ABSTRACT

Title of dissertation: Advanced Lagrangian Simulation Algorithms for Magnetized Plasma Turbulence

Ingmar Bert Broemstrup,
Doctor of Philosophy, 2008

Dissertation directed by: Professor William Dorland
Department of Physics

Nonlinear processes in hot, magnetized plasma are notoriously difficult to understand without the use of numerical simulations. In recent decades, first principles, kinetic simulations have been widely and successfully used to study plasma turbulence and reconnection in weakly collisional systems. In this thesis, extensions of well-known, Lagrangian, particle-in-cell (PIC) simulation algorithms for problems such as these are derived and implemented. The algorithms are tested for multiple species (electrons and ions, with the physical mass ratio) in non-trivial magnetic geometry (cylindrical/toroidal). The advances presented here address two major shortcomings of conventional gyrokinetic PIC algorithms, with demonstrated excellent performance on large, parallel supercomputers. Although the gyrokinetic formalism rigorously describes the evolution of fluctuations which are small compared to a typical Larmor radius, most existing algorithms use low-order approximations of the gyroaveraging operator, and cannot accurately describe small scale fluctuations. The gyroaveraging algorithm presented here accurately and uniquely treats

a wide range of fluctuation scales, above and below the thermal gyroradius. The second shortcoming of traditional algorithms relates to the slow loss of accuracy that is associated with the build-up of noise. In this thesis, a PIC pitch-angle scattering collision operator is developed. This collision operator is physically motivated and controls the growth of noise without introducing non-physical dissipation. Basic tests of the new algorithms are presented in linear and nonlinear regimes, using one to thousands of processors simultaneously.

Advanced Lagrangian Simulation Algorithms for Magnetized Plasma Turbulence

by

Ingmar Bert Broemstrup

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2008

Advisory Committee:
Professor William Dorland, Chair/Advisor
Professor Thomas Antonsen, Jr.
Professor James Drake
Professor Ramani Duraiswami
Professor Parvez Guzdar
Professor Adil Hassam

Dedication

To my parents Marianne and Niels, with love

Table of Contents

List of Figures

Chapter 1

Gyrokinetic theory and simulation

1.1 Introduction

Turbulence, reconnection and other nonlinear plasma processes are important elements of many problems in plasma physics. Whether one is trying to understand the properties of the solar wind sweeping past the Earth or the confinement of thermonuclear plasma in laboratory experiments such as tokamaks, the critical role of nonlinear plasma processes usually cannot be ignored. The theoretical tool of choice for getting a conceptual understanding of nonlinear plasma processes is the simulation. This thesis describes extensions of well-known algorithms for plasma simulation. The specific algorithms presented are generalizations of widely-used gyrokinetic particle-in-cell algorithms.

Particle-in-cell (PIC) techniques were discovered and popularized by Dawson, [18] Buneman, [10] Hockney and Eastwood, [33] and Birdsall and Langdon. [6, 7] Kinetic simulation algorithms are important for problems for which the interparticle collisions are not frequent enough to enforce a Maxwellian distribution. When the

collisions are that fast, one prefers fluid models. In many plasmas, collisions are fast enough to keep the slowly evolving background distribution Maxwellian, but not the rapidly evolving fluctuations. For these plasmas, when sufficiently magnetized, gyrokinetic [1, 44] simulations are employed.

Gyrokinetic PIC simulations began with Lee in the 1980's, [44] and has flourished in the years since.[1] Kotschenreuther and Denton [19] and Dimits and Lee [21] developed δf algorithms for gyrokinetic PIC simulations. Kotschenreuther emphasized the low-noise properties of the δf scheme (to be discussed in detail below), while Dimits and Lee emphasized the favorable run-time performance. By the time the Kotschenreuther and Denton paper was published, it was clear that good conservation properties of the δf scheme were difficult to achieve, requiring large numbers of particles and expensive runs. Aydemir [2] discussed the problem of statistical noise in plasma particle simulations around the same time (1994). In the intervening years, the Lausanne group worked extensively on the problems of noise and conservation laws, in largely unpublished work. However, only recently have careful benchmarks led to a community consensus on the issue of the "growing weights" problem associated with δf algorithms.[48]

The heart of the matter of noise in GK δf simulations is this: each simulation particle represents the departure of the distribution function from a background Maxwellian with spatial gradients. This departure is identified as a particle "weight". As the simulation particles move up and down the gradient, their weights must grow (in absolute value), because the departure from the background that they

[1] Gyrokinetics will be defined and discussed below.

represent is proportional to the distance they are moved along the gradient. After a period of time, those particles whose weights have grown the largest begin to dominate the charge and current integrals for the sources in Maxwell's equations. It is as if the number of particles in the simulation were decreasing in time; clearly, beyond some point, if nothing can be done to control the RMS growth of the weights, statistical noise will swamp the signal, and the fluctuating electric and magnetic fields will no longer represent the coherent dynamics of interest.

The algebraic growth of the weights in time is not purely numerical in nature. In a real plasma, $\delta f(v)$ develops more and more structure as time increases, due to effects like Landau damping, or phase mixing. In the physical system, there are almost always sufficient numbers of collisions to bring about irreversibility by smoothing out these oscillatory structures at some scale in velocity space. The growing weights in a δf PIC simulation are the numerical manifestation of phase mixing.

Krommes [41] and Brunner, *et al.,* [9] realized that collisions were needed in δf PIC simulations to give irreversibility. Lee and Tang [45] recognized that without collisions, the weights would grow indefinitely in a δf simulation of an unstable plasma. The collision operators in wide use, however, such as the Monte Carlo operator of Dimits and Cohen [20], surprisingly do not limit the growth of weights. In fact, as pointed out by Brunner, *et al.*, these collision operators actually speed the growth of the weights! One can understand this by realizing that the Monte Carlo schemes (for example) do not change the weight, but rather the velocity of a simulation particle. This puts the particle onto a different trajectory, with (on

average) a correspondingly larger weight than before. More sophisticated collision operators have been proposed [41], but these schemes are generally of the class for which the reduction in weight is proportional to the weight itself, rather than directly to the presence of structure in $f(v)$. Parker and Chen have proposed a coarse-graining algorithm to reduce the average weights. However, their scheme cannot be easily related to a physical effect. It tends to be very dissipative for large number of particles and ineffective for small number of particles. More will be said about these schemes in Chapters 4 and 5. Recently, a scheme similar to ours was published by Hinton [31], but Hinton's scheme has not yet been implemented in any simulation code.

In this thesis, the problem of growing noise in δf PIC simulations is solved by employing a pitch-angle scattering collision operator in the velocity space. To evaluate $C(f) \sim \nu \; d^2 f/d\xi^2$ requires interpolating $f(v)$ at each spatial grid point, which is why it has generally been avoided. The algorithm presented here parallelizes very efficiently, however, making this interpolation an acceptable cost. This is expected to be particularly the case when long-time simulations are desired, or when the physical form of the collision operator is thought to be important – such as for the trapped-electron-mode instability in tokamaks. In any case, the collisional algorithm derived here is straightforward, physical, and is guaranteed to control the growth of the weights as long as enough simulation particles are used. Because our collision operator allows accurate long-time simulation of fast turbulence dynamics, it is a critical component of a useful multiscale simulation algorithm for kinetic plasma turbulence.

A second problem addressed in this thesis is a multiscale *spatial* issue. Gyrokinetic theory treats perturbations both larger and smaller than the thermal ion gyroradius rigorously. This is achieved with the employ of a non-local, integral gyroaveraging operator (a ring average in the plane perpendicular to the magnetic field). All existing GK PIC codes approximate this operator with a ring-shaped stencil.[44] Typically, a four-point stencil is used, and perturbation comparable to or smaller than the thermal gyroradius are treated inaccurately. In Chapter 2, we present an alternative gyroaveraging algorithm which uses the spectral form of the gyroaveraging operator. This form is accurate even for very small spatial fluctuations. Again, such an algorithm was thought to be prohibitively expensive, but our implementation is as fast as existing, low-accuracy schemes, particularly in the limit in which one believes short wavelength fluctuations are important, and should be resolved. Again, the canonical example is the trapped electron mode instability, whose perpendicular wavelength is associated more with the electron banana width than with the ion gyroradius. A less familiar example, but one which we explore in Chapter 6, is the entropy mode in a Z-pinch.[53, 54] Our algorithm faithfully reproduces the entropy mode instability even when $k_\perp \rho_i \gg 1$. We reiterate that our combined algorithm, which amounts to a re-imagining of the GK PIC family of algorithms, is efficient and parallelizes well. Parallelization is achieved without domain decomposition, which would make the spectral solver more expensive to evaluate.

In this chapter we introduce the gyrokinetic orderings [1, 26] and derive the gyrokinetic equation. We will start from the Boltzmann equation, and focus on

including collisional physics systematically. Our derivation follows that of Howes, *et al.* [34]

1.2 The gyrokinetic equation

The Boltzmann equation, which can be derived from the fundamental Liouville equation with the application of the BBGKY hierarchy, describes the evolution of the single particle distribution function in a weakly coupled plasma, which satisfies $n_{0_e}\lambda_{De}^3 \ll 1$. Here, n_{0_e} is the mean electron number density and λ_{D_e} is the electron Debye length. Upon identifying the acceleration experienced by species s with the electromagnetic (Lorentz) force divided by the mass m_s, one may write the Boltzmann equation as:

$$\frac{df_s}{dt} = \frac{\partial f_s}{\partial t} + \mathbf{v}\cdot\nabla f_s + \frac{q_s}{m_s}\left(-\nabla\phi - \frac{1}{c}\frac{\partial \mathbf{A}}{\partial t} + \frac{\mathbf{v}\times\mathbf{B}}{c}\right)\cdot\frac{\partial f_s}{\partial \mathbf{v}} = \left(\frac{\partial f_s}{\partial t}\right)_{coll}. \quad (1.1)$$

Note that the electric field is expressed in potential form, as is customary in the gyrokinetic literature. When the magnetic vector potential is used, the Coulomb gauge will be employed, so that $\nabla\cdot\mathbf{A} = 0$.

The gyrokinetic ordering describes low-frequency plasma motions in magnetized plasma, such that any dynamical frequency of interest ω satisfies $\omega \ll \Omega$, where Ω is the ion cyclotron frequency. Due to strong magnetization of the plasma the ion Larmor radius ρ_i is much smaller than the typical macroscopic length scale L of the plasma, so that

$$\rho_i \equiv \frac{v_{T_i}}{\Omega_i} \ll L, \quad \omega \ll \Omega_i.$$

The gyrokinetic ordering is carried out in powers of ϵ, given by

$$\epsilon \equiv \frac{\rho_i}{L} \ll 1.$$

For the timescale of the turbulent fluctuations in the system one assumes:

$$\omega \sim \frac{v_{T_i}}{L} \sim \mathcal{O}(\epsilon \Omega_i).$$

An additional timescale in the problem is the transport rate at which the equilibrium solution of the problem evolves over time.

$$t^{-1}_{transport} \sim \epsilon^2 \frac{v_{T_i}}{L} \sim \mathcal{O}(\epsilon^3 \Omega_i).$$

The distribution function f_s as well as the magnetic and electric fields $\mathbf{B}$ and $\mathbf{E}$ can be broken up into an equilibrium part and into perturbed parts δ which vary at the frequency ω. The subscript for the perturbed part indicates the order in ϵ of the perturbation. Note that from here on we are dropping the subscript s that indicates the species.

$$\begin{aligned} f(\mathbf{x}, \mathbf{v}, t) &= F_0(\mathbf{x}, \mathbf{v}, t) + \delta f_1(\mathbf{x}, \mathbf{v}, t) + \delta f_2(\mathbf{x}, \mathbf{v}, t) + \ldots, \\ \mathbf{B}(\mathbf{x}, t) &= \mathbf{B_0} + \delta \mathbf{B}(\mathbf{x}, t) = \mathbf{B_0} + \nabla \times \mathbf{A} \ , \\ \mathbf{E}(\mathbf{x}, t) &= \delta \mathbf{E}(\mathbf{x}, t) = -\nabla \phi - \frac{1}{c} \frac{\partial \mathbf{A}}{\partial t}. \end{aligned}$$

The following table contains all the gyrokinetic ordering assumptions:

TABLE 1

Gyrokinetic ordering assumptions

Slowly varying equilibrium	$\frac{1}{F_0}\frac{\partial F_0}{\partial t} \sim \mathcal{O}\left(\epsilon^2 \frac{v_{T_i}}{L}\right)$
Small amplitude fluctuations about the equilibrium	$\frac{\delta f_1}{F_0} \sim \frac{\delta B}{B_0} \sim \frac{\delta E}{(v_{T_i}/c)B_0} \sim \mathcal{O}(\epsilon)$
Medium time-scale variation of fluctuations	$\frac{1}{\delta f}\frac{\partial \delta f}{\partial t} \sim \frac{1}{\delta B}\frac{\partial \delta B}{\partial t} \sim \frac{1}{\delta E}\frac{\partial \delta E}{\partial t} \sim \omega$
Medium time-scale collisions	$\nu \sim \omega \sim \mathcal{O}(\epsilon \Omega_i)$
Small scale spatial variations of perturbations across $\mathbf{B_0}$	$k_\perp \sim \frac{\mathbf{b_0}\times\nabla\delta f}{\delta f} \sim \frac{(\mathbf{b_0}\times\nabla)\delta\mathbf{B}}{\lvert\delta\mathbf{B}\rvert} \sim$ $\sim \frac{(\mathbf{b_0}\times\nabla)\delta\mathbf{E}}{\lvert\delta\mathbf{E}\rvert} \sim \mathcal{O}\left(\frac{1}{\rho_i}\right)$
Large scale spatial variations of perturbations along $\mathbf{B_0}$	$k_\parallel \sim \frac{\mathbf{b_0}\cdot\nabla\delta f}{\delta f} \sim \frac{(\mathbf{b_0}\cdot\nabla)\delta\mathbf{B}}{\lvert\delta\mathbf{B}\rvert} \sim$ $\sim \frac{(\mathbf{b_0}\cdot\nabla)\delta\mathbf{E}}{\lvert\delta\mathbf{E}\rvert} \sim \mathcal{O}\left(\frac{1}{L}\right)$
$\Rightarrow$	$\frac{k_\parallel}{k_\perp} \sim \frac{\rho_i}{L} \sim \mathcal{O}(\epsilon)$
Large scale spatial variation of equilibrium across B	$\frac{\nabla_\perp F_0}{F_0} \sim \frac{\nabla_\perp T_0}{T_0}$ $\sim \frac{\nabla_\perp n_0}{n_0} \sim \mathcal{O}\left(\frac{1}{L}\right)$

We shall develop the gyrokinetic equation in a magnetic flux tube, assumed to be triply periodic. Commensurate with the orderings above, the flux tube is assumed to be macroscopically long in the direction of the magnetic field, but thin across the magnetic field. In such a flux tube, the discrete set of wavenumbers automatically satisfy the relation $k_\parallel L \sim k_\perp \rho$.

