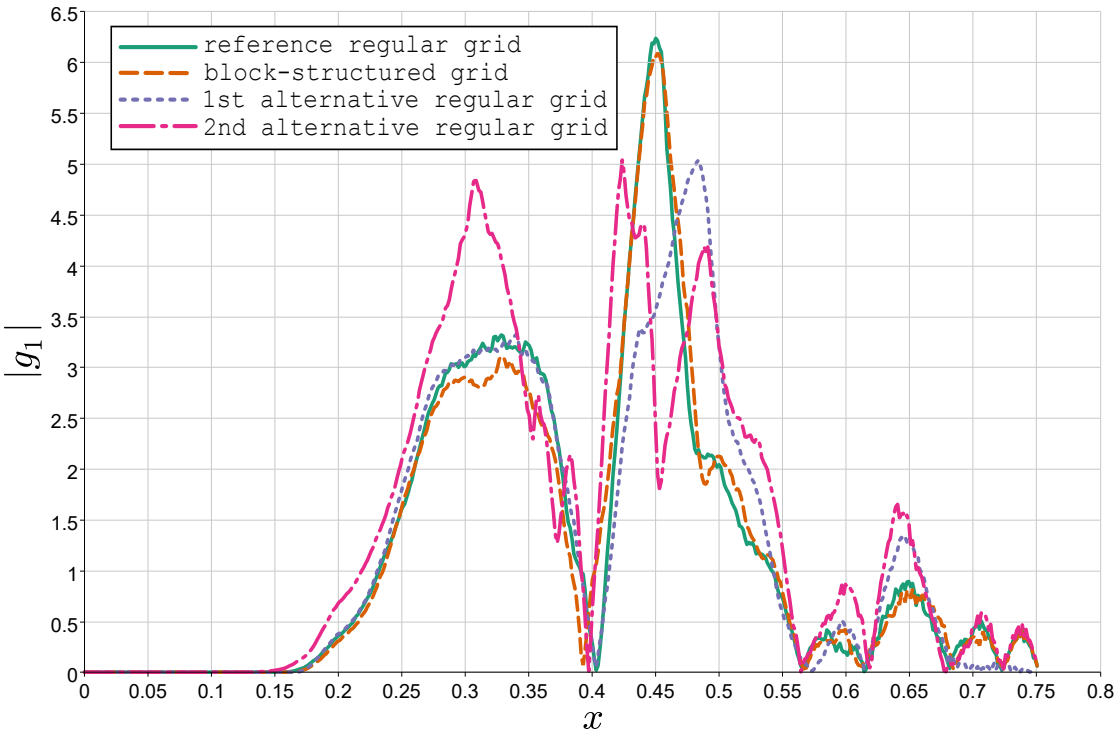


Figure 50: Plots over line of the absolute value of the distribution function for four different grids. The toroidal mode wave number is fixed to $k_y = 0$. The green solid line corresponds to the reference regular grid, the orange dashed line to the block-structured grid, the violet dotted line to the first alternative regular grid, and the magenta dash-dotted line to the second alternative regular grid.

In Figure 50, the plot over line of the regular reference grid (**RR**) (with a fine resolution in the $v_{||}$ direction) is represented by the green solid line, the plot over line of the block-structured grid (**BS**) by the orange dashed line, the plot over line of the first alternative regular grid (**1A**) (with a coarse resolution in the $v_{||}$ direction) by the violet dotted line, and the second alternative regular grid (**2R**) (with a short range in the $v_{||}$ direction) by the magenta dash-dotted line. From these plots, we observe that the plots over line of the (**RR**) and (**BS**) grids match at all radial positions. The (**1A**), however, does not resolve the fluctuating part of the distribution function at high x (the last two peaks), while the plot over line of the (**2A**) grid deviates strongly from the reference curve at all radial positions except at high x. This numerical experiment shows that our block-structured grid is capable to resolve well the distribution function at all radial positions with less grids points than the reference grid, whereas the alternative regular grids with the same number of computational nodes as the block-structured grid fail to provide reliable results.

In the next scenario, we compare the fluctuating part of the distribution function for a dominant turbulent component $k_y \neq 0$ driving the turbulent transport. To obtain a spectrum with a dominant toroidal mode, we suppress the electron scale instability in the described nonlinear runs by adding hyper-diffusion in the x and y directions. Furthermore, we save computational time by continuing the already performed nonlin-

ear runs with added hyper-diffusion, by starting the new simulations from generated checkpoints.

The reference regular grid for the nonlinear runs with added hyper-diffusion requires `(nx0,nk0,nz0,nv0,nw0)` $= (512, 16, 16, 82, 64)$, while all other three grids have `nv0` $= 34$ grids points in the parallel velocity direction. The dominant mode in the corresponding simulations has the wave number $k_y = 0.3546$.

For the nonlinear simulations with hyper-diffusion, we compare not only the plots over lines, but also the heat fluxes. All results are obtained by time-averaging on the time interval $20\,R/c_s$, with $40 - 50$ samples at different time steps.

The heat fluxes, obtained with four different grids ((**RR**), (**BS**), (**1A**), and (**2A**)), are summarized in Table 6. The results from the table show that only the block-structured

Table 6: Heat fluxes stemming from four different grids ((**RR**), (**BS**), (**1A**), and (**2A**)) averaged over a time interval of $20\,R/c_s$ at the whole radial range excluding buffer zones and provided in gyro-Bohm units.

	grid type			
	reference regular (**RR**)	block-structured (**BS**)	1st alternative (**1A**)	2nd alternative (**2A**)
Q_{gb}	1.63	1.62	1.53	3.70

grid yields a heat flux value comparable to the one obtained with the reference regular grid.

The plots over line obtained from simulations with hyper-diffusion in the x and y directions are shown in Figure 51 for all four grids. The time averaging interval is taken the same as for the heat fluxes ($20\,R/c_s$). The location of the line along which we plot the absolute value of the perturbed part of the distribution function is the same as for the nonlinear runs without hyper-diffusion (see Figure 50). The only difference is that we now consider the dominant toroidal mode, with $k_y = 0.3546$. In this numerical experiment, the block-structured grid yields again the closest matching plot over line (orange dashed line) to the reference regular grid plot (green solid line).

In the provided examples involving nonlinear simulations, the reduction in the number of computational nodes due to the block-structured grid of the second type is around 2.2. Theoretically, this leads to the same speedup. In practice, however, the speedup depends on the computer architecture and the chosen parallelization scheme. In our numerical experiments with $32 - 64$ CPUs, the observed speedups vary in the range $1.9 - 3.0$. This is already a valuable performance improvement, considering the enormous computational costs of the nonlinear gyrokinetic simulations.

Even higher performance improvements are achieved with the block-structured grids of the second type in the full velocity space. The corresponding numerical experiments are presented in the following subsection.

6.2.2 *Nonlinear Tests for Grids in $x - v_{\parallel} - \mu$ Subspace*

In the numerical tests run for the block-structured grids in the full velocity space, we again consider four different types of grids: a reference regular grid, a block-structured

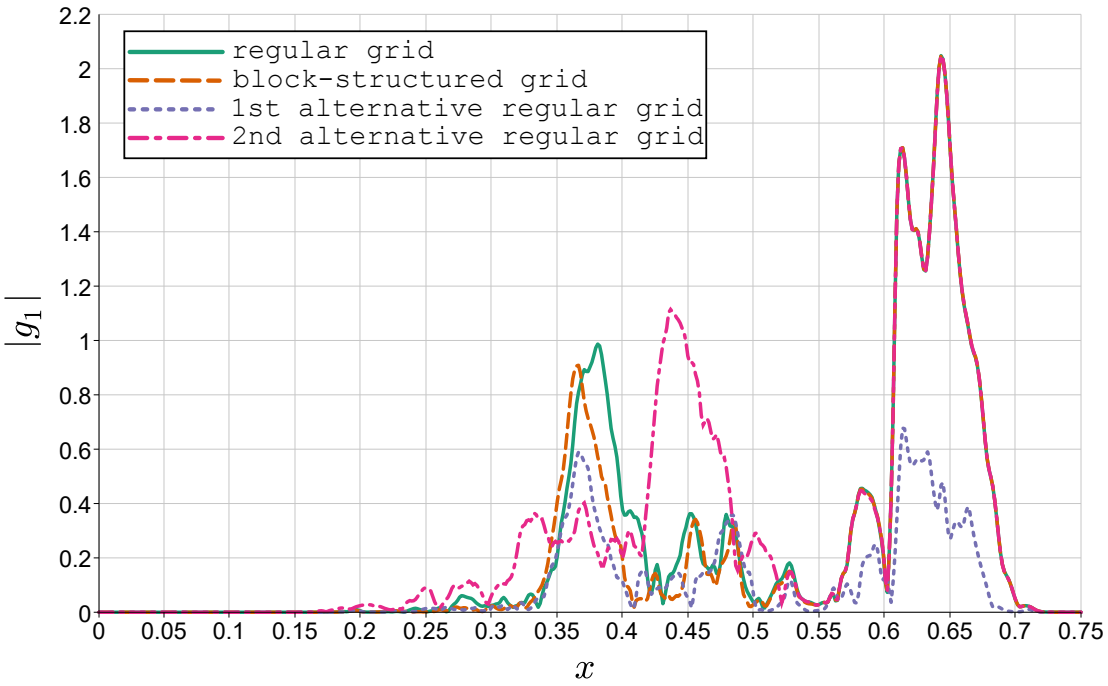


Figure 51: Plots over line of the absolute value of the distribution function for four different grids. The results are obtained from nonlinear runs with hyper-diffusion in the x and y directions. The plots are shown for the dominant toroidal mode with $k_y = 0.3546$. The green solid line corresponds to the reference regular grid, the orange dashed line to the block-structured grid, the violet dotted line to the first alternative regular grid, and the magenta dash-dotted line to the second alternative regular grid.

grid of the second type with five blocks (as for the $x - v_{\parallel}$ subspace), the first alternative regular grid with a coarse resolution, and the second alternative regular grid with short ranges in the velocity directions. The outlines of all these four grids, labeled by (**RR, BS, 1A, 2A**), are shown in Figure 52.

On this occasion, we run numerical simulations with hyper-diffusion in the x and y directions from the beginning, so that we obtain spectra with a dominant toroidal mode above the grid scale. The reference regular grid has the following number of computational nodes `(nx0,nk0,nz0,nv0,nw0)` $= (512, 16, 16, 82, 64)$. The other three grids have less grid points in the velocity space `(nv0,nw0)` $= (34, 16)$ and the same number of the grid points as the reference grid in the position space.

The reference regular grid is essentially the same like in Subsection 6.2.1. The other three grids, however, are different in the velocity space. Therefore, this time, we do not reuse the checkpoints, but rather restart the initial value solver with the same initial perturbed part of the distribution function for all four grids. The dominant toroidal mode in the quasi-stationary state reached after an initialization phase has the wave number $k_y = 0.5319$, which is different from the one obtained in the previous numerical experiment. This means that the current nonlinear runs yield a different quasi-stationary state than the previous nonlinear simulations did. Attaining different quasi-stationary states for one physical scenario is, nevertheless, normal for nonlinear gyrokinetic simulations.