1.3 From Boltzmann to Gyrokinetic

1.3.1 Coordinate system for gyrokinetics

The motion of a particle can be described in terms of its position in physical space, $\mathbf{r}$, or by following the center of its gyration motion, the gyro-center $\mathbf{R}$. The transformation from the particle position to its gyro-center is given by:

$$\mathbf{R} = \mathbf{r} + \frac{\mathbf{v} \times \mathbf{b_0}}{\Omega_0}.$$

This transformation to gyro-center position is known as the *Catto Transformation*. We also define the perpendicular velocity $v_\perp$, the parallel velocity $v_\parallel$ and the gyro-angle θ all with respect to the direction of the background magnetic field $\mathbf{B_0} = \mathbf{b_0} B_0$. One can write this as

$$\mathbf{v} = v_\parallel \mathbf{b_0} + v_\perp [\cos(\theta)\hat{\mathbf{e}}_\mathbf{1} + \sin(\theta)\hat{\mathbf{e}}_\mathbf{2}], \tag{1.2}$$

in which the unit vectors $\hat{\mathbf{e}}_\mathbf{1}$, $\hat{\mathbf{e}}_\mathbf{2}$ and $\mathbf{b_0}$ form a right handed coordinate basis, and in general vary on the macroscopic spatial scale L and the slow time scale $t^{-1}_{transport} \sim \mathcal{O}(\epsilon^3 \Omega_i)$. The fastest motion in the problem is the gyration motion of the particles around the strong background magnetic field lines. Upon denoting the gyrophase angle as θ, one can write this as

$$\frac{d\theta}{dt} = -\Omega + \mathcal{O}(\epsilon\Omega).$$

In gyrokinetics it is useful to take advantage of the separation of time scales and to average over the gyration motion of particles, so that one describes the evolution

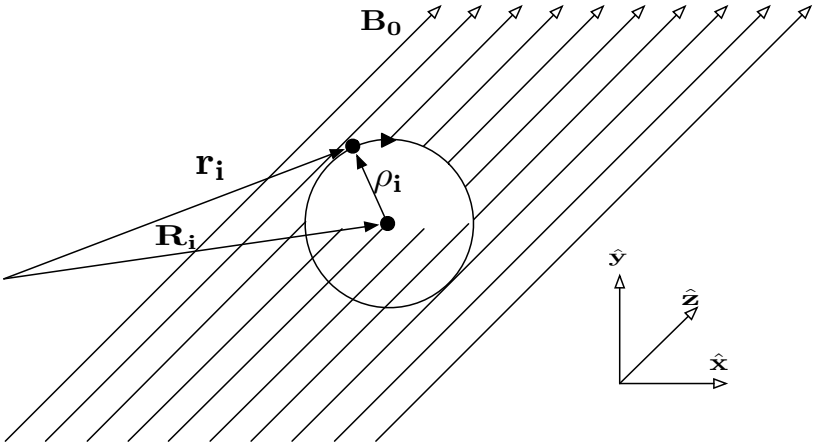

Figure 1.1: Physical and gyro-center coordinate. In a strong magnetic background field a particle executes fast gyration motion around its gyro-center $\mathbf{R}$. The radius of this motion is the gyro-radius ρ. The physical position of the particle is given by the vector $\mathbf{r}$. In this diagram the strong background magnetic field $\mathbf{B_0}$ points in the z-direction, while the gyration motion is perpendicular to $\mathbf{B_0}$ in the (x,y)-plane.

of a distribution of rings rather than the individual particle positions. There are two different forms for the gyro-average that are required.

Ring average at fixed gyro-center position R

We integrate over the gyro-angle θ while keeping the gyro-center $\mathbf{R}$ and the velocities $v_\perp$ and $v_\parallel$ fixed:

$$< \mathcal{A}(\mathbf{r}, \mathbf{v}, t) >_\mathbf{R} = \frac{1}{2\pi} \oint d\theta \mathcal{A} \left(\mathbf{R} - \frac{\mathbf{v} \times \mathbf{b_0}}{\Omega}, \mathbf{v}, t \right) \tag{1.3}$$

Ring average at fixed particle position r

Again we integrate over gyro-angle θ, but this time we hold the particle position $\mathbf{r}$

and the velocities $v_\perp$ and $v_\parallel$ fixed:

$$< \mathcal{A}(\mathbf{r}, \mathbf{v}, t) >_{\mathbf{r}} = \frac{1}{2\pi} \oint d\theta \mathcal{A} \left(\mathbf{r} + \frac{\mathbf{v} \times \mathbf{b_0}}{\Omega}, \mathbf{v}, t \right) \tag{1.4}$$

The assumptions described in Table 1 allow one to order the terms of the Boltzmann equation Eq.(1.1) relative to ϵ.

1.3.2 $\mathcal{O}(\epsilon^{-1})$

The lowest order term in the equation gives:

$$(\mathbf{v} \times \mathbf{B_0}) \cdot \nabla_{\mathbf{v}} F_0 = 0 \tag{1.5}$$

By transforming velocity variables from $\mathbf{v}$ to $(v_\perp, v_\parallel, \theta)$ one finds that the equilibrium solution F_0 is independent of gyrophase angle θ, as follows. Eqs. (1.2) and (1.5) can be combined to yield

$$(\mathbf{v} \times \mathbf{B_0}) \cdot \nabla_{\mathbf{v}} F_0 = \tag{1.6}$$

$$v_\perp \sin(\theta) \frac{\partial}{\partial v_x} F_0 - v_\perp \cos(\theta) \frac{\partial}{\partial v_y} F_0 = \tag{1.7}$$

$$-\frac{\partial \mathbf{v}}{\partial \theta} \cdot \nabla_{\mathbf{v}} F_0 = -v_\perp \frac{\partial}{\partial \theta} F_0 = 0 \tag{1.8}$$

$$\Rightarrow \frac{\partial F_0}{\partial \theta} = 0. \tag{1.9}$$

Thus, the lowest order (equilibrium) distribution function is independent of the gyrophase angle θ.

1.3.3 $\mathcal{O}(1)$

At next order the Fokker-Plank equation becomes

$$\mathbf{v}_\perp \cdot \nabla \delta f_1 + \frac{q}{m}\left(-\nabla\phi + \frac{\mathbf{v}\times\delta\mathbf{B}}{c}\right)\cdot\frac{\partial F_0}{\partial \mathbf{v}} - \Omega\frac{\partial \delta f_1}{\partial\theta} = C(F_0, F_0) \tag{1.10}$$

To find the solution for F_0 from Eq.(1.10), we multiply by $(1+F_0)$ and integrate over the entire phase space. Transverse to the magnetic field, the domain of integration is sufficiently small that $F_0(x,y) \sim$ const, and as noted earlier, periodicity is assumed. The perturbed quantities will therefore have zero spatial average. Only the RHS survives the phase-space integration. Using Boltzmann's H-Theorem we know that this gives us the unique solution that F_0 is Maxwellian:

$$\int d^3\mathbf{r}\int d^3\mathbf{v}(\ln F_0)C(F_0,F_0) = 0 \underbrace{\Rightarrow}_{H-theorem}$$

$$F_0 = \frac{n_0}{(\sqrt{2\pi}v_T)^3}\exp\left(-\frac{v^2}{2v_T}\right). \tag{1.11}$$

Since the Maxwellian solution F_0 leads to $C(F_0,F_0) = 0$, we can now substitute Eq.(1.11) into Eq.(1.10) to find

$$\mathbf{v}_\perp\cdot\nabla\delta f_1 - \Omega\frac{\partial\delta f_1}{\partial\theta} = -\mathbf{v}\cdot\nabla\left(\frac{q\phi}{T_0}\right)F_0. \tag{1.12}$$

On the right hand side of the Eq. (1.12) we can drop the term $v_\parallel \mathbf{b_0}\cdot\nabla\left(\frac{q\phi}{T}\right)F_0$ since this term is one order smaller in ϵ (because $k_\parallel \ll k_\perp$). Therefore the particular solution of the differential equation Eq.(1.12) can be identified as

$$\delta f_{1_p} = -\left(\frac{q\phi}{T}\right)F_0. \tag{1.13}$$

By going from particle position to gyro-center position and using the identity

$$\Omega\left(\frac{\partial}{\partial\theta}\right)_{\mathbf{R}} = \Omega\left(\frac{\partial}{\partial\theta}\right)_{\mathbf{r}} - \mathbf{v}_\perp\cdot\nabla \tag{1.14}$$

we find that the homogenous solution, h, has to satisfy the following equation:

$$\mathbf{v}_\perp \cdot \nabla h - \Omega \left(\frac{\partial h}{\partial \theta}\right)_{\mathbf{r}} = \left(\frac{\partial h}{\partial \theta}\right)_{\mathbf{R}} = 0 \tag{1.15}$$

Therefore the homogenous solution to Eq.(1.12) is independent of the gyro-angle θ at constant gyro-center position $\mathbf{R}$:

$$h = h(\mathbf{R}, v, v_\perp, t) =< h(\mathbf{R}, v, v_\perp, t) >_{\mathbf{R}} .$$

Collecting the results obtained so far, the distribution function f can be written as

$$f = \left(1 - \frac{q\phi}{T}\right) F_0 + h(\mathbf{R}, v, v_\perp, t) + \mathcal{O}(\epsilon^2) \tag{1.16}$$

with

$$\delta f_1 = h(\mathbf{R}, v, v_\perp, t) - \frac{q\phi}{T} F_0.$$

The *gyro-averaged* perturbed distribution function thus can be written as

$$< \delta f_1 >_{\mathbf{R}} = h(\mathbf{R}, v, v_\perp, t) - \frac{q < \phi >_{\mathbf{R}}}{T} F_0.$$

We note for future reference that $< \delta f_1 >$ describes the turbulence, and will be the quantity that we simulate.

1.3.4 $\mathcal{O}(\epsilon)$

Using the solution given in Eq.(1.16) the Fokker-Planck equation, Eq.(1.1) to order ϵ in gyro-center coordinates becomes:

$$\begin{aligned} \frac{\partial h}{\partial t} \quad + \quad & \frac{d\mathbf{R}}{dt} \cdot \left(\frac{\partial h}{\partial \mathbf{R}} + \frac{\partial F_0}{\partial \mathbf{R}}\right) + \mathbf{v}_\perp \cdot \nabla F_0 + \frac{q}{m}\left(-\nabla_\perp \phi + \frac{\mathbf{v} \times \delta \mathbf{B}}{c}\right) \cdot \left(\frac{\mathbf{v}}{v}\frac{\partial h}{\partial v} + \frac{\mathbf{v}_\perp}{v_\perp}\frac{\partial h}{\partial v_\perp}\right) \\ = \quad & C(h, F_0) + C(F_0, h) + \Omega \left(\frac{\partial \delta f_2}{\partial \theta}\right)_{\mathbf{R}} + \frac{q}{T_0}\left(\frac{\partial \phi}{\partial t} - \frac{\mathbf{v}}{c} \cdot \frac{\partial \mathbf{A}}{\partial t}\right) F_0 \end{aligned} \tag{1.17}$$

with:

$$\frac{d\mathbf{R}}{dt} = v_{\parallel}\mathbf{b_0} + \frac{c}{B_0}\left(-\nabla\phi - \frac{1}{c}\frac{\partial \mathbf{A}}{\partial t} + \frac{\mathbf{v}\times\delta\mathbf{B}}{c}\right)\times\mathbf{b_0}$$

Note that the collision operator involve the distribution functions F_0 and h both for electrons and ions. To get an equation that no longer depends on the second order perturbed distribution function δf_2 we use Eq.(1.3) to take gyro-average at constant gyro-center position $\mathbf{R}$. By using the identity that $< \mathbf{v}_\perp \cdot \nabla\mathcal{A} >_{\mathbf{R}} = 0$ for an arbitrary function $\mathcal{A}(\mathbf{r})$ we can simplify Eq.(1.17) a great deal and get the following equation to solve:

$$\frac{\partial h}{\partial t} + \left\langle \frac{d\mathbf{R}}{dt} \right\rangle_{\mathbf{R}} \cdot \left(\frac{\partial h}{\partial \mathbf{R}} + \frac{\partial F_0}{\partial \mathbf{R}}\right) = \left(\frac{\partial h}{\partial t}\right)_{coll} + \frac{q}{T_0}\frac{\partial < \chi >_{\mathbf{R}}}{\partial t} F_0 \tag{1.18}$$

with:

$$\left(\frac{\partial h}{\partial t}\right)_{coll} = \langle C(F_0, h) + C(h, F_0)\rangle_{\mathbf{R}} \tag{1.19}$$

$$\left\langle \frac{d\mathbf{R}}{dt} \right\rangle_{\mathbf{R}} = v_{\parallel}\mathbf{b}_0 - \frac{c}{B_0}\frac{\partial \langle\chi\rangle_{\mathbf{R}}}{\partial \mathbf{R}} \times \mathbf{b}_0 \tag{1.20}$$

$$\text{and the gyrokinetic potential } \chi = \phi - \frac{\mathbf{v}\cdot\mathbf{A}}{c} \tag{1.21}$$

The final term on the right hand side of Eq. (1.18) describes the work done on the particles by the fluctuating fields. The physical meanings for the terms in Eq. (1.20) are:

- $v_{\parallel}\mathbf{b}_0$: free streaming along the equilibrium field
- $\frac{c}{B_0}\frac{\partial\langle\chi\rangle_{\mathbf{R}}}{\partial\mathbf{R}} \times \mathbf{b}_0$: gyro-averaged perpendicular drifts.

Together with the low-frequency Maxwell equations,[2] these equations constitute the model we will study through the first part of the thesis. In the final chapter,

[2]Actually, since we are addressing only the electrostatic gyrokinetic problem, we need only quasineutrality.

we change the geometry of the problem and go from a slab to a toroidal geometry. This will add a curvature drift to the set of equations. Furthermore we are going to allow for the magnetic field to have a spatial dependence in the perpendicular direction. This will lead to an additional drift term, the $\nabla\mathbf{B}$-drift.

1.4 Non-dimensionalization

Eqs. (1.18-1.21) can be expressed in non-dimensional form, consistent with the orderings of Table I. Short distances across the magnetic field, characteristic of the scale of the perturbations of interest, are normalized by ρ_i. For simplicity, if the equilibrium magnetic field points in the $\mathbf{z}$ direction, then

$$(x_N, y_N) = (\frac{x}{\rho_i}, \frac{y}{\rho_i}).$$

Distances along the magnetic field are normalized by the macroscopic length L, previously introduced, so that

$$z_N = \frac{z}{L}.$$

Time is normalized by the quantity v_t/L,

$$t_N = \frac{tv_t}{L}.$$

It is convenient to scale the perturbed quantities by $L/\rho_i = \epsilon^{-1}$, so that they are manifestly of order unity. Apart from this factor, the electrostatic potential is normalized in the usual way,

$$\Phi_N = \frac{q\Phi}{T}\frac{L}{\rho_i}$$

where q and T are conventionally taken to be the (hydrogenic) ion charge and temperature, respectively. Upon making these substitutions, one finds that the final equations (and thus any simulation results) depend on only a few nondimensional quantities, such as the relative charges of the plasma species, their relative densities, various ratios of equilibrium gradients, and so on. There is no dependence of this nonlinear gyrokinetic set of equations on the expansion parameter ρ_i/L itself. When modeling a particular physical system, one should always check that that physical system has a sufficiently small value of ρ_i/L. How small this parameter must be for gyrokinetic simulations to be a faithful description of the system is a question that is outside gyrokinetic theory itself.