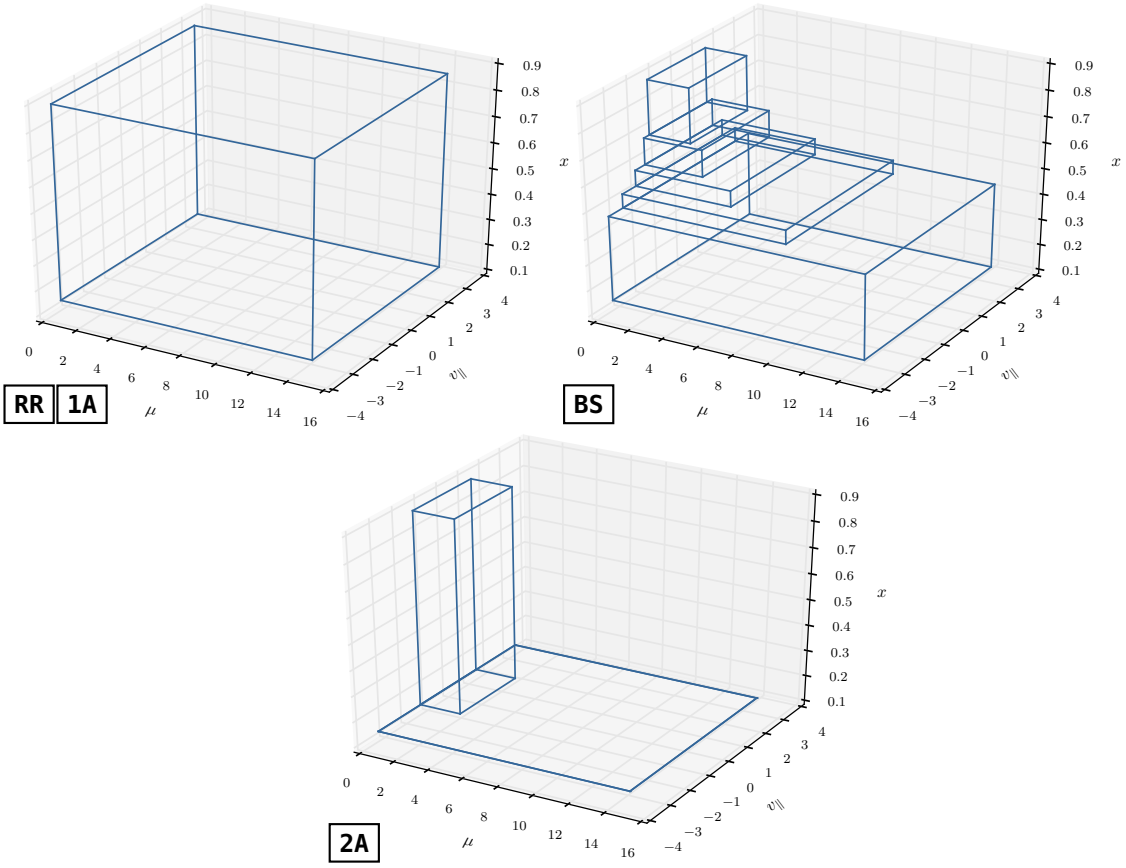


Figure 52: The outlines of four types of grids in the $x - v_{||} - \mu$ subspace: (top left) the reference regular grid (**RR**) and the first alternative regular grid (**1A**) with a coarse resolution in the velocity space; (top right) block-structured grid (**BS**); (bottom plot) the second alternative regular grid (**2A**) with short ranges in the velocity space.

We compute the heat fluxes and the plots over line averaged out on the time interval $35\,R/c_s$, with a minimum of 45 samples at different time steps. The obtained heat fluxes are provided in Table 7. From this table, we observe that the block-structured grid of the second type (**BS**) yields the heat flux closest to the regular reference grid (**RR**). The second closest result belongs to the first alternative regular grid (**1A**). The second alternative regular grid (**2A**) results in a significantly different heat flux compared to the reference one.

The plots over line for all four grids are shown in Figure 53 and Figure 54. In these figures, the results are demonstrated for the toroidal mode with the most prominent fluctuating part of the distribution function ($k_y = 0$) and for the dominant toroidal mode causing turbulent transport ($k_y = 0.5319$), respectively.

From the obtained heat fluxes and plots over line, we observe the following:

- The block-structured grid of the second type (**BS**) achieves the closest heat flux to the reference one. It yields also the closest match of the fluctuating part of the distribution function to the reference regular grid (**RR**).

Table 7: Heat fluxes (given in gyro-Bohm units) stemming from four different grids ((**RR**), (**BS**), (**1A**), and (**2A**)), averaged over a time interval of $35\,R/c_s$ at the whole radial range excluding buffer zones.

	grid type			
	reference regular (**RR**)	block-structured (**BS**)	1st alternative (**1A**)	2nd alternative (**2A**)
Q_{gb}	1.92	1.83	1.64	0.78

- The first alternative regular grid (**1A**) attains a quite accurate heat flux. However, this grid does not resolve the distribution function in the low temperature region corresponding to $x \in (0.55, 0.75)$, as shown in Figure 53 and Figure 54.
- The second alternative regular grid (**2A**) yields an unacceptably small heat flux. However, the plot over line obtained with this grid is surprisingly close to the reference regular grid (**RR**). This demonstrates that, in the obtained quasi-stationary state, the fluctuating part of the distribution function at the location of the line is not influenced by the shortened ranges in the velocity directions. However, this is a rather specific case and cannot be generalized. For example, in the nonlinear run, whose plot over line is shown in Figure 50, the absolute values of the distribution functions stemming from the (**2A**) grid do not match the values stemming from the (**RR**) grid.

Consequently, we conclude that, in the numerical nonlinear test for the grid in the $x - v_{\parallel} - \mu$ subspace, only the block-structured grid (**BS**) achieves results comparable with those obtained from the reference regular grid.

It can be thus observed that the block-structured grid of the second type in the full velocity space reduces the number of computational nodes without a loss of accuracy. Moreover, it also leads to much faster nonlinear simulations compared to its reference regular counterpart.

To estimate the performance improvements, we measure how much time it takes to execute one time iteration in the previously discussed nonlinear simulation with the block-structured and reference regular grids. The corresponding times are obtained by performing 100 time iterations and taking the average times. In this numerical test, we use compiler optimization flags, like for production runs with GENE. The nonlinear runs are performed with 128 CPUs for both types of grids. The block-structured grid (**BS**) requires 6.2 s and the reference regular grid (**RR**) 64.8 s for one time step. The corresponding speedup is 10.5, which is even larger than the reduction in the number of grid nodes $(82 \cdot 64)/(34 \cdot 16) \approx 9.4$. The superlinear speedup can be explained by a cache effect, which outweighs the overheads in the block-structured grid caused by the side-exchanges during the communication and the additional interpolations for the radial derivative computations. For details on the implementation see Chapter 5.

In some nonlinear simulation scenarios, we observe that wide ranges in the velocity directions might cause numerical instabilities in low temperature regions. The block-structured grid can easily fix this issue by restricting the velocity ranges. This is another useful property of the proposed grids.

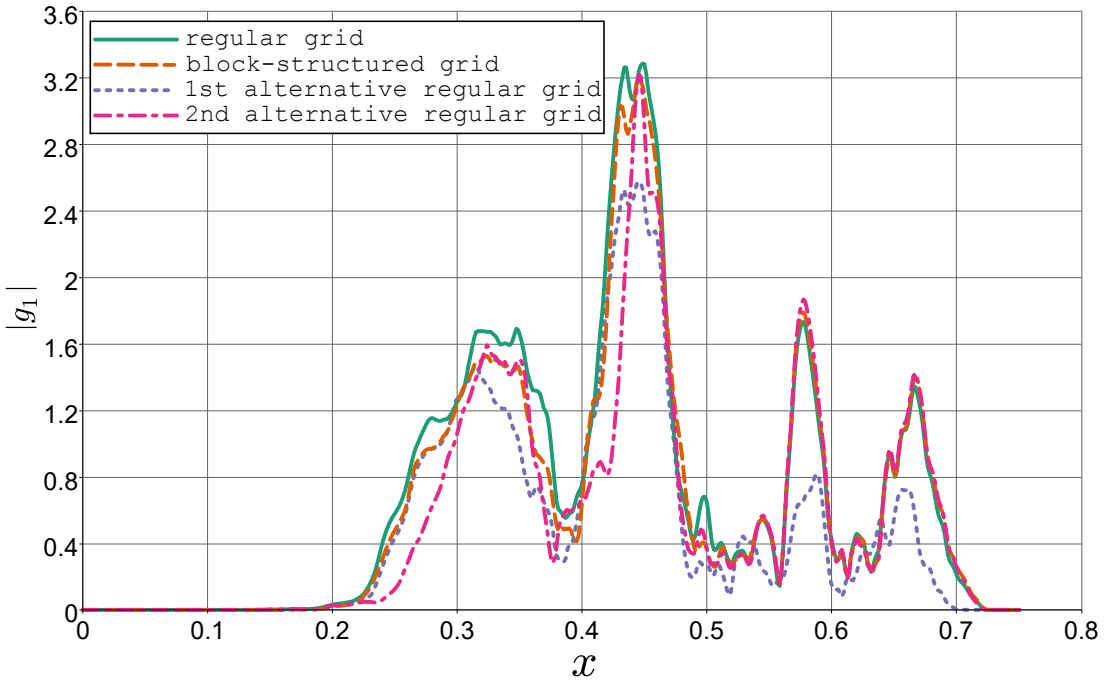


Figure 53: Plots over line of the absolute value of the distribution function for the toroidal mode with wave number $k_y = 0$. The reference regular grid (**RR**) is represented by the green solid line, the block-structured grid (**BS**) is represented by the orange dashed line, the first alternative regular grid (**1A**) is represented by the violet dotted line, and the second alternative regular grid (**2A**) is represented by the magenta dash-dotted line.

6.3 SUMMARY

In this chapter, we presented results and estimated performance improvements obtained by introducing the block-structured grids for linear and nonlinear radially global gyrokinetic simulations. The whole concept of the block-structured grids was developed initially for the $x - v_{||}$ subspace and only then for the $x - v_{||} - \mu$ subspace. Therefore, in each section, we first tested the proposed grids considering first just the parallel velocity coordinate and only then including the full velocity space.

The linear gyrokinetic tests presented in Section 6.1 serve as our main benchmark for two reasons. First of all, the linear runs are computationally much cheaper than the nonlinear ones. Secondly, it is quite easy to compare the results stemming from different grids, as the corresponding microturbulence is characterized primarily by two parameters: the linear growth rate (γ) and real frequency (ω). To verify the results stemming from the block-structured grids, we ran convergence tests for γ and ω to check how fast these parameters reach a three-digit precision after decimal point when the number of computational nodes in the velocity directions is increased. In all convergence tests, the block-structured grids achieved the prescribed precision with less grid points than the regular counterpart. This resulted in a speedup of 2.03 of the proposed grids in the full velocity space. Furthermore, the convergence plots revealed an additional useful feature of the block-structured grids: they still provide reliable results even for extremely coarse velocity space grids, when the regular grids do not function

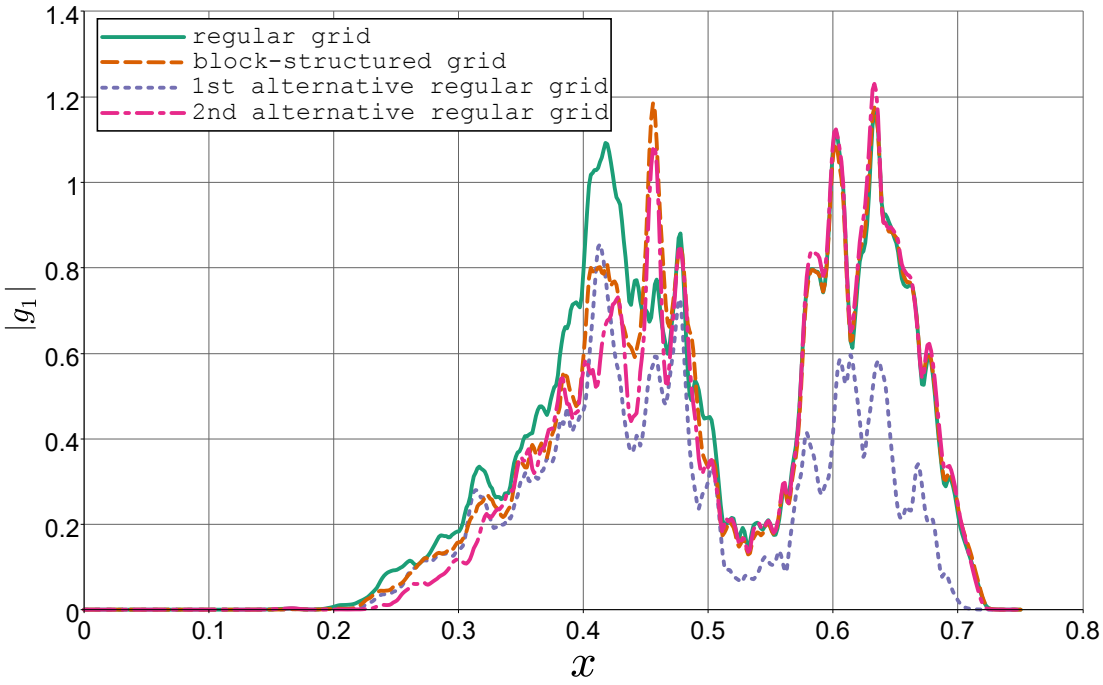


Figure 54: Plots over line of the absolute value of the distribution function for the toroidal mode with wave number $k_y = 0.5319$. The results are shown for four grids: (**RR**) — the green solid line, (**BS**) — the orange dashed line, (**1A**) — the violet dotted line, (**2A**) — the magenta dash-dotted line.

anymore. This property is useful for methods combining results obtained with coarse grids to compute a final accurate result.