The full set of non-dimensional equations will be presented in the following chapter, after the method of characteristics has been employed to produce the form of the equations simulated in this thesis.

Chapter 2

Introduction to the Code

To study gyrokinetic dynamics, a new *Particle In Cell Code* (GSP) has been produced. The numerical method used is a *δf-method*, which solves the nonlinear gyrokinetic equation along a set of characteristics. This method for solving the gyrokinetic equation was first implemented by Kotschenreuther and Denton [19] and by Dimits and Lee [21]. Since then several codes have been developed and the method has been extended to solve the electromagnetic gyrokinetic equation in a toroidal geometry. In the last decade massively parallel nonlinear δf-PIC codes such as GTC [46] and PG3EQ [22] were developed.

In a δf-method, only the perturbed part of the distribution function evolves over time and the equilibrium part of the distribution function is held constant. The part of the noise in the system that is associated with capturing the equilibrium part F_0 from the actual particle positions in the code is eliminated. This leads to a great advantage in terms of noise reduction since the fluctuations in the perturbed part of the distribution function may be orders of magnitude smaller than the equilibrium distribution, but are still crucial for determining physical quantities such as heat flux

and particle transport. Hence, by not having to resolve F_0 with the sampled particles in the code, one zeros the discrete particle noise in the background distribution and it becomes possible to address the problem with a smaller number of particles. The alternative to a δf-code is a *Full-f-Code*. In such a code the particles sample the full distribution $f = F_0 + \delta f$. Early full-f particle gyrokinetic codes were developed by Lee [44], and by Tajima and co-workers. Currently, a full-f semi-Lagrangian algorithm has been developed by Grandgirard, *et al.,* at Cadarache. Approaches to these extensions of gyrokinetics have varied. Tajima, *et al.,* integrated the full characteristics of the Vlasov equation, including particle gyration. This is very expensive, as one must treat frequencies much higher than those of direct interest. More recently, it has been the practice to include *ad hoc* higher-order corrections to Eqs. (1.18-1.21), such as might arise in a system for which ρ_i/L is not very small. In most or all of these cases, no formal derivation of a set of equations to replace the gyrokinetic model has been attempted. Here, we focus on simulating gyrokinetic dynamics in the $\rho_i/L = 0$ limit.

Besides all the development of particle codes there have also been extensive studies of the gyrokinetic equation using continuum methods. The first initial value continuum code, GS2, was developed by Kotschenreuther [40] to solve the linear gyrokinetic equation in ballooning coordinates. Liu and Dorland [24] extended this code to include nonlinear dynamics and general geometry [5] for parallel computers. Other nonlinear contiuum gyrokinetic codes in wide use for fusion calculations are GYRO developed by Candy and Waltz [11] and GENE developed by Jenko [37].

In this thesis we use GS2 and AstroGK [35] to benchmark the results from

our PIC-code GSP with the results from those codes. AstroGK is a simpler of version of gs2 that was developed by Dorland, Tatsuno, Howes, and Numata [35], in which geometry effects from a toroidal geometry have been eliminated from gs2 and instead the gyrokinetic equation is solved in a slab.

To explain the numerical scheme that we are using we choose a fairly simple physical case. We allow for background gradients in the temperature profile T_0 and in the particle density n_0. Both these gradients are in the perpendicular direction to the background magnetic field $\mathbf{B}_0$. In addition we make the system collisionless, electrostatic ($\chi = \phi$) and solve it in a slab (no curvature). Note that already in Chapter 1 we made the assumption of being in a slab and had no curvature drifts. Under these assumptions, we can rewrite the equations in terms of $< \delta f >$ instead of h. The system of equations Eq.(1.18) - Eq.(1.21) then becomes:

$$\frac{\partial}{\partial t}\left\langle \delta f \right\rangle_{\mathbf{R}} + v_{\parallel}\mathbf{b}_0 \cdot \nabla \left\langle \delta f \right\rangle_{\mathbf{R}} + \left\langle \mathbf{v}_{E\times B} \right\rangle_{\mathbf{R}} \cdot \nabla \left\langle \delta f \right\rangle_{\mathbf{R}} =$$
$$- \left\langle \mathbf{v}_{E\times B} \right\rangle_{\mathbf{R}} \cdot \nabla F_0 - v_{\parallel}\frac{q}{T}F_0 \ \mathbf{b}_0 \cdot \nabla \left\langle \phi \right\rangle_{\mathbf{R}} \tag{2.1}$$

$$\text{with:} \quad \mathbf{v}_{E\times B} = \frac{c}{B_0}\mathbf{b}_0 \times \nabla\phi \tag{2.2}$$

The symbol $\mathbf{v}_{E\times B}$ stands for the $\mathbf{E}\times\mathbf{B}$-drift, which is entirely in the plane perpendicular to the background magnetic field $\mathbf{B}_0$. From here on we are going to use the operators $\nabla_{\parallel} = \mathbf{b}_0 \cdot \nabla$, and $\nabla_{\perp}$, which is in the perpendicular direction to $\mathbf{b}_0$. Since our code is a δf particle-in-cell code we now consider the method of characteristics.

2.1 Method of Characteristics

A PIC-δf-code solves Eq.(2.1) using the method of characteristics together with time-dependent weights. Since Eq.(2.1) is a first-order partial differential equation (PDE), we can find characteristic curves along which the PDE becomes a set of Ordinary Differential Equations (ODEs). We then solve the set of ODEs along the characteristic curves.

Following the notation from Sarra [57] we can rewrite a PDE in a general form with n independent variables q_i to get the following expression:

$$\sum_{i=1}^{n} \mathcal{A}_i \frac{\partial \delta f}{\partial q_i} = \mathcal{B} \tag{2.3}$$

We have to solve Eq.(2.1), a quasilinear PDE. Quasilinear means that the coefficients $\mathcal{A}_i$ can be functions of all variables q_i and the value of δf, but not of any partial derivative. Therefore the characteristic curves are given parametrically by $(q_1.q_2, \ldots, q_n, \delta f) = (q_1(c), q_2(c), \ldots, q_n(c), \delta f(c))$ and we can rewrite Eq.(2.3) by the following system of $(n+1)$ ODEs:

$$\frac{d}{dc} q_i = \mathcal{A}_i(q_i, \ldots, q_n, \delta f), \quad \text{for} \;\; i = 1, \ldots, n \tag{2.4}$$

$$\frac{d}{dc} \delta f = \sum_{i=1}^{n} \frac{\partial u}{\partial q_i} \frac{d \delta f}{dc} = \sum_{i=1}^{n} \frac{\partial \delta f}{\partial q_i} \mathcal{A}_i = \mathcal{B} \tag{2.5}$$

Eq.(2.4) gives us the n characteristic curves for this system while Eq.(2.5) defines that δf is a constant along those characteristics.

We rewrite Eq.(2.1) so we can apply this method, where the parameter c becomes the time t and the variables q_i represent the 5-dimensional phase space $(x, y, z, v_\perp, v_\parallel)$, given by the spatial components of the gyro-center position $\mathbf{R}$ and

two velocity components:

$$\begin{aligned}
&\frac{d}{dt} < \delta f >_{\mathbf{R}} = \\
= \quad & \frac{\partial < \delta f >_{\mathbf{R}}}{\partial t} + \left(\frac{d}{dt}\mathbf{R}\right) \cdot \nabla < \delta f >_{\mathbf{R}} + \left(\frac{d}{dt} v_{\perp}\right) \frac{\partial < \delta f >_{\mathbf{R}}}{\partial v_{\perp}} + \left(\frac{d}{dt} v_{\parallel}\right) \frac{\partial < \delta f >_{\mathbf{R}}}{\partial v_{\parallel}} = \\
= \quad & - \left\langle \mathbf{v}_{E\times B} \right\rangle_{\mathbf{R}} \cdot \nabla F_0 - v_{\parallel} \frac{q}{T} F_0 \ \mathbf{b}_0 \cdot \nabla \left\langle \phi \right\rangle_{\mathbf{R}} \qquad (2.6)
\end{aligned}$$

So the characteristics for this system are:

$$\frac{d}{dt}\mathbf{R}_{\perp} = < \mathbf{v}_{\mathbf{E}\times\mathbf{B}} >_{\mathbf{R}} \qquad (2.7)$$

$$\frac{d}{dt} R_{\parallel} = v_{\parallel} \qquad (2.8)$$

$$\frac{d}{dt} v_{\parallel} = 0 \qquad (2.9)$$

$$\frac{d}{dt} v_{\perp} = 0 \qquad (2.10)$$

Along those characteristics the PDE δf is a solution of:

$$\frac{d}{dt}\delta f = - \left\langle \mathbf{v}_{E\times B} \right\rangle_{\mathbf{R}} \cdot \nabla F_0 - v_{\parallel} \frac{q}{T} F_0 \ \mathbf{b}_0 \cdot \nabla \left\langle \phi \right\rangle_{\mathbf{R}} . \qquad (2.11)$$

Thus in the code the particles that are pushed around represent gyro-center positions and move in the 5D phase-space accordingly to the Eq.(2.7)-Eq.(2.8). We will take advantage of the fact that the trajectories are constant in velocity space (Eq.(2.9) - Eq.(2.10)). We will refer to this again when we talk about how the velocity space is set up in section 2.3.1.

To deal with Eq.(2.11) we define particle weights. The weight of an individual particle i with gyro-center position $\mathbf{R_i}$ and velocities $v_{\perp i}, v_{\parallel i}$ is defined to be the ratio of the gyro-averaged perturbed distribution function divided by the background distribution F_0 evaluated at the particle's position in the 5D phase space:

$$w_i \equiv \frac{< \delta f >_{\mathbf{R}}}{F_0} |_{\mathbf{R_i}, v_{\perp i}, v_{\parallel i}} \qquad (2.12)$$

The perturbed distribution function for a total number of N simulation particles is then obtained by evaluating the following sum:

$$\delta f(x, y, z, v_\parallel, v_\perp) = \sum_{i=1}^{N} w_i \delta(\mathbf{x} - \mathbf{R_i}) \delta(v_\parallel - v_{\parallel i}) \delta(v_\perp - v_{\perp i}) \tag{2.13}$$

In the code we use a regular grid for the spatial coordinates (x, y, z). To represent the perturbed distribution function on such a grid, we have to replace in Eq.(2.13) the δ-function in $\mathbf{x}$ by a particle shaping function S that depends on the grid-coordinates (X, Y, Z). For the shaping function we use a bilinear interpolation scheme that is second order accurate to determine the values of $\delta f(X, Y, Z, v_\parallel, v_\perp)$.

The bilinear interpolation scheme uses interpolation weights which we represent by the symbol $g_{i,j}$. The subscript i stands for the individual particle i whose gyro-center is located at x_i, y_i, z_i and the subscript j is a triplet of integer numbers (j_1, j_2, j_3) that characterizes a grid-point $X_{j_1}, Y_{j_2}, Z_{j_3}$. Therefore $g_{i,j}$ gives us the contribution of the bilinear interpolation of particle i onto grid-point $X_{j_1}, Y_{j_2}, Z_{j_3}$. The grid-points are equally spaced in each direction, respectively by $\Delta x, \Delta y$ and Δz. The formula for the interpolation weights is:

$$g_{i,j} = \frac{1}{\Delta x \Delta y \Delta z} G_{x_{j,i}} G_{y_{j,i}} G_{z_{j,i}} \tag{2.14}$$

$$\text{with:} \quad G_{x_{j,i}} = \begin{cases} X_{j+1} - R_{x_i} & : \quad X_j < R_{x_i} < X_{j+1} \\ R_{x_i} - X_{j-1} & : \quad X_{j-1} < R_{x_i} < X_j \\ 0 & : \quad else. \end{cases} \tag{2.15}$$

Hence the perturbed distribution function interpolated onto the grid is found by evaluating:

$$\delta f(X_j, Y_j, Z_j, v_\parallel, v_\perp) = \sum_{i=1}^{N} w_i g_{i,j} \delta(v_\parallel - v_{\parallel i}) \delta(v_\perp - v_{\perp i})$$

The time dependence of δf has two contributions. First, the gyro-center particle positions change over time based on Eq.(2.7) and Eq.(2.8), leading to a change in the interpolation weights $g_{i,j}$. Second, the particle weights evolve over time determined by Eq.(2.21). Before we go on to derive the time dependence of the weights we want to emphasize the time dependence of the gyro-center positions in the perpendicular direction, Eq.(2.7), is a nonlinear term. When we are running the code linearly, the RHS of Eq.(2.7) is set to zero. In this case the characteristics consist just of a free streaming motion down the field lines and we only need to update the particles parallel coordinate, z, while the other four coordinates in phase space are held fixed for each individual particle. Thus in a linear run, all dynamics are captured by updating the weights over time while the particles are free streaming down the field lines.

To find the equation that describes the evolution of the particle weights over time we start from the ODE that was given in Eq.(2.11) and find the following behavior for the particle weights over time:

$$\frac{d}{dt}w_i = -\left(< \mathbf{v}_{\mathbf{E}\times\mathbf{B}} >_{\mathbf{R}} \cdot \frac{\nabla F_0}{F_0} + \frac{q}{T_0} v_\parallel < E_\parallel >_{\mathbf{R}} \right). \tag{2.16}$$

The equilibrium F_0 is described by a Maxwellian with $v^2 = v_\perp^2 + v_\parallel^2$. For the particle density and the temperature profile we allow a spatial dependence in the x-direction. Hence F_0 is given by:

$$F_0 = \frac{n(x)}{\left(\sqrt{2\pi}\sqrt{\frac{T(x)}{m_i}}\right)^3} \exp\left(-\frac{v^2}{2\frac{T(x)}{m_i}}\right) \tag{2.17}$$

$$\text{with:} \quad T(x) = T_0 \exp\left(-\frac{x}{L_T}\right) \quad \text{and} \quad n(x) = n_0 \exp\left(-\frac{x}{L_n}\right). \tag{2.18}$$

Now we can use this definition for the Maxwellian F_0 to evaluate the gradient of the background distribution function in Eq.(2.16):

$$\frac{\nabla F_0}{F_0} = \frac{n'(x)}{n(x)} - \frac{3}{2}\frac{T'(x)}{T(x)} + \frac{v^2 T'(x) m_i}{2T^2(x)} = \tag{2.19}$$

$$= -\frac{1}{L_n} + \frac{3}{2L_T} - \frac{v^2}{2L_T v_{T_i}^2}. \tag{2.20}$$

We end up with the following ODE for the particle weights:

$$\frac{d}{dt}w_i = -\left(< \mathbf{v}_{\mathbf{E}\times\mathbf{B}} >_{\mathbf{R}} \cdot\hat{\mathbf{x}} \left(-\frac{1}{L_n} + \frac{3}{2L_T} - \frac{v^2}{2L_T v_{T_i}^2} \right) + \frac{q}{T_0} v_{\|} < E_{\|} >_{\mathbf{R}} \right) \tag{2.21}$$

This leaves us with a set of three ODEs solve, Eq.(2.7), Eq.(2.8) and Eq.(2.21). The code uses a second order predictor-corrector method to solve those ODEs. Eq.(2.7) and Eq.(2.21) both have an explicit dependence on the electrostatic potential ϕ and its gradient $\nabla\phi$. Hence, in order to be able to solve these equations we need to use Maxwell's equations to find the electrostatic potential ϕ given the perturbed distribution function δf. Before turning to the field equations, we quickly non-dimensionalize the characteristic equations.

2.2 Non-dimensional gyrokinetic equations

We wish to evolve Eqs. 2.7-2.10) and (2.21) numerically. It is convenient to non-dimensionalize the equations first. See Appendix B for a complete tabulated list of the normalized quantities.

Consider Eq. (2.7) first:

$$\frac{d}{dt}\mathbf{R}_{\perp} = < \mathbf{v}_{\mathbf{E}\times\mathbf{B}} >_{\mathbf{R}} .$$

We define a "reference" Larmor radius ρ_{ref} whose temperature, mass, charge and density are chosen appropriately for any given problem. The reference Larmor radius is defined in turn in terms of the reference thermal speed $v_{T_{ref}} \equiv \sqrt{T_{ref}/m_{ref}}$ and the reference cyclotron frequency $\Omega_{ref} \equiv Z_{ref}|e|B/m_{ref}c$. In the directions perpendicular to the local equilibrium magnetic field, we normalize lengths by ρ_{ref}. Upon dividing both side of Eq. (2.7) by ρ_{ref}, one finds

$$\frac{d}{dt}\mathbf{R}_{\perp\mathbf{N}} = \frac{< \mathbf{v}_{\mathbf{E}\times\mathbf{B}} >_{\mathbf{R}}}{\rho_{ref}}, \tag{2.22}$$

where $\mathbf{R}_{\perp\mathbf{N}} = \mathbf{R}_{\perp}/\rho_{ref}$. As anticipated at the end of the previous chapter, we normalize the electrostatic potential as

$$\phi_N \equiv \frac{Z_{ref}|e|\phi}{T_{ref}}\frac{a}{\rho_{ref}}, \tag{2.23}$$

where we have introduced an arbitrary equilibrium scale length a and all other symbols are conventional. It is convenient to normalize time by $a/v_{T_{ref}}$,

$$t_N \equiv \frac{t v_{T_{ref}}}{a}.$$

After normalizing the perpendicular gradient in Eq. (2.22) by ρ_{ref} and upon employing the definition of ϕ_N in Eq. (2.23), one finds

$$\frac{d}{dt_N}\mathbf{R}_{\perp\mathbf{N}} = \frac{< \mathbf{v}_{\mathbf{E}\times\mathbf{B},\mathbf{N}} >_{\mathbf{R}}}{\rho_{ref}}, \tag{2.24}$$

where the normalized $\mathbf{E}\times\mathbf{B}$ velocity is given by

$$< \mathbf{v}_{\mathbf{E}\times\mathbf{B},\mathbf{N}} >_{\mathbf{R}} \equiv \hat{\mathbf{z}} \times \nabla_N < \phi_N >_{\mathbf{R}} .$$

Unless there is ambiguity, we will drop the subscript N in the sequel.

To normalize Eq. (2.8), we recall that since $k_\parallel/k_\perp \sim \epsilon$, it is most convenient to normalize lengths along the field line by a, the reference macroscopic length. Thus,

$$z_N \equiv \frac{z}{a}.$$

We choose to normalize particle speeds by v_{ts}, the thermal speed of the s^{th} species, instead of by $v_{T_{ref}}$. The normalized version of Eq. (2.8) then reads

$$\frac{d}{dt_N} R_{\parallel,N} = \sqrt{\frac{T_{sN}}{m_{sN}}} v_{\parallel N}.$$

Proceeding entirely analogously, and remembering that the perturbed fields $(\delta f, \phi)$ are normalized with a factor of a/ρ_{ref} so that they are manifestly of order unity in a nonlinear state, the normalized version of Eq. (2.21) reads

$$\frac{d}{dt_N} w_{iN} = -\left(< \mathbf{v_{E\times B,N}} >_{\mathbf{R}} \cdot \hat{\mathbf{x}} \left(-\frac{a}{L_{ns}} + \frac{3}{2}\frac{a}{L_{Ts}} - \frac{v^2}{2}\frac{a}{L_{Ts}} \right) + \frac{Z_{sN}}{\sqrt{T_{sN} m_{sN}}} v_{\parallel N} < E_{\parallel N} >_{\mathbf{R}} \right).$$

With these normalizations, the quantity a/ρ_{ref} does not appear in the simulated system, as is appropriate, since the gyrokinetic equations are independent of the asymptotic expansion parameter ϵ. Of course, to compare the simulated quantities with any physical system, it is necessary to define ρ_{ref} and a, and to use these values to express the simulation results in physical units.

2.3 Poisson's Equation

We use Poisson's equation to determine the electrostatic potential ϕ. In the electrostatic case Poisson's equation has the following form:

$$\nabla^2 \phi = -4\pi \left(q_i n_i + q_e n_e \right) \tag{2.25}$$

When $k_\perp^{-1}$ is long compared to the Debye length, λ_D, we can drop the left hand side of the Poisson's equation and get the **quasineutrality** condition:

$$q_i n_i = -q_e n_e \tag{2.26}$$

For simplicity we are continuing considering a hydrogenic plasma ($\Rightarrow n_i = n_e$). The perturbed particle density n is calculated by gyro-averaging the perturbed distribution function at constant particle position $\mathbf{r}$ and integrating it over the entire velocity part of phase space:

$$n = \int < \delta f >_{\mathbf{r}} \; dv_\parallel v_\perp \; dv_\perp \tag{2.27}$$

We recall from Chapter 1 that $\delta f = h - \frac{q}{T}\phi F_0$ and $< \delta f >_{\mathbf{R}} = h - \frac{q}{T} < \phi >_{\mathbf{R}} F_0$. By combining these two identities for h we find the following equation for the perturbed distribution function δf:

$$\delta f_s = < \delta f_s >_{\mathbf{R}} + \frac{q_s}{T_s} F_0 \left(< \phi >_{\mathbf{R}} - \phi \right) \tag{2.28}$$

Plugging Eq.(2.28) into Eq.(2.27) we obtain the following integral equation that we have to solve for finding the perturbed ion particle density:

$$\begin{aligned} n_i &= \int < \delta f_i >_{\mathbf{r}} dv_\parallel v_\perp dv_\perp = \\ &= \int \left\langle \left(< \delta f_i >_{\mathbf{R}} + \frac{|e|}{T_i} F_0 \left(< \phi >_{\mathbf{R}} - \phi \right) \right) \right\rangle_{\mathbf{r}} dv_\parallel v_\perp dv_\perp. \end{aligned} \tag{2.29}$$

The electron particle density n_e has an easy solution since we assume Boltzmann electrons and therefore obtain

$$n_e = \frac{|e|\phi}{T_e} n_0. \tag{2.30}$$

The assumption of Boltzmann allows us to have just one kinetic species in the code, the ions. So from here on when we talk about particles and their trajectories we mean ions if not otherwise specified. So now we can set Eq.(2.30) equal to Eq.(2.29) and get the following equation:

$$\int \left\langle \left(< \delta f >_{\mathbf{R}} + \frac{|e|}{T_i} F_0 \left(< \phi >_{\mathbf{R}} - \phi \right) \right) \right\rangle_{\mathbf{r}} dv_{\parallel} v_{\perp} dv_{\perp} = n_e = \frac{|e|\phi}{T_e} n_0 \qquad (2.31)$$

We can then use Eq.(2.31) to find a solution for ϕ. Numerically we do this in Fourier space and by explicitly evaluating the gyro-averages in Eq.(2.31). This is something new that we are doing in this thesis. In a standard gyrokinetic PIC code the gyro-averaging is done in a different way. When calculating $\langle \delta f \rangle_{\mathbf{R}}$ the standard PIC-codes place a finite number of test-particles onto a ring that is centered at the gyro-center position $\mathbf{R}$ of a particle. The radius of this ring for particle i is given by its ion-gyro-radius, $\rho_i = \frac{v_{\perp i}}{\Omega_0}$. This is done for every individual gyro-center position. Then the test-particle positions are interpolated onto the grid on which the code the code represents δf. So if the code uses a total number of particles that is $\mathcal{N}$, the interpolation onto the grid will actually be done for $N_{ring} \times \mathcal{N}$ particles, where N_{ring} is the number of test-particles that are placed on each ring around the gyro-center position of a particle. In Fig(2.1) we give a pictorial description for this approximation approach for taking the gyro-average by placing test particles on a ring.

The following two identities for the gyro-agerages are important to describe our method and are used in Eq.(2.31) to solve for ϕ in Fourier Space.

$$< e^{i\mathbf{k}\cdot\mathbf{r}} >_{\mathbf{R}} = J_0(\frac{k_\perp v_\perp}{\Omega}) e^{i\mathbf{k}\cdot\mathbf{R}}, \quad < e^{i\mathbf{k}\cdot\mathbf{R}} >_{\mathbf{r}} = J_0(\frac{k_\perp v_\perp}{\Omega}) e^{i\mathbf{k}\cdot\mathbf{r}} \qquad (2.32)$$

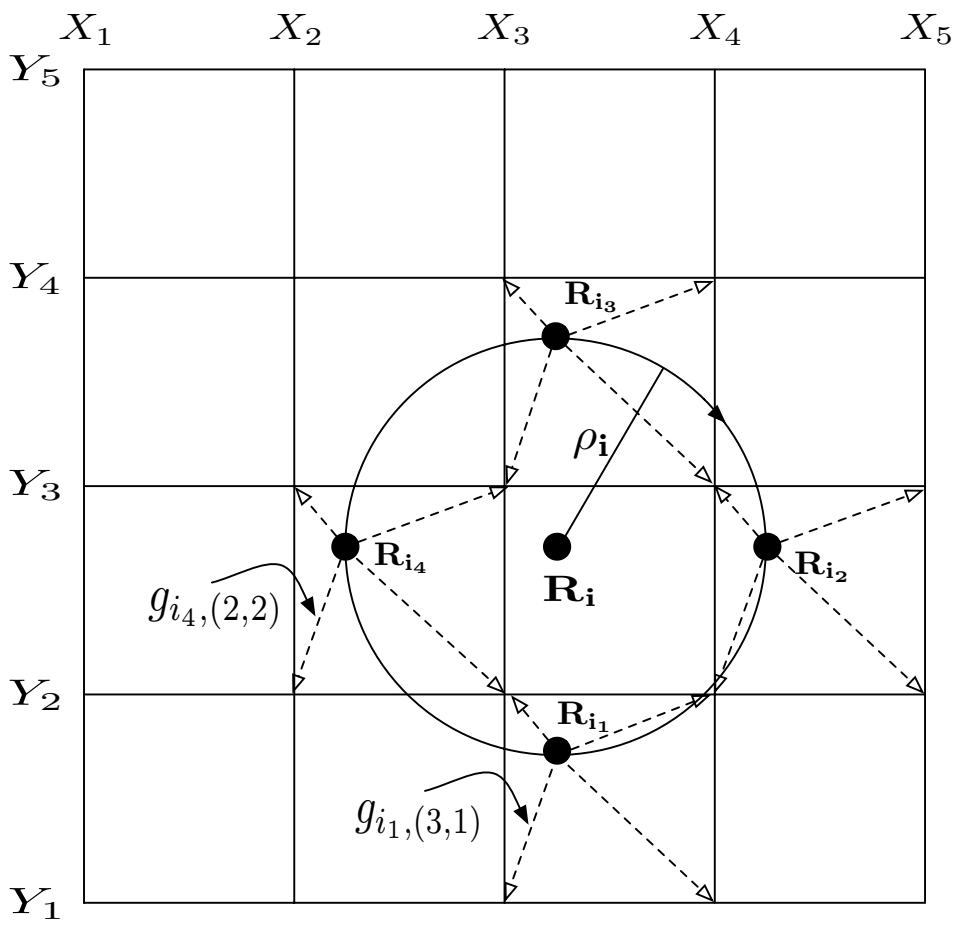

Figure 2.1: Evaluation of the gyro-average by placing a number of test-particles on a ring that describes the gyration motion of a particle. To simplify the problem we show an interpolation scheme onto a 2-dimensional grid (X, Y) instead of the full 3D version. We are using four test particles ($N_{ring} = 4$) that we place on a ring with radius ρ_i around the gyro-center $\mathbf{R_i}$ of particle i. The test particles' positions are given by $\mathbf{R_{i_k}}$ with $k = 1, 2, 3, 4$. The dashed arrows represent the interpolation weights $g_{i_k,(j_x,j_y)}$ that are non-zero a. $g_{i_k,(j_x,j_y)}$ is the interpolation weight of test particle i_k that is interpolated onto the grid-point (X_{j_x}, Y_{j_y})

Applying them to Eq.(2.31) we find an equation for ϕ:

$$n_0\left(\frac{|e|}{T_e}+\frac{|e|}{T_i}\left(1-\Gamma_0(k_\perp^2\rho_i^2)\right)\right)\phi=\int J_0\left(\frac{k_\perp v_\perp}{\Omega}\right)<\delta f>_{\mathbf{R}} v_\perp dv_\perp dv_\parallel \Rightarrow$$

$$\phi=\frac{\int J_0\left(\frac{k_\perp v_\perp}{\Omega}\right)<\delta f>_{\mathbf{R}} v_\perp dv_\perp dv_\parallel}{n_0\left(\frac{|e|}{T_e}+\frac{|e|}{T_i}\left(1-\Gamma_0(k_\perp^2\rho_i^2)\right)\right)} \tag{2.33}$$

The quantity J_0 is the zeroth order Bessel function of the first kind and $\Gamma_0(x) = I_0(x)e^{-x}$, where I_0 is the zeroth order Bessel function of the second kind. To explicitly evaluate J_0 in the velocity space integral of Eq.(2.33) we use a grid in $v_\perp$. In the next section we describe the representation of the velocity space that we are using to calculate the integrals in Eq.(2.33) for obtaining ϕ.

2.3.1 Velocity Space for Solving Poisson's Equation

In GSP the perpendicular velocities, $v_\perp$, are initialized on a regular grid and the parallel velocities, $v_\parallel$, are initialized gridless, drawn from a uniform random distribution. Each individual particle i therefore gets a random velocity $v_{\parallel i}$ and a velocity $v_{\perp i}$ that is part of a discrete set of velocities $v_\perp$ which lie on an equally spaced grid. We restrict the value for the energy of a single particle, $E_i = v_{\perp i}^2 + v_{\parallel i}^2$ to be smaller than a maximum value E_{max}, which is a parameter that we can set in the input file to the code. By doing this we initialize the particles inside a semi-circle with radius = E_{max} in the upper half of the $(v_\perp, v_\parallel)$-plane, as shown in Fig.(2.3).