The main target of the block-structured grids is given by nonlinear gyrokinetic simulations, which were discussed in Section 6.2. Results stemming from nonlinear runs are more challenging to verify, because of reasons such as:

- extremely high computational costs
- a vast number of possible observables
- sensitivity to initial conditions
- only quasi-stationary (time-averaged) results can be compared
- no clear termination criteria

Taking this into consideration, the convergence-like tests become both overly expensive computationally and also less revealing than those used for the linear runs. Therefore, we developed another method to estimate the quality of our proposed grids. Specifically, we compared a block-structured grid with three regular counterparts:

- a reference grid with a large number of grid points in both velocity directions
- two alternative regular grids with the same number of grid points in the velocity space as the block-structured grid

- the first alternative grid had the same ranges, but coarse resolutions in the velocity directions compared to the reference grid; it was used to check whether a simple coarsening of the velocity space can achieve results of the same accuracy as the block-structured grids
- the second alternative grid had the same resolutions, but shorter ranges in the velocity directions than the reference grid had; it served to examine whether taking shorter velocity ranges can reduce the number of computational nodes without a loss in accuracy, as block-structured grids were able to do

For all these four types of grids, we compared time averaged heat fluxes and absolute values of the distribution function on a prescribed line. The comparison results revealed that only the block-structured grid yielded results comparable by accuracy to the results obtained with the reference regular grid. Furthermore, the block-structured grid attained a speedup of 10.5 in comparison to the reference regular grid, which entails a significant reduction in the computational costs.

7 CONCLUSION

The current work addressed a critical and challenging problem considering the spatial temperature variation in radially global gyrokinetic simulations. This problem is known to necessitate a huge number of computational nodes for the grids in the velocity space mixed with the radial distance. Thus, many important physical scenarios are overly expensive computationally, making methods that reduce the number of grid points prerequisite.

Solutions proposed in this work relied on introducing block-structured grids and applying these in the context of the gyrokinetic simulations. Such grids consisting of multiple connected blocks are used in a vast number of scientific applications to accurately resolve geometry, i. e., the boundaries of the simulation domain. However, the exact shape of the domain boundaries in the velocity space does not play an important role in gyrokinetic simulations. Instead, in this context, the proposed grids focus on choosing adequate resolutions and ranges for the parallel velocity direction, as well as sets of magnetic moment nodes optimal for quadrature-like operations in each grid-block along the radial direction.

The contribution of this work can be roughly classified into three categories:

- *Concepts:* We introduced block-structured grids and compared this approach with the alternative method of transformed velocity coordinates. We opted for block-structured grids, because they allow to adjust the velocity discretization without changing the physical meaning of the velocity coordinates. Consequently, in this manner, we could preserve the mathematical model (the governing equations did not change) and easily port the well-established code, which had been originally written for regular grids. Moreover, preserving the governing equations allowed reusing already highly optimized existing routines, without hindering future physical extensions.

- *Implementation aspects:* We applied block-structured grids to implement an initial value solver for linear and nonlinear gyrokinetic simulations. Most of implementation specific to block-structured grids is concentrated in specially developed data structures. These data-structures store computational and ghost grids related data, and are also responsible for interpolations and data exchanges between processes.

- *Numerical experiments:* We verified block-structured extensively in the context of both linear and nonlinear radially global gyrokinetic simulations:
  - For the linear tests, we compared the convergence plots depending on the number of grid points in the velocity directions, for both the growth rate and the real frequency of the dominant mode. These tests revealed that the block-structured grids converge faster and attain reliable results even for very coarse velocity resolutions, when the regular grids are rendered useless.

  - For the nonlinear tests, which generally are excessively computationally expensive for the convergence-like verification, we elaborated a special test. In this test, a block-structured grid is compared with a reference regular grid and two other alternative grids with the same number of computational nodes as the block-structured one. We could observe that only the block-structured grid achieved the same accuracy results as the reference regular counterpart, which typically necessitates much more grid points. Moreover, as the nonlinear runs with the alternative regular grids showed, the same accuracy results cannot be gained by a simple coarsening or a shortening of velocity ranges.

The theoretical investigations and numerical experiments showed that the block-structured grids possess several beneficial properties for the Eulerian gyrokinetic simulations, such as:

- Allowing a significant reduction in the number of computational nodes, without a loss of accuracy in the gyrokinetic simulations. Not only does this lead to a much smaller memory footprint, but it also reduces the storage necessary for the diagnostics data and checkpoints. The latter benefit is especially valuable for the nonlinear simulations, where the output has to be saved for many time steps to be able to analyze reliably quasi-stationary properties of the plasma turbulence.

- Achieving speedup values roughly proportional to the reduction in the number of computational nodes. Since the number of floating point operations in the gyrokinetic code is proportional to the number of computational nodes, reducing the number of grid points leads to the same theoretical speedup. This aspect was confirmed by numerical tests: For example, in the linear runs, the reduction in the number of grid points was 2.36, whereas the speedup was 2.03, which is a bit smaller due to the overheads in the treatment of the grid-block boundaries and the side-exchanges in the radial direction. In nonlinear simulations, however, the created overheads were outweighed by a cache effect, resulting in a superlinear speedup of 10.5 for a reduction in the number of grid points of 9.4.

- Yielding reliable results with very few computational nodes in the velocity directions, even when regular grids no longer perform.

- Preserving the original mathematical model, which allows an easy porting of code for block-structured grids into existing and well-established gyrokinetic codes.

- Simplifying code extensions for new physical phenomena by decoupling the code specific to block-structured grids in specially developed data structures.

- Allowing applications to other grid-based gyrokinetic codes, due to the generality of the proposed approach.

These benefits of block-structured grids open new possibilities for the GENE software package. To capitalize further on the developed method, the application area of this technique can be extended to include collisions between different species, which

can be simulated on different block-structured grids. A challenging numerical treatment of collisions between species with disparate thermal speeds is a recurring problem in plasma simulations (see [31, 33, 34] for different plasma simulation codes or [32] for GENE). The proposed grids can be applied to solve an orthogonal problem of strong spatial temperature variations in these demanding gyrokinetic simulations with collisions.

The main focus of this study was to introduce and apply block-structured grids to the Eulerian gyrokinetic code GENE, in order to make simulations with strong temperature variations feasible. Given that our method is minimally invasive, the good performance properties of the GENE implementation for regular grids were inherited by the block-structured grids. This was confirmed by the achieved speedups. Nevertheless, further performance enhancements are possible, for example, by exploring different implementations of the data structures for block-structured grids. A potential candidate could store only ghost grids and meta-data necessary for interpolation and parallelization in these data-structures. Such an approach would decrease the data movement in the current implementation and would thus make memory usage more efficient.

Finally, while the block-structured grids are already fully functional in the gyrokinetic code GENE, it is still necessary to make the new implementation available to a wide community of researchers using this software package. To this purpose, several steps need to be performed: smoothen the user interface for the grid construction and other related simulation settings, add support for the proposed grids in key diagnostics, include the grid description in the user manual, and make the proposed grids available in the release version of GENE.

A

PRE AND POST PROCESSING TOOLS

The description of a block-structured grid is provided to GENE via a set of parameters. These are specified in an input file called `parameters`, which is read by GENE before each simulation. In this file, the values for each option are grouped and specified in `namelists` according to the Fortran standard, for details on `Namelist I/O` see [138]. An example of a `namelist` group with block-structured grid related settings is listed below:

```
&bsgrid
 is_grid_adptv = .T.
 is_nv0_fixed = .T.
 is_numofvpoints_const = .T.
 is_numofwpoints_const = .T.
 is_gpg = .F.
 gavg = 'std'
 wx_style = 'contour'
 bbfdscheme = 'ic4th'
 opt_mthd = 'integral'
 nblks = 5
 vp_std = 3.0
 blk_mks_r = 0.125, 0.454, 0.507, 0.567, 0.667, 0.875
 lv_mks = 3.315, 2.665, 2.046, 1.598, 1.349, 0.973
 lw_mks = 14.445, 9.337, 5.504, 3.356, 2.392, 0.211
/
```

The currently available options of the `bsgrid namelist` include[1]:

- **`is_grid_adptv` [`bool`]**: use block-structured grids or not
- **`is_nv0_fixed` [`bool`]**: keep the provided `nv0` value or adjust it
- **`is_numofvpoints_const` [`bool`]**: if true, use the block-structured grids of the second type in the $x - v_{||}$ subspace; otherwise, use the block structured grids of the first type
- **`is_numofwpoints_const` [`bool`]**: if true, use the block-structured grids of the second type in the $x - \mu$ subspace is used; otherwise use the block-structured grids of the first type
- **`is_gpg` [`bool`]**: if true, set the matrix for gyrophase averaging of the distribution function equal to the gyro-matrix for fields
- **`gavg` [`str`]**: determines how to store the gyro-matrices for block-structured grids; currently, there are two options:
  1. **`'std'`** — the matrices are saved in the same arrays as for regular grids

1 The provided options may change in the future to accommodate the various needs of the users.

2. **'bnd'** — the matrices are saved by using an alternative format, where each matrix corresponding to a grid-block is saved in a separate band matrix storage, see Section 5.1

- **wx_style [str]**: defines how to compute the μ nodes for Gauss Laguerre nodes; currently, three styles are supported:
  1. **'lambda'** — the Gauss Laguerre rule is adjusted for each grid-block, taking the exponential factor optimal for the middle of the radial grid-block range
  2. **'contour'** — the Gauss Laguerre rule is first computed like in the 'lambda' case; then, all nodes and weights are rescaled, so that all μ grid points fit inside the desired contour of the simulation domain
  3. **'gaulag'** — like for **'contour'**, except that the Gauss Laguerre nodes and weights are rescaled using a Gauss Laguerre rule with a fixed number of nodes provided to the grid generator, see Appendix Section A.1
- **bbfdscheme [str]**: the finite difference scheme used on grid-block boundaries; currently, the following schemes are available:
  1. **'ic4th'** — the default choice, uses a central difference scheme specified for the inner nodes, in combination with a fourth order polynomial interpolation, see Table 3 and Table 4
  2. **'iclin'** — as for 'ic4th', only with a linear interpolation
  3. **'v4th'** — the first alternative scheme, presented in Table 3, in combination with a fourth order polynomial interpolation
  4. **'xv4th'** — the second alternative scheme, presented in Table 3, in combination with a fourth order polynomial interpolation
- **opt_mthd [str]**: determines how to compute grid-blocks ranges and positions; currently, there are two ways:
  1. **'integral'** — the step-shape approximation of a contour is obtained by minimizing the probability difference of the background distribution function inside the step-shape approximation and desired simulation domain, for more details see Appendix Section A.1
  2. **'sum'** — as for 'integral', except that the difference of probabilities (alternatively standard deviation or velocity values) at discrete radial locations is minimized
- **nblks [int]**: the number of blocks in the block-structured grid
- **vp_std [real]**: the number of standard deviations of the background distribution function in the $x - v_{\parallel}$ subspace used to compute the ranges in the parallel velocity direction; the ranges for the magnetic moment direction are computed by using vp_std, as described in Subsection 3.3.2; previously, two deprecated options prob_vx (confidence level in the parallel velocity direction) and prob_wx (confidence level in the magnetic moment direction) were used instead of vp_std; these options were inconvenient for setting the velocity ranges consistently

- **blk_mks_r [real *]**: the radial locations of boundaries for all grid-blocks; the first and last values correspond to the radial range of the simulation domain
- **lv_mks [real *]**: the parallel velocity ranges of all grid-blocks; the last value is not used and corresponds to the parallel velocity range of the last grid-block in the limit of infinitely many grid-blocks
- **lw_mks [real *]**: the magnetic moment ranges of all grid-blocks; the last value is not used and corresponds to the magnetic moment range of the last grid-block in the limit of infinitely many grid-blocks

These listed options are used not only as an input to GENE, but also for pre and processing tools. Furthermore, some of the options explained, namely, `wx_style`, `opt_mthd`, and `vp_std`, are used exclusively by the pre processing tool. These options are kept in the `namelist` to describe thoroughly how the grid was generated and to be able to reconstruct it.