We want to have a velocity-space density of simulation particles that is roughly constant. That means that for a constant value of $v_\perp$ the number of particles is inversely proportional to the value of $v_\perp$. This is explained by the fact that the maximum value of $v_\parallel$ for a fixed value of $v_\perp$ is limited by $v_\parallel \leq \sqrt{E_{max} - v_\perp^2}$. So

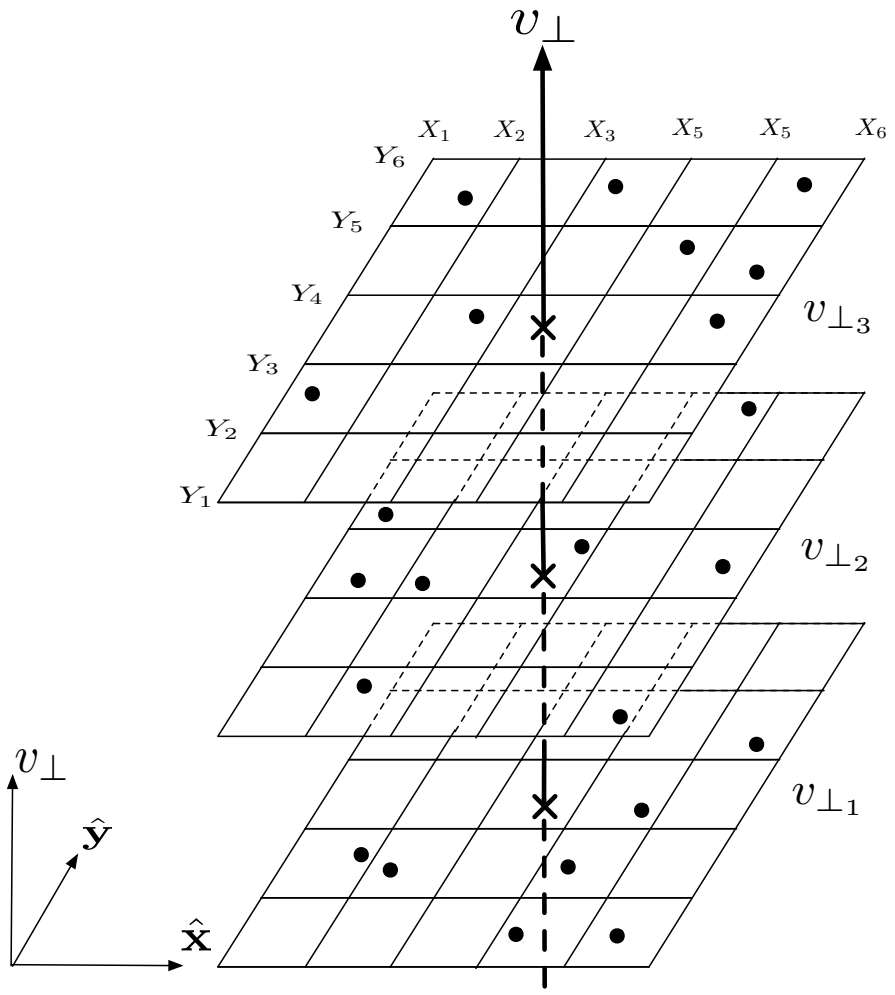

Figure 2.2: This sketch shows how we are dividing phase space in "slices" with constant value of $v_\perp$ to solve Poisson's equation. We interpo late the weights of a all particles that have the same value for $v_\perp$ onto a four-dimensional slice of phase-space. That four-dimensional slice consists of the three spatial dimensions and the parallel velocity. When solving Eq. (2.33) we interpolate the particles onto a grid in physical space while integrating out the the $v_\parallel$-dependence.

as $v_\perp$ increases, the maximum value for $v_\parallel$ decreases. In order to keep the particle density fixed in velocity space the number of particles has to decrease in the same

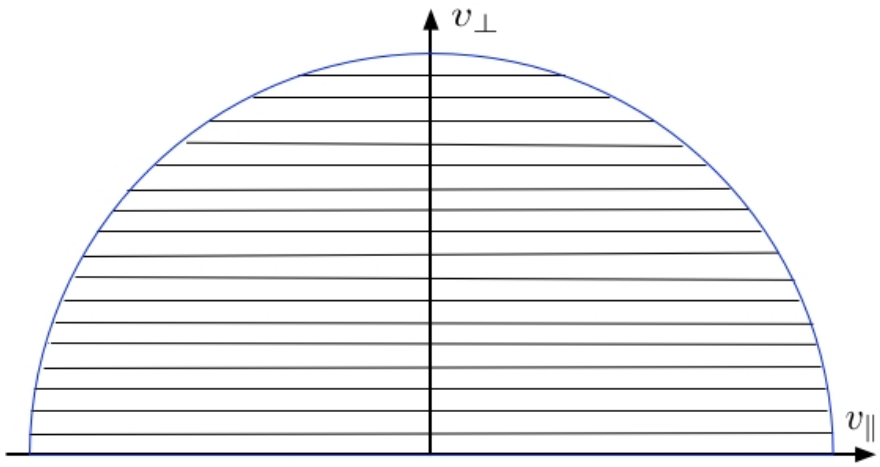

Figure 2.3: Velocity grid used in calculation of ϕ. The perpendicular velocities are initialized on an equally spaced grid while the parallel velocities are initialized grid-less. Hence, we have horizontal lines with a fixed value of $v_\perp$ on which the $v_\parallel$ values are initialized randomly. The energy of a particle is limited to be smaller than E_{max} . Therefore the coordinate-pair $(v_{\parallel i}, v_{\perp i})$ for a particle is forced to lie within a semicircle with radius E_{max}.

order as the maximum value of $v_\perp$ decreases while increasing the value of $v_\perp$. This means that the number of particles, $\tilde{N}(v_\perp)$, initialized for a given value of $v_\perp$ has to fulfill the following proportionality relationship

$$\tilde{N}(v_\perp) \sim \sqrt{E_{max} - v_\perp^2} \tag{2.34}$$

to ensures a constant particle density in $(v_\parallel, v_\perp)$.

There is a further complexity about the organization of the velocity part of phase space. As stated above the equilibrium part of the distribution function, F_0, is a Maxwellian in velocity space. Furthermore, one of the ordering assumptions

in the derivation of the gyrokinetic equation from the Fokker-Planck-Equation in chapter 1 is that the ratio of δf to F_0 is of order ϵ. To ensure that this ratio is satisfied for all value of v^2 we initialize the velocity dependence of δf to be also Maxwellian. Above we described that the parallel particle velocities are drawn from a uniform distribution and that we have a scheme that gives us a constant particle density in velocity space. Hence, the distribution of the particles is uniform and not Maxwellian. To obtain nevertheless a Maxwellian velocity distribution for δf we assign a "velocity-weight" for each individual particle. The "velocity-weights" for particle i are factors that are given by:

$$H_0(v_{\perp i}) = e^{-\frac{v_{\perp i}^2}{2v_T^2}},$$

$$H_0(v_{\parallel i}) = e^{-\frac{v_{\parallel i}^2}{2v_T^2}}.$$

We need to normalize $H_0(v_\perp)$ and $H_0(v_\parallel)$. This is done in the code for $H_0(v_\parallel)$ on a point by point basis in the 4-dimensional space $(x, y, z, v_\perp)$. Since we have a limited number of particles and a finite maximum value of $v_\parallel$ for each of those 4-dimensional points we do not expect to be able to fully resolve the Maxwellian with the number of particles that we have in the code. So instead of normalizing by the exact factor of $\frac{1}{\sqrt{2\pi}}$ for a Maxwellian distribution, we normalize by a factor that is a function of X, Y, Z and $v_\perp$ and is called $NORM_{v_\parallel}(X, Y, Z, v_\perp)$. It is a local estimate for how well we represent the Maxwellian distribution with a finite number of particles. The normalization function is given on the grid in physical space. That means that we need an interpolation from the particle positions to the grid-point coordinates when

solving for the integral while calculating $NORM_{v_\parallel}(X,Y,Z,v_\perp)$,

$$\begin{aligned} NORM_{v_\parallel}(X,Y,Z,v_\perp) &= \int H_0(v_\parallel)dv_\parallel|_{x,y,z,v_\perp} = \\ &= \int \sum_{i=1}^{N} \delta(v_\perp - v_{\perp i}) S(\mathbf{X}-\mathbf{R_i}) H_0(v_\parallel)dv_\parallel = \\ &= \int \sum_{i'=1}^{\tilde{N}(v_\perp)} g_{i',(j_x,j_y,j_z)} H_0(v_\parallel)dv_\parallel = \\ &= \sum_{i'=1}^{\tilde{N}(v_\perp)} g_{i',(j_x,j_y,j_z)} H_0(v_{\parallel_{i'}})\Delta v_\parallel. \end{aligned} \tag{2.35}$$

The definition for $\tilde{N}(v_\perp)$ is given in Eq.(2.34) as the number of particles that we have in the code for a fixed value of $v_\perp$. The interpolation weights $g_{i,j}$ are specified in Eq.(4.6). Note: the interpolation weights at a fixed grid-point (X,Y,Z) are going to be zero for a huge fraction of the $\tilde{N}(v_\perp)$ particles. That leaves us with a fairly small number of particles for which we are estimating $NORM_{v_\parallel}(X,Y,Z,v_\perp)$. But this is exactly the logic behind getting a local estimate for the normalization factor. All velocity integrals in $v_\parallel$ are done while holding a 4-dimensional grid-point in phase space fixed. Therefore the number of particles for which contribute to such an integral is reduced by substantial amount. We can reduce the error that we get from discrete particle noise by having a local estimate for the normalization. If the total number of particles N gets large enough the results in the code are independent whether we are using a local or a global normalization for the velocity integrals.

We use a similar approach for the normalization of $H_0(v_\perp)$. But since we have a grid for the $v_\perp$ values we don't have to solve a Monte-Carlo integration and the problem becomes quite easier. Nevertheless, since we have a finite number of grid-points and do have a upper limit for the maximum value of $v_\perp$ we are not going to

be able to numerically evaluate the integral of a Maxwellian to match the analytical result. Therefore we are not normalizing the velocity-weights $H_0(v_\perp)$ by 2π. Instead we normalize by the value we receive by numerically calculating the 1-dimensional integral $\int H_0(v_\perp)v_\perp dv_\perp$ on the $N_{v_\perp}$ grid points for $v_\perp$ which are equally spaced with an interval length of $\Delta v_\perp$:

$$NORM_{v_\perp} = \int H_0(v_\perp)v_\perp dv_\perp = \sum_i^{N_{v_\perp}} H_0(v_{\perp i})\Delta v_\perp. \tag{2.36}$$

2.3.2 Advantages of explicitly solving for the gyro-average

In the previous section we explained the numerically differences between evaluating the gyro-averages in Eq.(2.33) by either using a finite number of test-particles placed on a ring around $\mathbf{R_i}$ or by explicitly calculating the Bessel functions. We needed to initialize the particles on a grid in $v_\perp$ in order to be able to calculate $J_0(k_\perp v_\perp/\Omega)$. In other words we are solving Eq.(2.33) for 4-dimensional slices of phase-space with a fixed value of $v_\perp$ and then integrating over these slices. Numerically we have to pay the price that we need to take a Fourier transformation for each value of $v_\perp$ before we can integrate the results for the different phase-space slices with constant $v_\perp$. This means that we are have to perform $N_{v_\perp}$ Fourier Transformations instead of just one that is used in the ring-average approximation method. Recall $N_{v_\perp}$ is the number of grid-points for $v_\perp$. A typical value is $N_{v_\perp} \sim 32$. In

exchange for the numerical price we need to pay for this scheme we also get two major advantages. First, the number of interpolation steps we need to evaluate in the code is reduced by a factor of N_{ring}. Since the number of particles N is a very large number (for nonlinear runs of the order of several hundred million particles) even when N_{ring} is just equal to four it makes numerically a substantial difference to reduce the work needed for the interpolation step by a factor of four, since the overall algorithm scaling would be $\mathcal{O}(N_{ring}N)$. By contrast, although the cost of an FFT is non-trivial, there are not many extra to do, leading to a scaling of $\mathcal{O}(N) + \mathcal{O}(N_{v_\perp})$. Since for a large number of particles the interpolation is one of the bottlenecks of the algorithm this is an important advantage. The second benefit of our algorithm compared to the standard approach with test particles placed on a ring is that we get better accuracy. This advantage gets more significant as the value for $k_\perp \rho_i$ gets larger.

We can actually show this in a rigorous way by using the identity that was given in Eq.(2.32), $< e^{i\mathbf{k}\cdot\mathbf{r}} >_{\mathbf{R}} = J_0(k_\perp\rho_i)e^{i\mathbf{k}\cdot\mathbf{R}}$. We approximate the right hand side of this identity by using either four, eight or 16 points that we place on a ring with gyro-radius ρ_i around the gyro-center position $\mathbf{R}$. The algebra for the 4-point averaging leads to the following expression:

$$\begin{aligned} J_0(k_\perp\rho_i)e^{i\mathbf{k}\cdot\mathbf{R}} =< e^{i\mathbf{k}\cdot\mathbf{r}} >_{\mathbf{R}} &\approx \frac{1}{4}e^{i\mathbf{k}\cdot\mathbf{R}}\left(e^{ik_x\rho_i} + e^{-ik_x\rho_i} + e^{ik_y\rho_i} + e^{-ik_y\rho_i}\right) \\ &= \frac{1}{4}e^{i\mathbf{k}\cdot\mathbf{R}}\left(2\cos(k_x\rho_i) + 2\cos(k_y\rho_i)\right) \\ &\quad \text{with } k_x = k_y = \frac{1}{\sqrt{2}}k_\perp \Leftrightarrow \\ &= \cos\left(\frac{k_\perp\rho_i}{\sqrt{2}}\right)e^{i\mathbf{k}\cdot\mathbf{R}} \end{aligned} \tag{2.37}$$

Therefore the error from using the approximation that uses 4 test particles becomes the difference between $J_0(k_\perp\rho_i)$ and $\cos(k_\perp\rho_i/\sqrt{2})$. These two functions are both plotted in Fig(2.4) versus $k_\perp\rho_i$. In addition to the two functions we also plot the relative difference between them in Fig.(2.5). We do the same algebra for a ring-average that uses eight and 16 points. For the eight-point case we get:

$$\begin{aligned} J_0(k_\perp\rho_i) &\approx \frac{1}{8}\left(e^{ik_x\rho_i}+e^{-ik_x\rho_i}+e^{ik_y\rho_i}+e^{-ik_y\rho_i}\right)+ \\ &+ \frac{1}{8}\left(e^{i\left(\frac{k_x\rho_i}{\sqrt{2}}+\frac{k_y\rho_i}{\sqrt{2}}\right)}+e^{i\left(-\frac{k_x\rho_i}{\sqrt{2}}+\frac{k_y\rho_i}{\sqrt{2}}\right)}+e^{i\left(\frac{k_x\rho_i}{\sqrt{2}}-\frac{k_y\rho_i}{\sqrt{2}}\right)}+e^{i\left(-\frac{k_x\rho_i}{\sqrt{2}}-\frac{k_y\rho_i}{\sqrt{2}}\right)}\right) \\ &\quad \text{and with } k_x=k_y=\frac{1}{\sqrt{2}}k_\perp \Leftrightarrow \\ &= \frac{1}{8}\left(2+4\cos\left(\frac{k_\perp\rho_i}{\sqrt{2}}\right)+2\cos(k_\perp\rho_i)\right) \end{aligned} \tag{2.38}$$

And the same algebra done for the 16-point gyro-averaging approximation leads to:

$$\begin{aligned} J_0(k_\perp\rho_i) &\approx \frac{1}{16}\left(e^{ik_x\rho_i}+e^{-ik_x\rho_i}+e^{ik_y\rho_i}+e^{-ik_y\rho_i}\right)+ \\ &+ \frac{1}{16}\left(e^{i\left(\frac{k_x\rho_i}{\sqrt{2}}+\frac{k_y\rho_i}{\sqrt{2}}\right)}+e^{i\left(-\frac{k_x\rho_i}{\sqrt{2}}+\frac{k_y\rho_i}{\sqrt{2}}\right)}+e^{i\left(\frac{k_x\rho_i}{\sqrt{2}}-\frac{k_y\rho_i}{\sqrt{2}}\right)}+e^{i\left(-\frac{k_x\rho_i}{\sqrt{2}}-\frac{k_y\rho_i}{\sqrt{2}}\right)}\right) \\ &+ \frac{1}{16}\left(e^{i(\cos(\pi/8)k_x\rho_i+\sin(\pi/8)k_y\rho_i)}+e^{i(\cos(\pi/8)k_x\rho_i-\sin(\pi/8)k_y\rho_i)}\right)+ \\ &+ \frac{1}{16}\left(e^{i(-\cos(\pi/8)k_x\rho_i+\sin(\pi/8)k_y\rho_i)}+e^{i(-\cos(\pi/8)k_x\rho_i-\sin(\pi/8)k_y\rho_i)}\right) \\ &+ \frac{1}{16}\left(e^{i(\cos(3\pi/8)k_y\rho_i+\sin(3\pi/8)k_x\rho_i)}+e^{i(\cos(3\pi/8)k_y\rho_i-\sin(3\pi/8)k_x\rho_i)}\right)+ \\ &+ \frac{1}{16}\left(e^{i(-\cos(3\pi/8)k_y\rho_i+\sin(3\pi/8)k_x\rho_i)}+e^{i(-\cos(3\pi/8)k_y\rho_i-\sin(3\pi/8)k_x\rho_i)}\right) \\ &\quad \text{with } k_x=k_y=\frac{1}{\sqrt{2}}k_\perp \Leftrightarrow \\ &= \frac{1}{16}\left(2+4\cos\left(\frac{k_\perp\rho_i}{\sqrt{2}}\right)+2\cos(k_\perp\rho_i)\right)+ \\ &+ \frac{1}{16}\left(2\cos\left(\frac{k_\perp\rho_i}{\sqrt{2}}\left(\cos(\pi/8)+\sin(\pi/8)\right)\right)\right)+ \\ &+ \frac{1}{16}\left(2\cos\left(\frac{k_\perp\rho_i}{\sqrt{2}}\left(\cos(\pi/8)-\sin(\pi/8)\right)\right)\right)+ \end{aligned}$$

$$+ \quad \frac{1}{16}\left(2\cos\left(\frac{k_\perp\rho_i}{\sqrt{2}}\left(\cos\left(3\pi/8\right)+\sin\left(3\pi/8\right)\right)\right)\right)+$$
$$+ \quad \frac{1}{16}\left(2\cos\left(\frac{k_\perp\rho_i}{\sqrt{2}}\left(\cos\left(3\pi/8\right)-\sin\left(3\pi/8\right)\right)\right)\right) \tag{2.39}$$

The results for the eight and 16-point estimates are plotted along with the four-point estimate in Fig(2.4) and Fig.(2.5). These plots show for which range of values of $k_\perp\rho_i$ the ring-averages are good approximations and when they become inaccurate. For values of $k_\perp\rho_i > 2$ the error from using four-averages get significant and the method seems to be no longer valid. For eight-point averages this is the case for $k_\perp\rho_i > 5$ and for the 16-point average for $k_\perp\rho_i > 12$. This is an important result since in most other gyrokinetic PIC-codes the default setting is $N_{ring} = 4$ and they are able just to resolve problems up to $k_\perp\rho_i$ of 2.

This becomes of importance when we study in chapter 5.5 instabilities in a Z-pinch configuration. As discussed by Ricci *et al.,* [53, 54] there exists a regime for which the ideal interchange mode is stable but leaves behind a non-MHD mode known as the entropy mode (also referred to as the drift-temperature-gradient mode) [38, 59]. Since it was found [38, 39, 59, 60] that the growth rate of the entropy modes linearly dependent on k codes that are restricted to a regime with $k\rho_i \sim 1$ or even $k\rho_i \ll 1$ will not be able to simulate the most unstable regime of the entropy mode. So explicitly this means that δf-PIC codes using 4-point averaging method for calculating gyro-averaged quantities (and are therefore restricted to regimes with $k\rho_i < 2$) cannot be used for studying entropy modes in a Z-pinch configuration. A similar restriction likely applies to trapped electron modes in the tokamak.

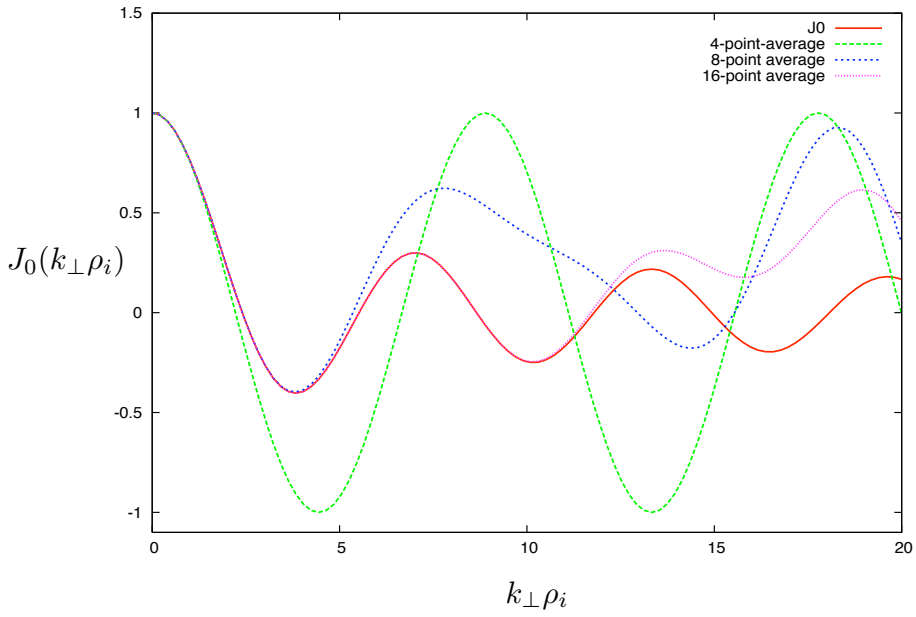

Figure 2.4: Comparision of the zeroth ordered Bessel Function to approximations with different number of test particles. We vary the number N_{ring} of test particles that we use to approximate $J_0(k_\perp \rho_i)$ and plot the expressions Eq.(2.37), Eq.(2.38) and Eq.(2.39) that we obtained by evaluating the gyro-averaging estimates with four, eight and 16 test particles respectively. In comparison to the approximations we also plot the analytical form of $J_0(k_\perp \rho_i)$ versus $k_\perp \rho_i$.

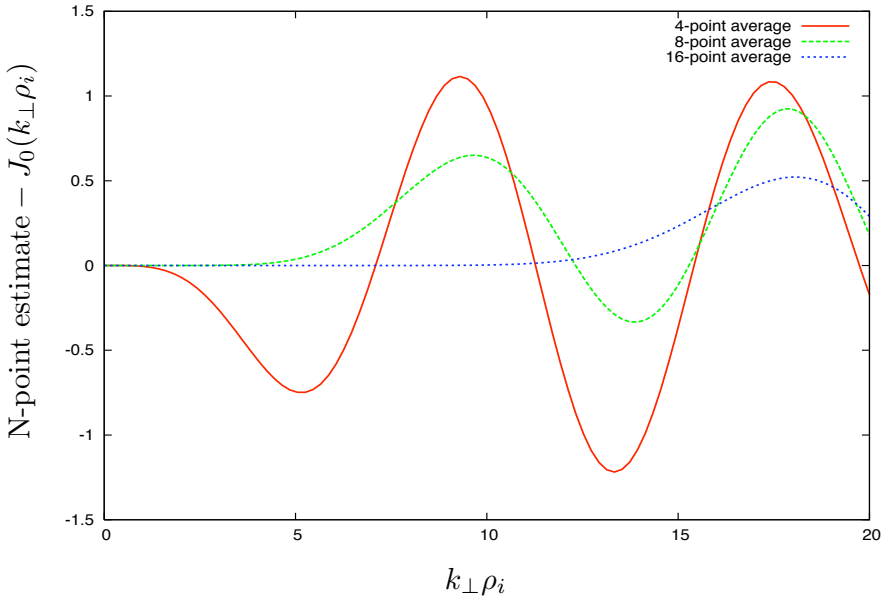

Figure 2.5: Difference between test particle estimates and the analytical form of $J_0(k_\perp\rho_i)$. The 4-point approximation starts to differ significantly from $J_0(k_\perp\rho_i)$ at a value of $k_\perp\rho_i \sim 2$. For the 8-point approximation the approximation becomes inaccurate for $k_\perp\rho_i \sim 5$ while for the 16-point approximation is in agreement with the analytical form of $J_0(k_\perp\rho_i)$ until a value of $k_\perp\rho_i \sim 12$.

Chapter 3

Challenges for particle simulations

As described in the previous chapter we need to calculate the electrostatic potential ϕ in the code. In order to do this we integrate out the velocity part of the gyro-averaged 5D perturbed distribution function δf. The accuracy of this operation is of crucial importance for the accuracy of the overall algorithm. Since we represent the distribution function with a finite number of particles that we interpolate onto a grid, we have to deal with discrete particle noise. As mentioned above we reduce the discrete particle noise by a significant amount by representing only the perturbed distribution δf, which is the departure from the full distribution function, the Maxwellian background solution. But the scheme is dependent on the local variance of the particle weights. When the particle weights have a variance that is too large we cannot assume that the code is still correctly resolving the problem. In this chapter we are developing a measure for the local variance in the code and show how we can relate our estimate for the variance to an estimate for the discrete particle noise error in the code.

3.1 General properties of the Monte-Carlo integration schemes

It is a well known problem that the accuracy of the local estimator for $\delta f(\mathbf{x}, \mathbf{v})$ at a given point on the 5D phase-space grid is facing challenges from discrete particle noise [48] [9]. Aydemir wrote the seminal paper on applications of Monte-Carlo integration schemes for particle simulations [2]. He found the estimate for the error ϵ of an general integral that calculates a moment of the form

$$\mathcal{I}(A) = \int_V A(\mathbf{q}) f(\mathbf{q}) d\Gamma$$

to be:

$$\epsilon \simeq \frac{\sigma_{\delta f}}{\sqrt{N}}$$

with N, the total number of random samples and the variance $\sigma_{\delta f}$ given by:

$$\sigma^2_{\delta f} = \int_V \left(\delta g - < \delta g >\right)^2 p(\mathbf{q}) d\Gamma \tag{3.1}$$

$$\text{with:} \quad \delta g \equiv \frac{A(\mathbf{q})\delta f(\mathbf{q})}{p(\mathbf{q})}; \quad p(\mathbf{q}) \text{ probability density} \tag{3.2}$$

$< \delta g >$ is the expected value of the random variable $\delta g \equiv A(\mathbf{q} f(\mathbf{q}/p(\mathbf{q})$.

$$\mathcal{I}(A) =< \delta g >= \int_V \frac{A(\mathbf{q}) f(\mathbf{q})}{p(\mathbf{q})} p(\mathbf{q}) d\Gamma \tag{3.3}$$

So we need to translate this error estimate for a general Monte-Carlo integral into an error estimates for our problem. We are mostly concerned with the error we get when solving for ϕ. Discrete particle noise will effect the estimator for δf on the grid and carry over as an error to ϕ. As a consequence the values for the particle

drift velocities $\mathbf{v}_{E\times B}$ will be affected by the accuracy of this problem as well. In Eq.(2.33) we are evaluating integrals of the form: $\int \delta f d\Gamma$. Since the definition of the particle weights in a δf-method is $w_i = \delta f/F_0$ the integrals we are concerned about are proportional to the integral of the average weights of the particle.

$$\int \delta f d\mathbf{v} \sim \frac{1}{N}\sum_{i=1}^{N} w_i =< w > \tag{3.4}$$

and the variance for this becomes:

$$\sigma_w^2 = \frac{1}{N}\sum_{i=i}^{N}(w_i - < w >)^2 =< w^2 > - < w >^2 \tag{3.5}$$

As was pointed out by Aydemir [2] we are not interested in the error in absolute terms. Instead we want to know how big the error is compared to the actual value of the integral to have a measure for how accurate our scheme is. Hence, we define the relative variance $\sigma_{w,r}^2$:

$$\sigma_{w,r}^2 = \frac{1}{N}\sum_{i=i}^{N}\frac{(w_i - < w >)^2}{\bar{w}} = \frac{1}{< w >}\left(< w^2 > - < w >^2\right) \tag{3.6}$$

We are interested in understanding how the error estimate is evolving over time. With the fully nonlinear code we are studying physical quantities such as the heat flux. We would like to determine when the code reaches saturation level, and is out of the linear growth phase. In such a phase the exponential growth of $\bar{w}$ has stopped. Hence we assume that $\frac{d}{dt} < w >= 0$ and the time-behavior of Eq.(3.6) is then given by:

$$\frac{d}{dt}\sigma_{w,r}^2 = \frac{2}{N < w >}\sum_{i=1}^{N}\left(w_i\frac{d}{dt}w_i\right)$$

Thus, the change of the variance is given by the time evolution of the mean-squared weights $< w^2 >$. In the previous chapter we described the δf-method for solving the gyrokinetic equation with a PIC code. We explained that the particle weights are treated as an independent variable and evolved over time accordingly to Eq.(2.21). This has direct consequences to the time behavior of the mean-squared particle weight $< w^2 >$:

$$\begin{aligned}
&\frac{d}{dt} < w^2 > = \frac{2}{N} \sum_{i=1}^{N} w_i \frac{d}{dt} w_i = \\
= \; & -\frac{2}{N} \sum_{i=1}^{N} w_i \left(\underbrace{< \mathbf{v}_{\mathbf{E}\times\mathbf{B_i}} >_{\mathbf{R}} \cdot \hat{\mathbf{x}}}_{v_{Dx_i}} \left(-\frac{1}{L_n} + \frac{3}{2L_T} - \frac{v_i^2}{2L_T v_{T_i}^2} \right) + \frac{q}{T_0} v_{\|_i} < E_{\|_i} >_{\mathbf{R}} \right) \\
= \; & 2 \left(\frac{1}{L_n} - \frac{3}{2L_T} \right) \Gamma_p + \frac{2}{L_T} \Gamma_e + \frac{2q}{N T_0} \sum_{i=1}^{N} w_i v_{\|_i} < E_{\|_i} >_{\mathbf{R}} \qquad (3.7) \\
& \text{with the particle flux :} \quad \Gamma_p \equiv \frac{1}{N} \sum_{i=1}^{N} w_i v_{Dx_i} \qquad (3.8) \\
& \text{and the energy flux :} \quad \Gamma_e \equiv \frac{1}{N} \sum_{i=1}^{N} w_i v_i^2 v_{Dx_i} \qquad (3.9)
\end{aligned}$$

Lee and Tang [45] relate the rate of increase in the $< w^2 >$ in the entropy theorem to the energy flux which they identify as the dominant term in Eq.(3.7) as:

$$\frac{d < w^2 >}{dt} \sim \frac{2\Gamma_e}{L_T} \qquad (3.10)$$

This means that the relative error is also going to show the same monotonically increasing behavior. So it will be increasing in time without a bound and eventually we won't be able to trust the outcome of the simulation anymore. When we are

using a large number of particles the discrete particle noise is initially reduced, and it will take a longer time until the error becomes too big. But nevertheless increasing the number of particles does not solve the problem of having an increase in the mean-squared weights and therefore an increase in the discrete particle noise over time. This is particularly important for the study of problems for which the algebraic growth of the weights is fast, or when one needs to integrate over many nonlinear eddy turnover times.