The rest of this appendix is structured as follows: In Section A.1, a grid generator is described, providing the most important options defining the shape of the grid-blocks: `blk_mks_r`, `lv_mks`, and `lw_mks`. In spite of extensive capabilities of already existing diagnostic tools for GENE, we had to implement a separate post processing tool, which is aware of the structure of the proposed grids. The usage and features of the post processing tool are explained in Section A.2.

A.1 PRE PROCESSING TOOL

The pre processing tool or grid generator for the block-structured grids is represented by a command line tool written in the high-level programming language `Python`. Interpreted programming languages like `Python` allow a rapid and efficient prototyping of different conceptual grids before applying these grids in real simulations in GENE application, the latter being written in `Fortran`. Another benefit is keeping a high performance code like GENE free of additional implementations like pre or post processing tools, which are subject to very frequent changes. This also reduces the chances of introducing programming errors in the high priority code. Moreover, by using `Python`, we have access to scientific data visualization and plotting libraries, such as `Matplotlib` (see [139]) for 2D data and `Mayavi` (see [140]) for 3D data. These two libraries were used to produce most of the plots in the current work.

The purpose of the pre processing tool for block-structured grids is twofold: constructing the grids and theoretically estimating the grid's properties. The main code resides in the `bsgrids.py` script, which can be called from the command line. To have help messages printed on screen, we execute the **`python bsgrids.py --help`** command, which results in the following output

```
usage: bsgrid.py [-h] [--debug] {cons,anlz} ...

construct or analyze different types of block-structured grids

optional arguments:
  -h, --help   show this help message and exit
  --debug      turn on logging debug mode
```

```
available types of operations:
  choose an option

  {cons,anlz}  list of tools
```

As this output suggests, there are two main sub-commands available: `cons` and `anlz`, which correspond to the construction and analysis of the grid, respectively. Furthermore, there two optional arguments of general purpose: `--help` and `--debug`, where the `-help` argument causes the tool to output the previous message and exit, while the `--debug` argument turns on the debugging mode for an embedded logging of our tools. The logging option is very useful during development, since they provide thorough information on the generation process and help identify errors. All logs are saved in the `bsgrid.log` file.

The main function of the aforementioned script is the construction of the block-structured grid and is available through the `cons` sub-command. Like in the previous command line example, we can use the `--help` argument to see all the functionality available. Executing **`python bsgrids.py cons --help`** produces the following output:

```
usage: bsgrid.py cons [-h] [--plt] [--pdf PDF] [--tfi] [--nm] [--ginfo]
                      [--nblks NBLKS] [--vpstd VPSTD] [--nglpoints NGLPOINTS]
                      pfile

construct block-structured grid based on parameters file

positional arguments:
  pfile                 parameters file name

optional arguments:
  -h, --help            show this help message and exit
  --plt                 matplotlib plots on screen
  --pdf PDF             base name of publication quality plots in pdf format
  --tfi                 show tfi plots
  --nm                  do not show mesh
  --ginfo               provide grid info
  --nblks NBLKS         number of blocks in grid
  --vpstd VPSTD         range of vp in terms of std deviations
  --nglpoints NGLPOINTS
                        number of Gauss-Laguerre points for mu range
```

As the help message explains, the block-structured grid is generated based on a `parameters` file. The minimum command to construct the grid is **`python bsgrids.py cons parameters`**. After triggering this command, the `parameters` file is parsed to extract the options described in the introduction of Appendix A. Then, the desired contour of the simulation domain is approximated by a step-shaped contour and the result (`blk_mks_r`, `lv_mks`, and `lw_mks`) is written back to the `parameters` file.

The grid constructor has a number of optional arguments, which are subject to frequent changes. Two arguments `--nblks` and `--vpstd` are duplicating the corresponding options `nblks` and `vp_std` from the `bsgrid namelist`. These arguments are often tuned during the adjustment process of block-structured grids. Therefore, they are introduced for convenience to avoid modifying the `parameters` file each time.

Two other important arguments are `--plt` and `--pdf`. The first argument `--plt` turns on the visual feedback for the grid construction. Examples of triggered outputs are shown in Figure 18, Figure 19, etc. The second argument `--pdf` allows to save the visualization of resulting grids in a publication-quality portable document format. The rest of the arguments are part of the current experimental configurations and are self-explanatory.

Next, we briefly explain two options of the grid construction specified by `opt_mthd`: `'integral'` and `'sum'`. The ideas of these two methods are graphically represented in Figure 55.

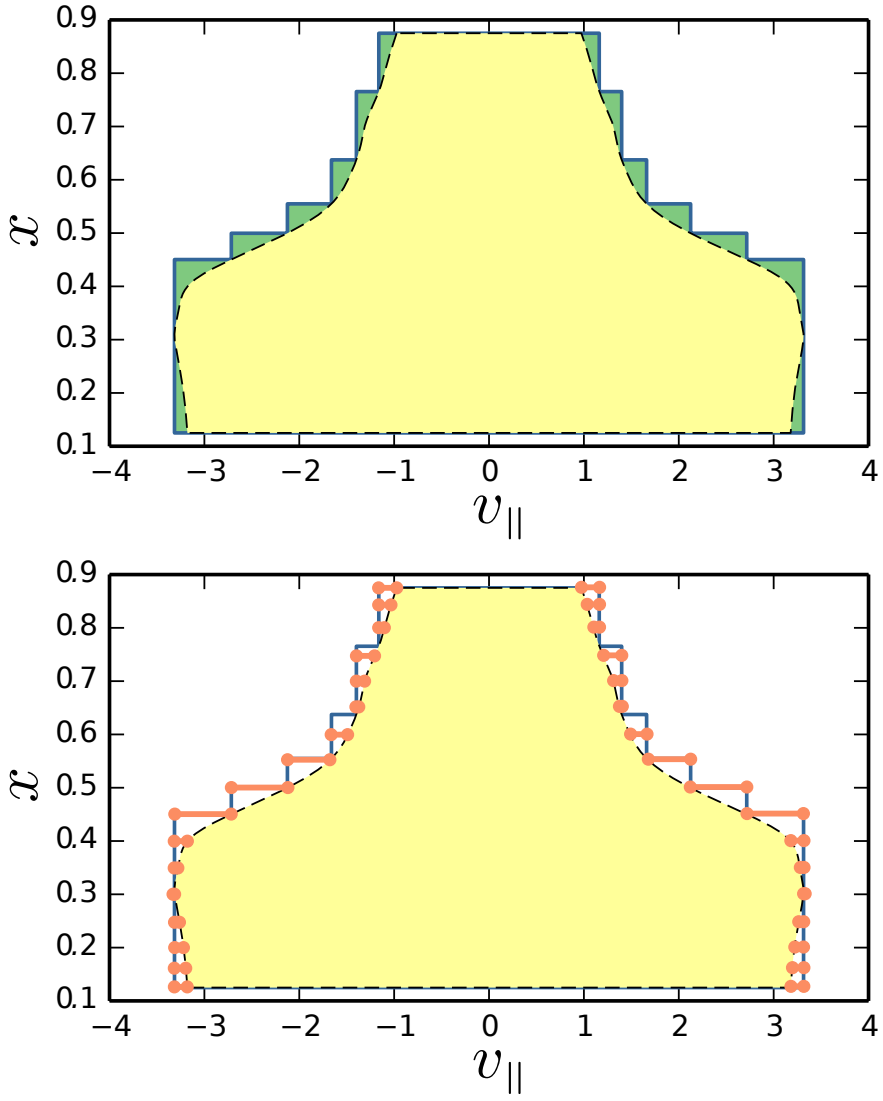


Figure 55: Illustrations of two methods for finding a step-shaped approximation of the simulation domain: `'integral'` (top) and `'sum'` (bottom). In the `'integral'` method, the probability in the green area (difference between step-shaped and precise contours) is minimized. In the `'sum'` method, the sum of differences in confidence levels (or other measures) between the step-shaped and precise contours is minimized; the differences are computed for the locations marked by orange dots and connected by lines.

The `'integral'` method minimizes the probability of the background distribution function in the area enclosed between the step-shaped approximation and the exact contours of the desired simulation domain. In Figure 55 (top), the corresponding area is marked in green. The name of the method stems from the fact that we have to compute integrals to evaluate the aforementioned probability. This minimization procedure might take some time (around one minute), especially when the exact location of the desired domain contour is known only at discrete points (experimental radial profiles) and we have to interpolate to get a smooth representation. This issue is insignificant when a grid is constructed just once, but becomes quite tedious when many different configurations have to be tried before choosing the most appropriate one.

To improve the performance of the grid generation, we implement a faster version, which is triggered by choosing the `'sum'` option. In this case, instead of evaluating integrals, we compute the sum of the confidence level (or of other quantities like parallel velocity values) differences between the step-shaped and exact contours at fixed radial distances. This sum is minimized to obtain the optimal step-like shape. The set of fixed radial distances is given by experimental density and temperature profiles.

For analytic profiles, the `'integral'` method is already fast and can be used by default. Otherwise, we take 32 equidistant points (by default) on the radial range for the `'sum'` method. The `'sum'` method is schematically represented in Figure 55 (bottom), where the set of the points at which the differences are computed are marked by orange dots and connected by lines.

The help message of the second sub-command `anlz` in the `bsgrid.py` script is similar to the help message of the `cons` and is given by

```
usage: bsgrid.py anlz [-h] [--plt] [--pdf PDF] [--tfi] [--nm] [--ginfo]
                      [--nv0] [--profiles]
                      pfile

analyze already constructed block-structured grid

positional arguments:
  pfile       parameters file name

optional arguments:
  -h, --help  show this help message and exit
  --plt       matplotlib plots on screen
  --pdf PDF   base name of publication quality plots in pdf format
  --tfi       show tfi plots
  --nm        do not show mesh
  --ginfo     provide grid info
  --nv0       print the number of nv0 points in final grid
  --profiles  output density and temperature profiles with their gradients
```

The purpose of this sub-command is to provide information on already generated block-structured grids, which are described in the `parameters` files. The script is capable of visualizing radial profiles, block-structured grids in different subspaces, computing the final number of the grid points, etc.

The analysis and visualization of the perturbed part of the distribution function obtained with the block-structured grid is realized in a separate post processing tool, which is presented in the following section.

A.2 POST PROCESSING TOOL

GENE comes with several post processing tools such as `gplot` and `vm_diag.sav`. The first tool provides a `gnuplot` based visualization (see [141]) of spatially averaged normalized fluctuating quantities, which are detailed in [130]. The second tool comprises a collection of visualization routines for GENE generated data, such as fields, moments, velocity space, or `nrg` file data. This tool is written in IDL[2] (see [142]), which is able to run under the free IDL Virtual Machine without a license. For details on how to use these tools, we refer to the GENE Manual [130].

All aforementioned tools analyze outputs files, which are generated by different diagnostic routines implemented in GENE. The GENE diagnostic routines are implemented primarily for the regular grids. Therefore, to be able to use the IDL post processing tool, we adjusted some diagnostic routines to the block-structured grids. The complete revision of all diagnostics for the block-structured grids is part of future work.

During the development and verification phases of the newly introduced grids, we need to compare or analyze not only (completely or partly averaged-out) quantities in the phase space, but also the raw data, such as the fluctuating part of the distribution function. For this purpose, we implemented a post processing tool, which is aware of the block-structured grids. This tool is capable of parsing and analyzing `checkpoint` and `g1.dat`[3] files.

The post processing routines for block-structured grids can be executed with the python script `parse_bincp.py`. A help command **`python parse_bincp.py --help`** prints the following message:

```
usage: parse_bincp.py [-h] [--debug] {shot,anim,avg,L2,gqtest,rsc} ...

parse and visualize raw-binary gene g1 data files

optional arguments:
  -h, --help            show this help message and exit
  --debug               turn on logging debug mode

available analysis types:
  choose how to post-process your binary checkpoint

  {shot,anim,avg,L2,gqtest,rsc}
                        list of tools
```

This message provides six available sub-commands:

- **`shot`**: parses a `checkpoint` file and produces contour plots of the absolute value of the fluctuating distribution function in different subspaces; this sub-command can also generate `vtk` output files for further visualization; for the `vtk` file format description, we refer to [143, 144]
- **`anim`**: parses `g1.dat` files to produce plots showing the time evolution of the fluctuating part of the distribution function in different subspaces; also supports `vtk` output

2 Interactive Data Language

3 These files contain ensembles of checkpoints at different time steps.

- **`avg`**: parses `g1.dat` files to produce plots showing the time-averaged absolute values of the fluctuating part of the distribution function in different subspaces; also supports `vtk` output (this sub-command was used in combination with the Paraview data analysis and visualization application (see [145, 146]) to produce plots over line in Chapter 6)
- **`L2`**: an experimental sub-command to compute the Euclidean distance between the perturbed parts of the distribution functions obtained with two different grids
- **`gqtest`**: an experimental sub-command, which was applied primarily to prototype Gauss quadrature rules in the magnetic moment direction by using data from real simulations
- **`rsc`**: in case of a crash or when it is necessary to restart a simulation from a certain time moment, this sub-command is capable of retrieving a `checkpoint` from a `g1.dat` file.

The optional arguments of all presented sub-commands are similar and can be listed by choosing `--help` and a corresponding sub-command. For example, the command **`python parse_bincp.py avg --help`** prints the following self-explanatory help message:

```
usage: parse_bincp.py avg [-h] [--norm]
                          [--fixIJKLMN FIXIJKLMN FIXIJKLMN FIXIJKLMN FIXIJKLMN
                              FIXIJKLMN FIXIJKLMN]
                          [--subd SUBD] [--spec SPEC] [--allKy] [--vtu VTU]
                          [--plt] [--pdf PDF] [--cmap CMAP]
                          [--times TIMES TIMES] [--heat]
                          pfile g1files [g1files ...]

parse g1.dat to produce a vtk file with the time averaged distribution
function on the corresponding regular grid

positional arguments:
  pfile                 parameters file name
  g1files               file names with g1 data

optional arguments:
  -h, --help            show this help message and exit
  --norm                normalize g1 data to maximum 1
  --fixIJKLMN FIXIJKLMN FIXIJKLMN FIXIJKLMN FIXIJKLMN FIXIJKLMN
                        specifies fixed indices for x, ky, z, vp, w, n; the
                        list contains two negative indices representing which
                        cross-section should be post-processed; (x,y) <- sorted
                        from smaller to bigger negative indices;
                        example: -1 0 nz0/2 -2 0 0 -> (x,y) fixed = (vp, x)
  --subd SUBD           specifies which subdomain to parse: vx, wx, ...
  --spec SPEC           specifies index of species
  --allKy               make analysis for all ky values
  --vtu VTU             name base of vtk output files
  --plt                 matplotlib plots on screen
  --pdf PDF             name of publication quality plots in pdf format
  --cmap CMAP           matplotlib color map for output plots
```

```
--times TIMES TIMES   time interval to compute the average g1
--heat                visualize g1*vp**2 instead of the average absolute
                      value of g1
```

Python provides a lot of useful packages to efficiently parse and analyze the data. However, to be able to use external applications, we introduced the `--vtk` option for the corresponding output. The `vtk` files can be analyzed by different applications. An example of visualizing the absolute value of the fluctuating part of the distribution function with Paraview is shown in Figure 56. Paraview also comprises several filters

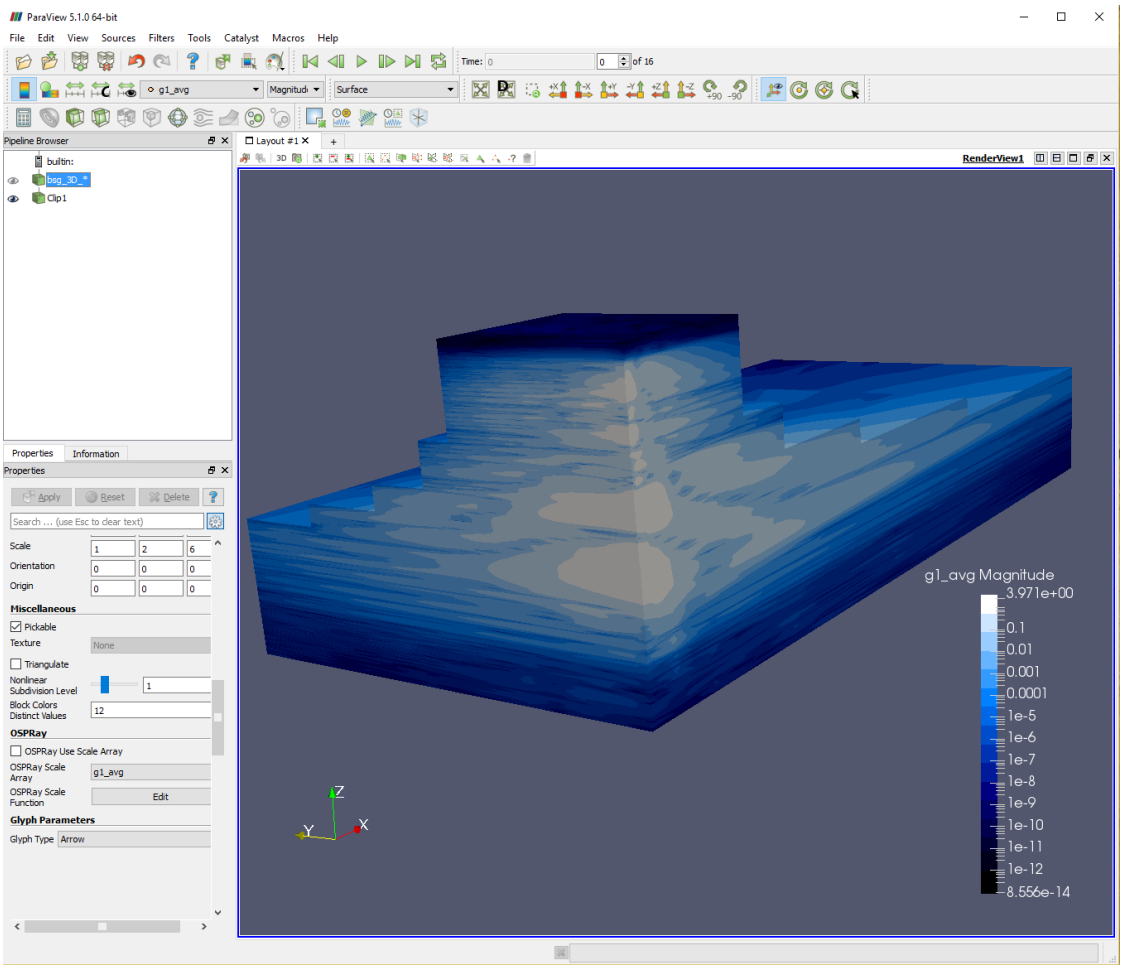


Figure 56: An example of a Paraview visualization of the fluctuating part of the distribution function in the $x - v_{||} - \mu$ subspace. The radial direction x corresponds to the Z axis in the figure, the parallel velocity $v_{||}$ corresponds to the Y axis, and the magnetic moment μ corresponds to the X axis.

for data analysis and visualization, which can be applied to our output files.

Our post processing tool was used for simple data analysis and visualization. Typical testing scenarios involved examining averaged quantities characteristic of quasi-stationary states. Nevertheless, by computing averages of `checkpoint` samples stored in `g1.dat` files, potentially interesting information may be lost. Furthermore, average values are not always representative of the sample behavior. For instance, for cases with more than one quasi-stationary state, the average does not adequately represent

any of the states. In such cases, other methods might be more appropriate to characterize the complex behavior, such as modelling samples with mixture models and summarizing each mixture component by its average. Such techniques have been applied successfully for ensemble data[4], to provide a proper characterization of potentially multimodal distributions (see, for instance, [147, 148]). Going beyond standard average analysis is subject of future work.

4 An ensemble is a collection of data sets, where each data set (ensemble member) has been obtained by running the same simulation, but with slightly different input parameters.

B

SOFTWARE DEVELOPMENT ASPECTS

GENE comes with an extensive suite of coarse-grained (or integration) testing, designed for a large portion of its implementation. This test suite comprises a set of different physical scenarios with already validated simulation results. The corresponding results were obtained by using regular grids. Therefore, the tests serve primarily to check whether an optimization of the original code has not introduced any errors.

The block-structured grids of the first type are also developed by using the coarse-grained tests. The main idea of these tests relies on comparing the results obtained with the block-structured grids against those obtained with the regular counterparts. This comparison is straightforward, because each block-structured grid of the first type is constructed by removing computational nodes from a certain region of the corresponding regular grid. Consequently, a regular reference grid is inherently available.

During integration tests, simulations are performed simultaneously with both block-structured and regular grids. Grid-based data from these two simulations, such as the fluctuating part of the distribution function or the right hand side of the governing equation, is analyzed after selected operations, comparing grid point against grid point. The selected operations include, for instance, the whole one time step iteration or that after specific terms of the governing equations are computed. It should be noted that, after each time step iteration, we have to nullify the portion of the grid-based data in the regular grid that is outside the step-shaped contour of the block-structured grid. This is necessary in order to ensure that both grids lead to the identical computations. If the comparison detects non-identical results, the simulation terminates with a printed message indicating the place where the test failed. This eases the process of identifying and correcting introduced errors. Furthermore, this approach does not require a complete implementation of the block-structured grid of the first type.

Unfortunately, this method based on integration tests does not work straightforwardly for the block-structured grids of the second type. This happens because there is no regular grid with grid points having the same locations as the block-structured grid of the second type (unless the grid has only one block). Consequently, the grid-based data comparison at each computational node is no longer possible. In this case, we verify numerical results by applying convergence tests for linear runs (presented in Section 6.1), and comparison of observables and plot over lines for nonlinear runs (presented in Section 6.2). This kind of verification is done only if the implementation of the block-structured grids of the second type is fully functional.

To mitigate the number of errors in the first fully functional implementation of the block-structured grids of the second type, we follow the principles of test driven development (for general details see [149], while, for an application pertaining to CSE[1] software development, we refer the reader to [150, 151]).

Porting the block-structured grids to GENE introduces some modifications of the original code and also requires additional implementations, such as special data structures for grid-based data. As the original code is not covered by unit tests, we apply

1 Computational Science and Engineering

techniques described in [152] to introduce changes. One important requirement for testing the proposed grids is the support of distributed memory parallelization. This introduces an additional requirement of supporting parallel routines on a unit testing platform. At the moment of choice, there was only one freely available unit testing library targeting parallel Fortran implementations — pFUnit, for details and usage see [153, 154].

The test driven development implies short development cycles, roughly represented by the following course of action:

1. Write a unit test for a new (not implemented yet) feature.
2. Run the test and validate that the test harness is working correctly (old tests pass, new introduced fail).
3. Implement the feature.
4. Run the test.
5. Refactor the code.

During the main steps of such development cycles (adding tests and implementing new functionality), we frequently have to introduce new dependencies between various Fortran modules. As the current default build tool is Make, these dependencies require hand modifications of the `files.mk` file, which is part of Make build scripts and maintains relations between GENE source files.

To avoid frequent modifications of the Make build scripts during the short development cycles, we use SCons software construction tool (for details, see [155–157]), which automatically analyzes the source files for dependencies. Furthermore, the configuration scripts of this construction tool are Python scripts, which grants access to a general purpose programming language. This could be beneficial for allowing the construction tool to interact with the GENE launcher, which is written in Python, see [44].

When the implementation is ready for a fully functional gyrokinetic simulation, we update the Make build scripts, which support various high performance computing hardware.