In chapter 4.1.2 we use GSP to study ITG driven turbulence. We have to run the code nonlinearly until it reaches a saturated state. We then want to determine the amount of heat flux that is present in the saturated nonlinear state. By the time that we reach saturation the level of noise that is present in the system must be low enough to not affect the accuracy of the measurement of the heat flux.

The above calculation does not look at size of the structures that are dominant in the simulations that need to be resolved. Although the increase in the mean-squared weight is seen in PIC δf-codes, the results for the Cyclone base case are well benchmarked against continuum codes. We hypothesize that this comes from the fact that the dominant spatial structures that develop in the Cyclone base case are much larger than the spatial grid size in the simulations, are isotropic in the perpendicular plane, and that the codes do not run long enough for the increase in the variance of the weights to have a significant effect on the measured physical quantities. These hypotheses were addressed in Nevins, *et al.* [48]

In Fig.(3.1 a)) and Fig.(3.1 b)) we show in a cartoon the effect that the growth for the variance of the weights has. On the abscissa we have a 1-dimensional subset

of particle positions in the 5-dimensional phase space. A vector in the 5-dimensional phase space is symbolized by $\mathbf{q}$. The particles are scattered in such a space. We also introduce a grid in the 5-D space onto which we will interpolate the particle weights. The value of the particle weights are represented on the ordinate. We show a representation of the particle weights on a grid-points that are symbolized by capital letters, $\mathbf{Q_j}$. We interpolate the weights of individual particles at position $\mathbf{q_i}$ to find the weighted average weights on the grid. The symbol for the weights of individual particles are crosses and the value for the weights on the grid-points are marked by circles. Fig.(3.1 a)) and Fig.(3.1 b)) both show such weight distribution in a steady state of the simulation but Fig(3.1 b)) shows the same system at a later time. In this cartoon the variance increases from the earlier time in a) to the later time in b) although the values on the grid haven't changed by much. Therefore this is a cartoon where the increase in the variance has not affected the estimate of the weight distribution too strongly. But we can see how on a finer grid in $\mathbf{q}$ the growth in the variance of the weights would have changed the estimate for the interpolated weights on the grid. One can also see that at a time which is even later, the monotone increase in the variance will eventually lead to growing error for $\bar{w}_\mathbf{j}$.

To test this hypothesis we are now developing a method that measures the variance as a local measure that depends on the scale that we are trying to resolve. We are going to look more carefully at the fact that we are using an interpolation algorithm in Eq.(2.33) when we are finding ϕ.

When we are now going to talk about the accuracy of Eq.(3.4) as a measure of how well local structures can be resolved we change to a new set of velocity variables.

We do this since we are going to use the same variables for the collision operator that we are going to introduce in the next chapter.

3.2 Pitch-angle, Energy coordinates

The two new variables are energy, $E = v^2 = v_\perp^2 + v_\parallel^2$ and pitch-angle $\xi = v_\parallel / v$. In Fig(3.2) we show a sketch of the velocity part of the 5D grid in these new variables.

The grid-points are placed in the old $v_\perp, v_\parallel$-grid on arcs which represent a constant value for the energy E. On those arcs we place the grid-points for different values of the pitch-angle. The arc-length between grid-points on those semi-circles is held fixed for a given value of E. The distance between two of these semi-circles with constant energy is $2\Delta E$. The lowest energy band is placed at a value of $E = \Delta E$. We have a total of N_E different energy-grid-points. For labeling the grid-points (X, Y, Z, E, ξ) we define the five dimensional index vector $\mathbf{j} = (j_x, j_y, j_z, j_E, j_\xi)$. In Fig.(3.2) we fix the position in physical space and all the grid-points shown have the same indices j_x, j_y, j_z. To find the value of the perturbed distribution function we use the bilinear interpolation scheme for the physical space with the interpolation weights $g_{i,jx,jy,jz}$ that were defined in Eq.(4.6) and a nearest neighbor interpolation in E and ξ. This means that all particles that have energy values that fall into the area that is given by $|E_i - E_{j_E}| \leq \Delta E$ will be projected onto a grid-point that lies on the semi-circle with radius E_{j_E}. The number of ξ grid points we are placing onto an arc with constant energy E is increased when we go to higher energy values E_{j_E}. The rationale behind putting more grid-points on arcs that

represent higher energy values is that we want to keep the volume fixed that is used for interpolating particles onto a specific grid-point. Since the particles were initialized on the $v_\parallel, v_\perp$-grid with a constant density, a constant interpolation volume relates to an approximately constant number of particles that are interpolated onto a gridpoint. Since the discrete particle noise scales with the number of particles used to determine the value of the function we are integrating, the property of roughly equal numbers of particles is desirable. To ensure a constant interpolation volume we find the following recursive relationship for the number of pitch-angle grid-points $N_\xi(j_E)$ as a function of the energy index, j_E. The calculation is based on setting the volumes for two interpolation volumes equal on subsequent energy grid-points, j_E and j_{E+1} and finding a relationship for the number of ξ-grid-points on those energy values.

$$\begin{aligned} \frac{(2j_E\Delta E)^2 - (2(j_E-1)\Delta E)^2}{N_\xi(j_E)} &= \frac{(2(j_E+1)\Delta E)^2 - (2j_E\Delta E)^2}{N_\xi(j_E+1)} \\ \Leftrightarrow \quad \frac{N_\xi(j_E+1)}{N_\xi(j_E)} &= 4\,\frac{2j_E+1}{2j_E-1} \end{aligned} \tag{3.11}$$

By choosing the number of grid-points on the lowest energy level, $N_\xi(j_E = 1)$, the remaining values for $N_\xi(j_E)$ are then specified by the recursive relationship in Eq.(3.11). Having the new velocity-space representation defined we can get back to the task of trying to calculating the accuracy of Eq.(3.4).

3.3 Discrete particle noise and resolving phase space

For the new coordinate system we are defining a variance that is dependent on how coarse we make the grid. This is an attempt to capture how the discrete particle noise has an impact on how well we can resolve small scale structures (in x and v) with the code. With the new velocity coordinates that we defined in section 3.2 we have a total of N_{5D} grid-points. Nx, Ny and Nz are the number of grid-points that we have in the spatial part of phase space in the $x-, y-$ and $z-$direction respectively.

$$N_{5D} = NxNyNz \left(\sum_{j_E=1}^{N_E} N_\xi(j_E) \right) \tag{3.12}$$

We now define now a local variance of the weights for one of those N_{5D} grid-points. The local variance is the weighted variance of those weights that have a finite interpolation weight onto a grid-point that is marked by the index vector $\mathbf{j}$. We are using nearest neighbor grid-point interpolation for the velocity components of phase space and bilinear interpolation for the spatial components of phase space. The bilinear interpolation uses the interpolation weights that were defined in Eq.(4.6),

$$\sigma^2_{\delta f, j_E, j_\xi, j_x, j_y, j_z} = \sigma^2_{\delta f, \mathbf{j}} = \frac{\sum_{i=1}^{N} \left(w_i - \bar{w}_j\right)^2 g_{i,j_{x,y,z}} \delta_{i,j_E} \delta_{i,j_\xi}}{\sum_{i=1}^{N} g_{i,j_{x,y,z}} \delta_{i,j_E} \delta_{i,j_\xi}}, \tag{3.13}$$

with:

$$\bar{w}_{\mathbf{j}} = \frac{\sum_{i=1}^{N} g_{i,j_{x,y,z}} \delta_{i,j_E} \delta_{i,j_\xi} w_i}{\sum_{i=1}^{N} g_{i,j_{x,y,z}}}, \tag{3.14}$$

$$\text{with :} \qquad \delta_{i,j_E} = \begin{cases} 1 & : \quad |E_i - E_{j_E}| \leq \Delta E \\ 0 & : \quad \text{else,} \end{cases} \tag{3.15}$$

$$\text{and :} \qquad \delta_{i,j_\xi} = \begin{cases} 1 & : \quad |\xi_i - \xi_{j_\xi}| \leq \Delta \xi \\ 0 & : \quad \text{else.} \end{cases} \tag{3.16}$$

The total variance then becomes the sum of the local variances divided by the number of grid-points, N_{5D},

$$\sigma^2_{\delta f} = \frac{1}{N_{5D}} \sum_{j_E=1}^{N_E} \sum_{j_\xi=1}^{N_\xi(j_E)} \sum_{j_x=1}^{Nx} \sum_{j_y=1}^{Ny} \sum_{j_z=1}^{Nz} \sigma^2_{\delta f,\mathbf{j}} = \sum_{\mathbf{j}} \sigma^2_{\delta f,\mathbf{j}}. \tag{3.17}$$

From here on we are going to use the symbol $\sum_{\mathbf{j}}$ which stands for the sum over all 5 j-indices j_x, j_y, j_z, j_E and j_ξ and also contains the factor $\frac{1}{N_{5D}}$.

Here again we are going to define a relative variance as a better measure for the accuracy of the integration scheme.

$$\sigma^2_{\delta f,r} = \frac{\sum_{\mathbf{j}} \sigma^2_{\delta f,\mathbf{j}}}{\sum_{\mathbf{j}} \bar{w}_j} \tag{3.18}$$

This allows us to define the relative error of Eq.(3.4) as:

$$\epsilon_r = \frac{\sigma_{\delta f,r}}{\sqrt{N}} \tag{3.19}$$

Now we have to redo the exercise of trying to find how this error estimate will evolve over time. To be able to do this we define the mean-squared weights $< w^2 >$ as the sum of the local mean-squared weights $< w^2 >_{\mathbf{j}}$.

$$< w^2 > = \sum_{\mathbf{j}} \frac{\sum_{i=1}^{N} w_i^2 g_{i,j_{x,y,z}} \delta_{i,j_E} \delta_{i,j_\xi}}{\sum_{i=1}^{N} g_{i,j_{x,y,z}} \delta_{i,j_E} \delta_{i,j_\xi}} = \sum_{\mathbf{j}} < w^2 >_{\mathbf{j}} \tag{3.20}$$

We are mostly concerned with noise in the code once the nonlinear simulation reaches a steady state and we want to calculate physical quantities like the heat flux from the code. So we are here now trying to estimate the time dependence of the noise in the code for a steady state. Since the weights are scaled to be $\mathcal{O}(\frac{\rho}{a})$ the nonlinear and linear terms become of the same order once the estimates for the weights are of order one. This is at the time when we reach a fully nonlinear state and we can therefore then make estimate that $(\sum_{\mathbf{j}} \bar{w}_{\mathbf{j}} \approx 1)$.

The time evolution of the relative variance $\sigma^2_{\delta f,r}$ in a steady nonlinear state is then

$$\frac{d}{dt}\sigma^2_{\delta f,r} = \frac{\sum_{\mathbf{j}} \frac{d}{dt}\sigma^2_{\delta f,\mathbf{j}}}{\sum_{\mathbf{j}} \bar{w}_{\mathbf{j}}} - \sigma^2_{\delta f,\mathbf{j}} \frac{\sum_{\mathbf{j}} \frac{d}{dt}\bar{w}_{\mathbf{j}}}{\left(\sum_{\mathbf{j}} \bar{w}_{\mathbf{j}}\right)^2}. \tag{3.21}$$

Since we are looking at a steady state we assume that the sum of the variations of the local changes of the average weights on the grid-points will evaluate to a very small term. Therefore we will drop the second term in Eq.(3.21) and use $(\sum_{\mathbf{j}} \bar{w}_{\mathbf{j}} \approx 1)$:

$$\begin{aligned}
\frac{d}{dt}\sigma^2_{\delta f,r} &\approx \sum_{\mathbf{j}} \frac{d}{dt}\left(\frac{\sum_{i=1}^{N}\left(w_i - \bar{w}_{\mathbf{j}}\right)^2 g_{i,j_{x,y,z}}\delta_{i,j_E}\delta_{i,j_\xi}}{\sum_{i=1}^{N} g_{i,j_{x,y,z}}\delta_{i,j_E}\delta_{i,j_\xi}}\right) \\
&= \sum_{\mathbf{j}}\left(\frac{\sum_{i=1}^{N} 2\left(w_i - \bar{w}_{\mathbf{j}}\right)\left(\frac{d}{dt}w_i - \frac{d}{dt}\bar{w}_{\mathbf{j}}\right) g_{i,j_{x,y,z}}\delta_{i,j_E}\delta_{i,j_\xi}}{\sum_{i=1}^{N} g_{i,j_{x,y,z}}\delta_{i,j_E}\delta_{i,j_\xi}} + \right. \\
&+ \frac{\sum_{i=1}^{N}\left(w_i - \bar{w}_{\mathbf{j}}\right)^2 \frac{d}{dt}g_{i,j_{x,y,z}}\delta_{i,j_E}\delta_{i,j_\xi}}{\sum_{i=1}^{N} g_{i,j_{x,y,z}}\delta_{i,j_E}\delta_{i,j_\xi}} - \\
&- \left.\frac{\sum_{i=1}^{N}\left(w_i - \bar{w}_{\mathbf{j}}\right)^2 g_{i,j_{x,y,z}}\delta_{i,j_E}\delta_{i,j_\xi}\sum_{i=1}^{N}\frac{d}{dt}g_{i,j_{x,y,z}}\delta_{i,j_E}\delta_{i,j_\xi}}{\left(\sum_{i=1}^{N} g_{i,j_{x,y,z}}\delta_{i,j_E}\delta_{i,j_\xi}\right)^2}\right).
\end{aligned} \tag{3.22}$$