The integration and unit tests are built as separate applications. This is reflected in a directory tree relevant for the building with SCons, which is shown in Figure 57. In this diagram, *`SConstruct`* is a top-level configuration. The integration and unit source files are located in **`itests`** and **`utests`** directories, separately from the GENE source files in **`src`**. These directories contain *`SConscript`* files, which are responsible for configuring the build of the corresponding source files. Python configuration scripts for different compilers and hardware reside in the **`scons_machines`** directory.

To conclude, the application of the construction tool and test platform presented in this appendix considerably reduces the time necessary for implementing block-structured grids. Furthermore, the test driven development approach help us achieve a simple design of our code, because even data produced on the lowest level has to be in a format suitable for testing (comparison with the expected values).

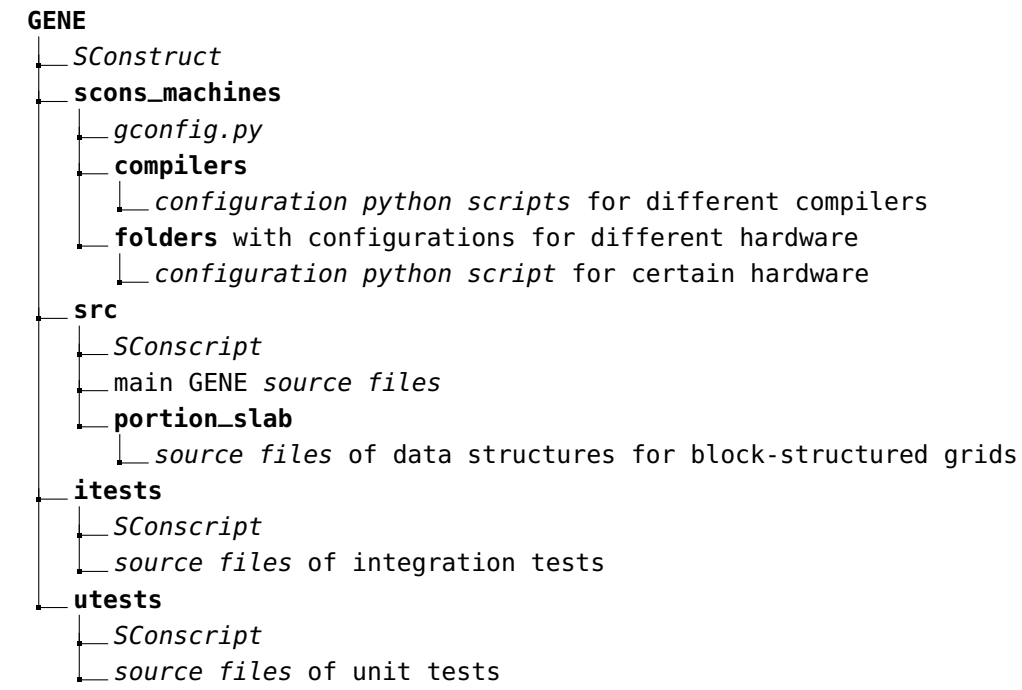


Figure 57: The part of the GENE directory tree relevant for the software construction with SCons. The directory names are written in **bold** and the file names are written in *italic*.

BIBLIOGRAPHY

[1] J. Dahlburg et al. "Fusion Simulation Project: Integrated Simulation and Optimization of Magnetic Fusion Systems." In: *Journal of Fusion Energy* 20.4 (2001), pp. 135–196.

[2] W. M. Tang. "Scientific and computational challenges of the fusion simulation project (FSP)." In: *Journal of Physics: Conference Series* 125.1 (2008).

[3] A. Bécoulet, P. Strand, H. Wilson, M. Romanelli, L. G. Eriksson, and The contributors to the European Task Force on Integrated Modelling Activity. "The way towards thermonuclear fusion simulators." In: *Computer Physics Communications* 177.1-2 (2007). Proceedings of the Conference on Computational Physics, pp. 55–59.

[4] G.L. Falchetto et al. "The European Integrated Tokamak Modelling (ITM) effort: achievements and first physics results." In: *Nuclear Fusion* 54.4 (2014).

[5] A. Fukuyama and M. Yagi. "Burning Plasma Simulation Initiative and Its Recent Progress." In: *Journal of Plasma and Fusion Research* 81.10 (2005), pp. 747–754.

[6] R. L. Reid et al. *ETR/ITER Systems Code*. research report. Available at http://web.ornl.gov/info/reports/1988/3445602756774.pdf. Oak-Ridge: Oak Ridge National Laboratory, Fusion Engineering Design Center, 1988.

[7] T. Hartmann. "Development of a modular systems code to analyse the implications of physics assumptions on the design of a demonstration fusion power plant." PhD thesis. Technische Universität München, 2013.

[8] M. Kovari, R. Kemp, H. Lux, P. Knight, J. Morris, and D. J. Ward. ""PROCESS": A systems code for fusion power plants – Part 1: Physics." In: *Fusion Engineering and Design* 89.12 (2014), pp. 3054–3069.

[9] M. Kovari, F. Fox, C. Harrington, R. Kembleton, P. Knight, H. Lux, and J. Morris. ""PROCESS": A systems code for fusion power plants — Part 2: Engineering." In: *Fusion Engineering and Design* 104 (2016), pp. 9–20.

[10] L. L. Lao, H. St. John, R. D. Stambaugh, A. G. Kellman, and W. Pfeiffer. "Reconstruction of current profile parameters and plasma shapes in tokamaks." In: *Nuclear Fusion* 25.11 (1985), p. 1611.

[11] H. Lütjens, A. Bondeson, and O. Sauter. "The CHEASE code for toroidal MHD equilibria." In: *Computer Physics Communications* 97.3 (1996), pp. 219–260.

[12] J. Blum, C. Boulbe, and B. Faugeras. "Reconstruction of the equilibrium of the plasma in a Tokamak and identification of the current density profile in real time." In: *Journal of Computational Physics* 231.3 (2012). Special Issue: Computational Plasma Physics, pp. 960–980.

[13] P. Hertout, C. Boulbe, E. Nardon, J. Blum, S. Brémond, J. Bucalossi, B. Faugeras, V. Grandgirard, and P. Moreau. "The CEDRES++ equilibrium code and its application to ITER, JT-60SA and Tore Supra." In: *Fusion Engineering and Design* 86.6–8 (2011). Proceedings of the 26th Symposium of Fusion Technology (SOFT-26), pp. 1045–1048.

[14] L. Degtyarev, A. Martynov, S. Medvedev, F. Troyon, L. Villard, and R. Gruber. "The KINX ideal MHD stability code for axisymmetric plasmas with separatrix." In: *Computer Physics Communications* 103.1 (1997), pp. 10–27.

[15] A. B. Mikhailovskii, G. T. A. Huysmans, W. O. K. Kerner, and S. E. Sharapov. "Optimization of computational MHD normal-mode analysis for tokamaks." In: *Plasma Physics Reports* 23 (Oct. 1997), pp. 844–857.

[16] E. Strumberger, S. Günter, P. Merkel, and C. Tichmann. "Linear stability studies including resistive wall effects with the CASTOR/STARWALL code." In: *Journal of Physics: Conference Series* 561.1 (2014), p. 012016.

[17] G. Cenacchi and A. Taroni. *Jetto a free boundary plasma transport code*. research report. Available at https://inis.iaea.org/search/search.aspx?orig_q=RN:19097143. Italy, 1988.

[18] G. Pereverzev and P. N. Yushmanov. *ASTRA Automated System for TRansport Analysis in a Tokamak*. research report. Available at http://w3.pppl.gov/~hammett/work/2009/Astra_ocr.pdf. Garching, Germany: Max-Planck-Institute für Plasmaphysic, 2002.

[19] W. Park, E. V. Belova, G. Y. Fu, X. Z. Tang, H. R. Strauss, and L. E. Sugiyama. "Plasma simulation studies using multilevel physics models." In: *Physics of Plasmas* 6.5 (1999), pp. 1796–1803.

[20] G.T.A. Huysmans and O. Czarny. "MHD stability in X-point geometry: simulation of ELMs." In: *Nuclear Fusion* 47.7 (2007), p. 659.

[21] H. Lütjens and J. F. Luciani. "The XTOR code for nonlinear 3D simulations of MHD instabilities in tokamak plasmas." In: *Journal of Computational Physics* 227.14 (2008), pp. 6944–6966.

[22] J.F. Artaud et al. "The CRONOS suite of codes for integrated tokamak modelling." In: *Nuclear Fusion* 50.4 (2010), p. 043001.

[23] D.P. Coster, V. Basiuk, G. Pereverzev, D. Kalupin, R. Zagorksi, R. Stankiewicz, P. Huynh, F. Imbeaux, and Members of the Task Force on Integrated Tokamak Modelling. "The European Transport Solver." In: *IEEE Transactions on Plasma Science* 38.9 (Sept. 2010), pp. 2085–2092.

[24] X. Garbet, Y. Idomura, L. Villard, and T. H. Watanabe. "Gyrokinetic simulations of turbulent transport." In: *Nuclear Fusion* 50.4 (2010), pp. 1–30.

[25] J. A. Krommes. "The Gyrokinetic Description of Microturbulence in Magnetized Plasmas." In: *Annual Review of Fluid Mechanics* 44.1 (2012), pp. 175–201.

[26] Y. Peysson, J. Decker, and L. Morini. "A versatile ray-tracing code for studying rf wave propagation in toroidal magnetized plasmas." In: *Plasma Physics and Controlled Fusion* 54.4 (2012), p. 045003.

[27] N.B. Marushchenko, Y. Turkin, and H. Maassberg. "Ray-tracing code TRAVIS for ECR heating, EC current drive and ECE diagnostic." In: *Computer Physics Communications* 185.1 (2014), pp. 165–176.

[28] Yu.V. Petrov and R.W. Harvey. "A fully-neoclassical finite-orbit-width version of the CQL3D Fokker–Planck code." In: *Plasma Physics and Controlled Fusion* 58.11 (2016), p. 115001.

[29] A. Loarte et al. "Chapter 4: Power and particle control." In: *Nuclear Fusion* 47.6 (2007), S203.

[30] A. J. Brizard and T. S. Hahm. "Foundations of nonlinear gyrokinetic theory." In: *Reviews of Modern Physics* 79.421 (2007), pp. 421–468.

[31] O. Larroche. "An efficient explicit numerical scheme for diffusion-type equations with a highly inhomogeneous and highly anisotropic diffusion tensor." In: *Journal of Computational Physics* 223.1 (2007), pp. 436–450.

[32] H. Doerk. "Gyrokinetic Simulation of Microtearing Turbulence." PhD thesis. Fakultät für Naturwissenschaften der Universität Ulm, 2012.

[33] B. E. Peigney, O. Larroche, and V. Tikhonchuk. "Fokker-Planck kinetic modeling of suprathermal α-particles in a fusion plasma." In: *Journal of Computational Physics* 278 (2014), pp. 416–444.

[34] W. T. Taitano, L. Chacón, and A. N. Simakov. "An adaptive, conservative oD-2V multispecies Rosenbluth-Fokker-Planck solver for arbitrarily disparate mass and temperature regimes." In: *Journal of Computational Physics* (2016).

[35] F. Jenko, W. Dorland, M. Kotschenreuther, and B. N. Rogers. "Electron temperature gradient driven turbulence." In: *Physics of Plasmas* 7.5 (2000), pp. 1904–1910.

[36] T. Görler, X. Lapillonne, T. Brunner, T. Dannert, F. Jenko, F. Merz, and D. Told. "The global version of the gyrokinetic turbulence code GENE." In: *Journal of Computational Physics* 230.18 (2011), pp. 7053–7071.

[37] F. Jenko and The GENE development team. *Gyrokinetic Electromagnetic Numerical Experiment*. http://genecode.org. Accessed: 2016-10-17.

[38] D. Jarema, H. J. Bungartz, T. Görler, F. Jenko, T. Neckel, and D. Told. "Block-structured grids for Eulerian gyrokinetic simulations." In: *Computer Physics Communications* 198 (2016), pp. 105–117.

[39] D. Jarema, H. J. Bungartz, T. Görler, F. Jenko, T. Neckel, and D. Told. "Block-Structured Grids in Full Velocity Space for Eulerian Gyrokinetic Simulations." In: *Computer Physics Communications* 215 (2017), pp. 49–62.

[40] F. F. Chen. *Introduction to Plasma Physics and Controlled Fusion*. 2nd ed. Vol. 1. New York: Plenum Press, 1984.

[41] K. Nishikawa and M. Wakatani. *Plasma physics - Basic Theory with Fusion Applications*. 2nd ed. Berlin: Springer Verlag, 1993.

[42] *The ITER Tokamak Magnets*. https://www.iter.org/mach/Magnets. Accessed: 2016-11-17.

[43] T. Görler. "Multiscale Effects in Plasma Microturbulence." PhD thesis. Fakultät für Naturwissenschaften der Universität Ulm, 2009.

[44] D. Told. "Gyrokinetic Microturbulence in Transport Barriers." PhD thesis. Fakultät für Naturwissenschaften der Universität Ulm, 2012.

[45] J. R. Cary and R. G. Littlejohn. "Noncanonical Hamiltonian mechanics and its application to magnetic field line flow." In: *Annals of Physics* 151 (1983), pp. 1–34.

[46] R. E. Denton and M. Kotschenreuther. "δf Algorithm." In: *Journal of Computational Physics* 119.2 (1995), pp. 283–294.

[47] Y. Chen and S. E. Parker. "A δf particle method for gyrokinetic simulations with kinetic electrons and electromagnetic perturbations." In: *Journal of Computational Physics* 189.2 (2003), pp. 463–475.

[48] G. W. Hammett and F. W. Perkins. "Fluid moment models for Landau damping with application to the ion-temperature-gradient instability." In: *Phys. Rev. Lett.* 64 (25 June 1990), pp. 3019–3022.

[49] A. Brizard. "Nonlinear gyrofluid description of turbulent magnetized plasmas." In: *Physics of Fluids B* 4.5 (1992), pp. 1213–1228.

[50] W. Dorland and G. W. Hammett. "Gyrofluid turbulence models with kinetic effects." In: *Physics of Fluids B* 5.3 (1993), pp. 812–835.

[51] G. W. Hammett, M. A. Beer, W. Dorland, S. C. Cowley, and S. A. Smith. "Developments in the gyrofluid approach to Tokamak turbulence simulations." In: *Plasma Physics and Controlled Fusion* 35.8 (1993), p. 973.

[52] M. A. Beer and G. W. Hammett. "Toroidal gyrofluid equations for simulations of tokamak turbulence." In: *Physics of Plasmas* 3.11 (1996), pp. 4046–4064.

[53] H. Doerk and F. Jenko. "Towards optimal explicit time-stepping schemes for the gyrokinetic equations." In: *Computer Physics Communications* 185.7 (2014), pp. 1938–1946.

[54] M. Kotschenreuther, G. Rewoldt, and W. M. Tang. "Comparison of initial value and eigenvalue codes for kinetic toroidal plasma instabilities." In: *Computer Physics Communications* 88.2–3 (1995), pp. 128–140.

[55] W. Dorland, F. Jenko, M. Kotschenreuther, and B. N. Rogers. "Electron Temperature Gradient Turbulence." In: *Phys. Rev. Lett.* 85 (26 Dec. 2000), pp. 5579–5582.

[56] The GS2 development team. *Gyrokinetic Simulations*. http://gyrokinetics.sourceforge.net. Accessed: 2016-11-03.

[57] R. E. Waltz, J. Candy, F. L. Hinton, C. Estrada-Mila, and J. E. Kinsey. "Advances in comprehensive gyrokinetic simulations of transport in tokamaks." In: *Nuclear Fusion* 45.7 (2005), p. 741.

[58] General Atomics. *Gyro - Theory Group - General Atomics*. https://fusion.gat.com/theory/Gyro. Accessed: 2016-11-03.

[59] T. H. Watanabe and H. Sugama. "Velocity-space structures of distribution function in toroidal ion temperature gradient turbulence." In: *Nuclear Fusion* 46.1 (2006), pp. 24–32.

[60] A. G. Peeters and D. Strintzi. "The effect of a uniform radial electric field on the toroidal ion temperature gradient mode." In: *Physics of Plasmas* 11.8 (2004), pp. 3748–3751.

[61] The GKW development team. *gkw / gkw / wiki / Home – Bitbucket*. https://bitbucket.org/gkw/gkw/wiki/Home. Accessed: 2016-11-03.

[62] Y. Idomura, H. Urano, N. Aiba, and S. Tokuda. "Study of ion turbulent transport and profile formations using global gyrokinetic full-f Vlasov simulation." In: *Nuclear Fusion* 49.6 (2009), p. 065029.

[63] R. Courant, K. Friedrichs, and H. Lewy. "On the Partial Difference Equations of Mathematical Physics." In: *IBM J. Res. Dev.* 11.2 (Mar. 1967), pp. 215–234.

[64] M. J. Pueschel, T. Dannert, and F. Jenko. "On the role of numerical dissipation in gyrokinetic Vlasov simulations of plasma microturbulence." In: *Computer Physics Communications* 181.8 (2010), pp. 1428–1437.

[65] C. K. Birdsall and A. B. Langdon. *Plasma physics via computer simulation*. Series in plasma physics. Originally published: New York ; London : McGraw-Hill, 1985. New York: Taylor & Francis, 2005.

[66] W. W. Lee. "Gyrokinetic approach in particle simulation." In: *Physics of Fluids* 26.2 (1983), pp. 556–562.

[67] Z. Lin, T. S. Hahm, W. W. Lee, W. M. Tang, and R. B. White. "Turbulent Transport Reduction by Zonal Flows: Massively Parallel Simulations." In: *Science* 281.5384 (1998), pp. 1835–1837.

[68] The GTC development team. *Gyrokinetic Toroidal Code GTC*. http://phoenix.ps.uci.edu/GTC/. Accessed: 2016-11-04.

[69] Y. Idomura, S. Tokuda, and Y. Kishimoto. "Global gyrokinetic simulation of ion temperature gradient driven turbulence in plasmas using a canonical Maxwellian distribution." In: *Nuclear Fusion* 43.4 (2003), p. 234.

[70] S. Jolliet, A. Bottino, P. Angelino, R. Hatzky, M. T. Tran, F. B. Mcmillan, O. Sauter, K. Appert, Y. Idomura, and L. Villard. "A global collisionless PIC code in magnetic coordinates." In: *Computer Physics Communications* 177.5 (2007), pp. 409–425.

[71] A. M. Dimits, T. J. Williams, J. A. Byers, and B. I. Cohen. "Scalings of Ion-Temperature-Gradient-Driven Anomalous Transport in Tokamaks." In: *Phys. Rev. Lett.* 77 (1 July 1996), pp. 71–74.

[72] W. X. Wang, T. S. Hahm, W. W. Lee, G. Rewoldt, J. Manickam, and W. M. Tang. "Nonlocal properties of gyrokinetic turbulence and the role of E×B flow shear." In: *Physics of Plasmas* 14.7, 072306 (2007).

[73] The GTS development team. *Gyrokinetic Tokamak Simulation code (GTS)*. http://theorycodes.pppl.wikispaces.net/GTS. Accessed: 2016-11-04.

[74] C. S. Chang, S. Ku, P. H. Diamond, Z. Lin, S. Parker, T. S. Hahm, and N. Samatova. "Compressed ion temperature gradient turbulence in diverted tokamak edgea)." In: *Physics of Plasmas* 16.5, 056108 (2009).

[75] The XGC1 development team. *Codes and Computing XGC1*. http://epsi.pppl.gov/computing/xgc-1. Accessed: 2016-11-04.

[76] The GEM development team. *Electromagnetic Gyrokinetic Turbulence Simulation*. http://cips.colorado.edu/Group/Simulation/Gem.php. Accessed: 2016-11-04.

[77] J. A. Heikkinen, T. P. Kiviniemi, T. Kurki-Suonio, A. G. Peeters, and S. K. Sipilä. "Particle Simulation of the Neoclassical Plasmas." In: *Journal of Computational Physics* 173.2 (2001), pp. 527–548.

[78] The ELMFIRE development team. *ELMFIRE*. http://physics.aalto.fi/en/groups/fusion/research/elmfire/. Accessed: 2016-11-04.

[79] S. E. Parker and W. W. Lee. "A fully nonlinear characteristic method for gyrokinetic simulation." In: *Physics of Fluids B* 5.1 (1993), pp. 77–86.

[80] R. Hatzky, A. Könies, and A. Mishchenko. "Electromagnetic gyrokinetic PIC simulation with an adjustable control variates method." In: *Journal of Computational Physics* 225.1 (2007), pp. 568–590.

[81] A. Y. Aydemir. "A unified Monte Carlo interpretation of particle simulations and applications to non-neutral plasmas." In: *Physics of Plasmas* 1.4 (1994), pp. 822–831.

[82] C. C. Kim and S. E. Parker. "Massively Parallel Three-Dimensional Toroidal Gyrokinetic Flux-Tube Turbulence Simulation." In: *Journal of Computational Physics* 161.2 (2000), pp. 589–604.

[83] R. Hatzky. "Domain cloning for a particle-in-cell (PIC) code on a cluster of symmetric-multiprocessor (SMP) computers." In: *Parallel Computing* 32.4 (2006), pp. 325–330.

[84] A. Staniforth and J. Côté. "Semi-Lagrangian Integration Schemes for Atmospheric Models - A Review." In: *Monthly Weather Review* 119.9 (1991), pp. 2206–2223.

[85] E. Sonnendrücker, J. Roche, P. Bertrand, and A. Ghizzo. "The Semi-Lagrangian Method for the Numerical Resolution of the Vlasov Equation." In: *Journal of Computational Physics* 149.2 (1999), pp. 201–220.

[86] G. Depret, X. Garbet, P. Bertrand, and A. Ghizzo. "Trapped-ion driven turbulence in tokamak plasmas." In: *Plasma Physics and Controlled Fusion* 42.9 (2000), p. 949.

[87] Y. Sarazin, V. Grandgirard, E. Fleurence, X. Garbet, Ph. Ghendrih, P. Bertrand, and G. Depret. "Kinetic features of interchange turbulence." In: *Plasma Physics and Controlled Fusion* 47.10 (2005), p. 1817.

[88] N. Crouseilles, T. Respaud, and E. Sonnendrücker. "A forward semi-Lagrangian method for the numerical solution of the Vlasov equation." In: *Computer Physics Communications* 180.10 (2009), pp. 1730–1745.

[89] N. Crouseilles, M. Mehrenberger, and E. Sonnendrücker. "Conservative semi-Lagrangian schemes for Vlasov equations." In: *Journal of Computational Physics* 229.6 (2010), pp. 1927–1953.

[90] C. Z. Cheng and G. Knorr. "The integration of the Vlasov equation in configuration space." In: *Journal of Computational Physics* 22.3 (1976), pp. 330–351.

[91] V. Grandgirard et al. "A drift-kinetic Semi-Lagrangian 4D code for ion turbulence simulation." In: *Journal of Computational Physics* 217.2 (2006), pp. 395–423.

[92] V. Grandgirard et al. "A 5D gyrokinetic full-global semi-Lagrangian code for flux-driven ion turbulence simulations." In: *Computer Physics Communication* 207 (2016), pp. 35–68.

[93] The GYSELA development team. *GYSELA5D: a code GYrokinetic SEmi-LAgrangian in 5D*. http://gyseladoc.gforge.inria.fr/. Accessed: 2016-11-06.

[94] H. Lederer, R. Tisma, R. Hatzky, A. Bottino, and F. Jenko. "Application enabling in DEISA: Petascaling of plasma turbulence codes." In: *Advances in Parallel Computing* 15 (2008), pp. 713–720.

[95] *Unified European Applications Benchmark Suite*. http://www.prace-ri.eu/ueabs/. Accessed: 2016-12-05.

[96] F. Merz. "Gyrokinetic Simulations of Multimode Plasma Turbulence." PhD thesis. Mathematisch-Naturwissenschaftliche Fakultät der Westfälischen Wilhelms-Universität Münster, 2008.

[97] M. A. Beer. "Gyrofluid Models of Turbulent Transport in Tokamaks." PhD thesis. Princeton University, 1995.

[98] J. Candy and R. E. Waltz. "An Eulerian gyrokinetic-Maxwell solver." In: *Journal of Computational Physics* 186.2 (2003), pp. 545–581.

[99] J. Candy and R. E. Waltz. "Velocity-space resolution, entropy production, and upwind dissipation in Eulerian gyrokinetic simulations." In: *Physics of Plasmas* 13.032310 (2006), pp. 1–11.

[100] M. Barnes, W. Dorland, and T. Tatsuno. "Resolving velocity space dynamics in continuum gyrokinetics." In: *Physics of Plasmas* 17.032106 (2010), pp. 1–13.

[101] J. Candy, C. Holland, R. E. Waltz, M. R. Fahey, and Belli E. "Tokamak profile prediction using direct gyrokinetic and neoclassical simulation." In: *Physics of Plasmas* 16.060704 (2009), pp. 1–4.

[102] M. Barnes, I. G. Abel, W. Dorland, T. Görler, G. W. Hammett, and F. Jenko. "Direct multiscale coupling of a transport code to gyrokinetic turbulence codes." In: *Physics of Plasmas* 17.056109 (2010), pp. 1–11.

[103] T. Dannert. "Gyrokinetische Simulation von Plasmaturbulenz mit gefangenen Teilchen und elektromagnetischen Effect." PhD thesis. Fakultät für Physik der Technischen Universität München, 2005.

[104] M. J. Püschel. "Electromagnetic Effects in Gyrokinetic Simulations of Plasma Turbulence." PhD thesis. Mathematisch-Naturwissenschaftlichen Fakultät der Westfälischen Wilhelms-Universität Münster, 2009.

[105] T. Hauff. "Transport of Energetic Particle in Turbulent Plasma." PhD thesis. Fakultät für Naturwissenschaften der Universität Ulm, 2009.

[106] X. Lapillonne. "Local and Global Eulerian Gyrokinetic Simulations of Microturbulence in Realistic Geometry with Applications to the TCV Tokamak." PhD thesis. École Polytechnique Fédérale de Lausanne, 2010.

[107] C. Kowitz. "Applying the Sparse Grid Combination Technique in Linear Gyrokinetics." PhD thesis. Technische Universität München, 2016.

[108] M. A. Beer, S. C. Cowley, and G. W. Hammett. "Field-aligned coordinates for nonlinear simulations of tokamak turbulence." In: *Physics of Plasmas* 2.7 (1995), pp. 2687–2700.

[109] W. D. D'haeseleer, W. N. G. Hitchon, J. D. Callen, and J. L. Shohet. *Flux Coordinates and Magnetic Field Structure: A Guide to a Fundamental Tool of Plasma Theory*. New York: Springer Verlag, 1991.

[110] P. Xanthopoulos and F. Jenko. "Clebsch-type coordinates for nonlinear gyrokinetics in generic toroidal configurations." In: *Physics of Plasmas* 13.9, 092301 (2006), pp. 1–10.

[111] M. D. Kruskal and R. M. Kulsrud. "Equilibrium of a Magnetically Confined Plasma in a Toroid." In: *Physics of Fluids* 1.4 (1958), pp. 265–274.

[112] X. Lapillonne, S. Brunner, T. Dannert, S. Jolliet, A. Marinoni, L. Villard, T. Görler, F. Jenko, and F. Merz. "Clarifications to the limitations of the $s - \alpha$ equilibrium model for gyrokinetic computations of turbulence." In: *Physics of Plasmas* 16.3, 032308 (2009), pp. 1–9.

[113] L. L. Lao, H. St. John, R.D. Stambaugh, and W. Pfeiffer. "Separation of $\bar{\beta}_p$ and l_i in tokamaks of non-circular cross-section." In: *Nuclear Fusion* 25.10 (1985), p. 1421.

[114] P. Xanthopoulos, W. A. Cooper, F. Jenko, Yu. Turkin, A. Runov, and J. Geiger. "A geometry interface for gyrokinetic microturbulence investigations in toroidal configurations." In: *Physics of Plasmas* 16.8, 082303 (2009).

[115] W. J. Gordon and C. A. Hall. "Construction of curvilinear co-ordinate systems and applications to mesh generation." In: *International Journal for Numerical Methods in Engineering* 7.4 (1973), pp. 461–477.

[116] W. J. Gordon and L. C. Thiel. "Transfinite mappings and their application to grid generation." In: *Applied Mathematics and Computation* 10 (1982), pp. 171–233.

[117] R. Haber, M. S. Shephard, J. F. Abel, R. H. Gallagher, and D. P. Greenberg. "A general two-dimensional, graphical finite element preprocessor utilizing discrete transfinite mappings." In: *International Journal for Numerical Methods in Engineering* 17.7 (1981), pp. 1015–1044.

[118] O. Haber and J. F. Abel. "Discrete transfinite mappings for the description and meshing of three-dimensional surfaces using interactive computer graphics." In: *International Journal for Numerical Methods in Engineering* 18.1 (1982), pp. 41–66.

[119] A. Arakawa. "Computational design for long-term numerical integration of the equations of fluid motion: Two-dimensional incompressible flow. Part I." In: *Journal of Computational Physics* 1.1 (1966), pp. 119–143.

[120] J. Candy and R. E. Waltz. "An Eulerian gyrokinetic-Maxwell solver." In: *Journal of Computational Physics* 186.2 (2003), pp. 545–581.

[121] G Chesshire and W. D. Henshaw. "Composite Overlapping Meshes for the Solution of Partial Differential Equations." In: *Journal of Computational Physics* 90 (1990), pp. 1–64.

[122] S. E. Sherer and J. N. Scott. "High-order compact finite-difference methods on general overset grids." In: *Journal of Computational Physics* 210 (2005), pp. 459–496.

[123] J. L. Vay, P. Colella, A. Friedman, D. P. Grote, P. McCorquodale, and D. B. Serafini. "Implementations of mesh refinement schemes for Particle-In-Cell plasma simulations." In: *Computer Physics Communications* 164.1–3 (2003), pp. 297–305.

[124] J. A. F. Hittinger and J. W. Banks. "Block-structured adaptive mesh refinement algorithms for Vlasov simulation." In: *Journal of Computational Physics* 241 (2013), pp. 118–140.

[125] P. Colella, M. R. Dorr, and D. D. Wake. "Numerical Solution of Plasma Fluid Equations Using Locally Refined Grids." In: *Journal of Computational Physics* 152 (1999), pp. 550–583.

[126] G. Tóth, Y. Ma, and T. I. Gombosi. "Hall magnetohydrodynamics on block-adaptive grids." In: *Journal of Computational Physics* 227 (2008), pp. 6967–6984.

[127] H. Frerichs, D. Reiter, Y. Feng, and D. Harting. "Block-structured grids in Lagrangian 3D edge plasma transport simulations." In: *Computer Physics Communications* 181 (2010), pp. 61–70.

[128] W. Gautschi. "Algorithm 726: ORTHPOL–a Package of Routines for Generating Orthogonal Polynomials and Gauss-type Quadrature Rules." In: *ACM Trans. Math. Softw.* 20.1 (Mar. 1994), pp. 21–62.

[129] W. Gander and J. Hřebíček. *Solving problems in scientific computing using Maple and MATLAB*. 3rd edition. Springer, 1997.

[130] GENE Development Team. *The Gyrokinetic Plasma Turbulence Code GENE: User Manual*. Aug. 23, 2013. 63 pp.

[131] S. Brunner, T. Dannert, X. Lapillonne, T. Görler, F. Jenko, F. Merz, and D. Told. *Developer's manual for the global version of the GENE code*. Aug. 31, 2015. 79 pp.

[132] E. Anderson et al. *LAPACK Users' Guide*. Third. Philadelphia, PA: Society for Industrial and Applied Mathematics, 1999.

[133] P.A Durbin and G Iaccarino. "An Approach to Local Refinement of Structured Grids." In: *Journal of Computational Physics* 181.2 (2002), pp. 639–653.

[134] F. Merz, C. Kowitz, E. Romero, J. E. Roman, and F. Jenko. "Multi-dimensional gyrokinetic parameter studies based on eigenvalue computations." In: *Computer Physics Communications* 183.4 (2012), pp. 922–930.

[135] C. Kowitz and M. Hegland. "The sparse grid combination technique for computing eigenvalues in linear gyrokinetics." In: *International Conference on Computational Science 2013*. Procedia Computer Science. Amsterdam: Elsevier, June 2013, pp. 449–458.

[136] D. Pflüger, H. J. Bungartz, M. Griebel, F. Jenko, T. Dannert, M. Heene, A. Parra Hinojosa, C. Kowitz, and P. Zaspel. "EXAHD: An Exa-Scalable Two-Level Sparse Grid Approach for Higher-Dimensional Problems in Plasma Physics and Beyond." In: *Euro-Par 2014: Parallel Processing Workshops*. Lecture Notes in Computer Science. Springer-Verlag, Sept. 2014.

[137] A. Parra Hinojosa, C. Kowitz, M. Heene, D. Pflüger, and H. J. Bungartz. "Towards a fault-tolerant, scalable implementation of GENE." In: *Proceedings of ICCE 2014*. Lecture Notes in Computational Science and Engineering. Accepted. Springer-Verlag, 2015.

[138] M. Metcalf, J. Reid, and M. Cohen. *Fortran 95/2003 Explained*. Oxford University Press, 2004.

[139] J. D. Hunter. "Matplotlib: A 2D graphics environment." In: *Computing In Science & Engineering* 9.3 (2007), pp. 90–95.

[140] P. Ramachandran and G. Varoquaux. "Mayavi: 3D Visualization of Scientific Data." In: *IEEE Computing in Science & Engineering* 13.2 (2011), pp. 40–51.

[141] T. Williams, C. Kelley, et al. *Gnuplot 5.0.5: an interactive plotting program*. http://gnuplot.sourceforge.net/. Oct. 2016.

[142] Exelis Visual Information Solutions. http://www.harrisgeospatial.com. Accessed: 2016-10-14. Boulder, Colorado.

[143] W. Schroeder, K. Martin, and B. Lorensen. *The Visualization Toolkit*. 4th ed. Kitware, 2006.

[144] L. S. Avila et al. *The VTK User's Guide*. 11th ed. Kitware, 2010.

[145] J. Ahrens, B. Geveci, and C. Law. *ParaView: An End-User Tool for Large Data Visualization, Visualization Handbook*. Elsevier, 2005.

[146] U. Ayachit. *The ParaView Guide: A Parallel Visualization Application*. Kitware, 2015.

[147] M. Jarema, I. Demir, J. Kehrer, and R. Westermann. "Comparative visual analysis of vector field ensembles." In: *Visual Analytics Science and Technology (VAST), 2015 IEEE Conference on*. IEEE. 2015, pp. 81–88.

[148] M. Jarema, J. Kehrer, and R. Westermann. "Comparative Visual Analysis of Transport Variability in Flow Ensembles." In: *Journal of WSCG* 24.1 (2016), pp. 25–34.

[149] K. Beck. *Test Driven Development: By Example*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2002.

[150] T. Clune and R. Rood. "Software Testing and Verification in Climate Model Development." In: *IEEE Software* 28.6 (Nov. 2011), pp. 49–55.

[151] T. Clune, M. Rilee, and D. Rouson. "Testing as an Essential Process for Developing and Maintaining Scientific Software." In: WSSSPE2. July 2014.

[152] M. Feathers. *Working Effectively with Legacy Code*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2004.

[153] M. Rilee and T. Clune. "Towards Test Driven Development for Computational Science with pFUnit." In: *Proceedings of the 2Nd International Workshop on Software Engineering for High Performance Computing in Computational Science and Engineering*. SE-HPCCSE '14. New Orleans, Louisiana: IEEE Press, 2014, pp. 20–27.

[154] The pFUnit development team. *Unit testing framework for Fortran with MPI extensions*. http://pfunit.sourceforge.net/.. Accessed: 2016-10-18.

[155] S. Knight. "Building Software with SCons." In: *Computing in Science and Engg.* 7.1 (Jan. 2005), pp. 79–88.

[156] S. Knight and the SCons Development Team. *SCons 2.5.0*. 2016. 185 pp.

[157] The SCons development team. *SCons: A software construction tool*. http://www.scons.org/. Accessed: 2016-10-18.