The changes in $g_{i,j}$ over time that come from the fact that the particles follow their characteristics should not have an effect that influences the variance over time since we do not expect to see a local accumulation of particles. Instead the

particles should keep filling out the physical part of phase space rather uniformly. So the changes from $\frac{d}{dt}g_{i,j}$ average to zero. That leaves us with the first term in Eq.(3.22) which has the four components (we are using the notation $\frac{d}{dt}a = \dot{a}$): $+w_i\dot{w}_i, -\bar{w}_{\mathbf{j}}\dot{w}_i, -w_i\dot{\bar{w}}_{\mathbf{j}}, +\bar{w}_{\mathbf{j}}\dot{\bar{w}}_{\mathbf{j}}$. The last two terms have opposite signs and when we evaluate the inner sum over i they cancel out. So we receive the following estimate for the time dependance of the variance:

$$\begin{aligned}\frac{d}{dt}\sigma^2_{\delta f,r} &\approx 2\sum_{\mathbf{j}} \frac{\sum_{i=1}^N \left(w_i\dot{w}_i - \bar{w}_{\mathbf{j}}\dot{w}_i\right) g_{i,j_{x,y,z}}\delta_{i,j_E}\delta_{i,j_\xi}}{\sum_{i=1}^N g_{i,j_{x,y,z}}\delta_{i,j_E}\delta_{i,j_\xi}} \\ &= 2\sum_{\mathbf{j}} \left(\frac{\sum_{i=1}^N w_i\dot{w}_i g_{i,j_{x,y,z}}\delta_{i,j_E}\delta_{i,j_\xi}}{\sum_{i=1}^N g_{i,j_{x,y,z}}\delta_{i,j_E}\delta_{i,j_\xi}} + \bar{\dot{w}}_{\mathbf{j}}\right). \end{aligned} \tag{3.23}$$

The second term is very small for a steady-state since the sum over all changes of the average values of the weights on the grid-points will nearly cancel out. Otherwise we wouldn't have reached a steady state yet. This means that the relative variance changes over time as the sum of the change of the local mean-squared weights divided by the local mean weight. Local refers here to the values we find from interpolation onto a grid-point. So we get the following estimate for the time evolution of the relative variance of the weights:

$$\frac{d}{dt}\sigma^2_{\delta f,r} \approx 2\sum_{\mathbf{j}} \frac{\sum_{i=1}^N w_i\dot{w}_i g_{i,j_{x,y,z}}\delta_{i,j_E}\delta_{i,j_\xi}}{\sum_{i=1}^N g_{i,j_{x,y,z}}\delta_{i,j_E}\delta_{i,j_\xi}}. \tag{3.24}$$

As in the previous section we can use the time derivative of the weights, described in Eq.(2.21) to plug into Eq.(3.24). This yields

$$\begin{aligned}&\frac{d}{dt}\sigma^2_{\delta f,r} \approx \\ &\sum_{\mathbf{j}} \frac{-2\sum_{i=1}^N w_i \left(v_{Dx_i}\left(-\frac{1}{L_n} + \frac{3}{2L_T} - \frac{v_i^2}{2L_T v_{T_i}^2}\right) + \frac{qv_{\|_i}}{T_0} < E_{\|_i} >_{\mathbf{R}}\right) g_{i,j_{x,y,z}}\delta_{i,j_E}\delta_{i,j_\xi}}{\sum_{i=1}^N g_{i,j_{x,y,z}}\delta_{i,j_E}\delta_{i,j_\xi}}\end{aligned} \tag{3.25}$$

$$= 2\sum_{\mathbf{j}} \left(\left(\frac{1}{L_n} - \frac{3}{2L_T} \right) \Gamma_{p,\mathbf{j}} + \frac{1}{L_T} \Gamma_{e,\mathbf{j}} + \frac{\sum_{i=1}^{N} w_i \frac{q v_{\parallel_i}}{T_0} < E_{\parallel_i} >_{\mathbf{R}} g_{i,j_{x,y,z}} \delta_{i,j_E} \delta_{i,j_\xi}}{\sum_{i=1}^{N} g_{i,j_{x,y,z}} \delta_{i,j_E} \delta_{i,j_\xi}} \right),$$

with the drift velocity in the x-direction: $v_{Dx_i} =< \mathbf{v}_{\mathbf{E}\times\mathbf{B_i}} >_{\mathbf{R}} \cdot \hat{\mathbf{x}}$ (3.26)

with the particle flux : $\Gamma_{p,\mathbf{j}} \equiv \frac{\sum_{i=1}^{N} w_i v_{Dx_i} g_{i,j_{x,y,z}} \delta_{i,j_E} \delta_{i,j_\xi}}{\sum_{i=1}^{N} g_{i,j_{x,y,z}} \delta_{i,j_E} \delta_{i,j_\xi}}$ (3.27)

and the energy flux : $\Gamma_{e,\mathbf{j}} \equiv \frac{\sum_{i=1}^{N} w_i v_i^2 v_{Dx_i} g_{i,j_{x,y,z}} \delta_{i,j_E} \delta_{i,j_\xi}}{\sum_{i=1}^{N} g_{i,j_{x,y,z}} \delta_{i,j_E} \delta_{i,j_\xi}}.$ (3.28)

Also for this more complicated system the arguments from Lee and Tang [45] will hold that the term involving the energy flux must be the dominant term. So we can simplify Eq.(3.25) in the same way as we did for Eq.(3.7) when we found Eq.(3.10). This results in the following estimate for the time evolution of the relative variance of the weights

$$\frac{d}{dt}\sigma^2_{\delta f,r} \approx \sum_{\mathbf{j}} \frac{2}{L_T} \Gamma_{e,\mathbf{j}}. \tag{3.29}$$

Note that if we had just one grid-point then the Eq.(3.29) reduces to Eq.(3.10). So with Eq.(3.29) we then get the following estimate for the error of the integrals that we are evaluating in the code,

$$\epsilon_r = \frac{1}{\sqrt{N}} \sum_{\mathbf{j}} \sigma_{\delta f,\mathbf{j}}, \tag{3.30}$$

$$\text{and :} \quad \frac{d}{dt}\epsilon_r = \frac{1}{\sqrt{N}} \sum_{\mathbf{j}} \frac{2}{L_T} \Gamma_{e,\mathbf{j}}. \tag{3.31}$$

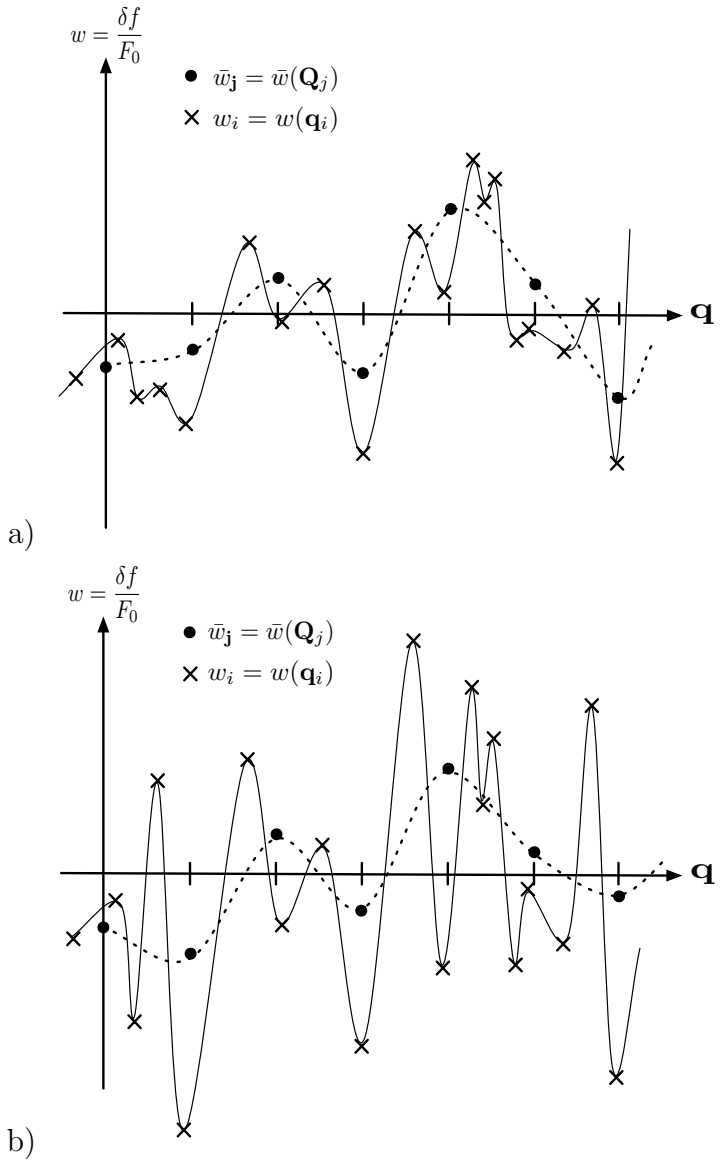

Figure 3.1: Cartoon of the weight distribution in phase space shown at different times in the nonlinear steady state phase of the simulation. The circles stand for the value of the interpolated weights on the grid-points $\mathbf{Q_j}$ and the crosses the value of the weight of particle i at the 5D phase space position $\mathbf{q_i}$. The variance of the weights increases form an earlier time that is shown in a) to a later time shown in b)

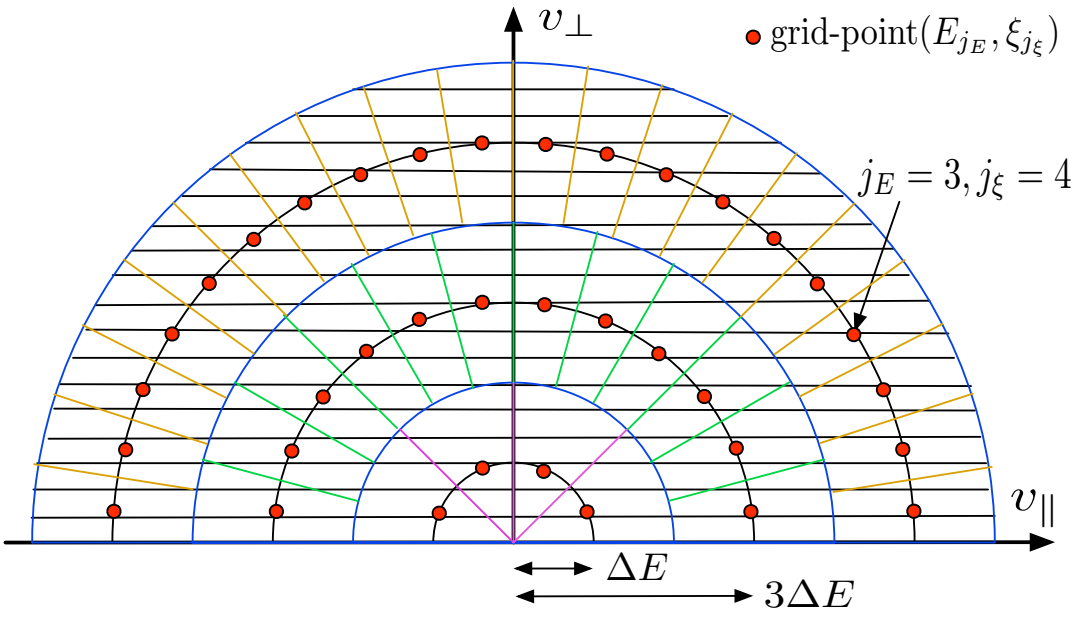

Figure 3.2: Sketch of the velocity grid using the coordinates energy $E = v_\perp^2 + v_\parallel^2$ and pitch-angle $\xi = \frac{v_\parallel}{v}$. The grid-points (E_{j_E}, ξ_{j_ξ}) lie on arcs with constant values of energy. The spacing between those semicircles of constant energy is $2\Delta E$, with ΔE being a parameter that we is set in the input file of our code. The number of grid-points with the same energy value but different values for the pitch-angle increases as the energy is increased and is determined by the recursive relationship in Eq.(3.11).

Chapter 4

Coarse-graining phase space

The fact that the particle weights grow continuously in a saturated state of a δf PIC-code simulation as we described in the previous chapter is well-known and has been addressed in the literature as the "growing weight problem" [41, 42, 48].

Chen and Parker [14] addressed the problem by introducing a coarse-graining algorithm. The basic idea behind their algorithm is to obtain δf on a 5-dimensional phase space grid and then to reset the particle weights according to the phase-space values. This will result in "averaging" particle weights that are close to the same grid-point in phase space. When the weights are reset the particle's spatial and velocity coordinates remain unchanged.

4.1 Resetting particle weights

In the above described method, one finds the average weights on the grid-points by interpolation and the resets the particle weights at the particle position using the same interpolation scheme. The formal description of this method is governed by the following equations. First we find the average weight on the N_{5D} phase-space

grid points. Chen and Parker [14] proposed the use of pitch-angle/energy velocity coordinates like those we introduced in section 3.2 (although the details differ). So we apply the same 5-dimensional index vector $\mathbf{j} = (j_x, j_y, j_z, j_E, j_\xi)$ that we used before to identify the grid-points, to find the best estimate of δf at $\mathbf{j}$:

$$\bar{w}_{\mathbf{j}} = \frac{\sum_{i=1}^{N} w_i g_{i,\mathbf{j}}}{\sum_{i=1}^{N} g_{i,\mathbf{j}}}. \tag{4.1}$$

The particle weights then get reassigned to:

$$w_i' = \sum_{j=1}^{N_{5D}} g_{i,\mathbf{j}} \bar{w}_{\mathbf{j}}. \tag{4.2}$$

We can choose any interpolation scheme. Here we are going to show two versions. First, we show this algorithm with nearest-grid-point interpolation, and second, the coarse-graining method that uses a bilinear interpolation method.

4.1.1 Coarse-graining with nearest-neighbor interpolation

The easiest interpolation scheme is a nearest grid-point interpolation. The interpolation weights $g_{i,\mathbf{j}}$ then become:

$$g_{i,\mathbf{j}}^{NGP} = \begin{cases} 1 & : \quad |q_{i,j} - Q_j| \leq \frac{\Delta Q_j}{2} \quad \text{with:} \quad j = j_x, j_y, j_z, j_E, j_\xi \\ 0 & : \quad \text{else.} \end{cases} \tag{4.3}$$

With Q_j we describe the position j^{th} component of the phase space grid-points, where j is describes one of the 5 elements of the index-vector $\mathbf{j}$. The position of a particle i in phase space is given by the vector $\mathbf{q}_{i,\mathbf{j}} = (q_{i,j_x}, q_{i,j_y}, q_{i,j_z}, q_{i,j_E}, q_{i,j_\xi}) = (x_i, y_i, z_i, E_i, \xi_i)$.

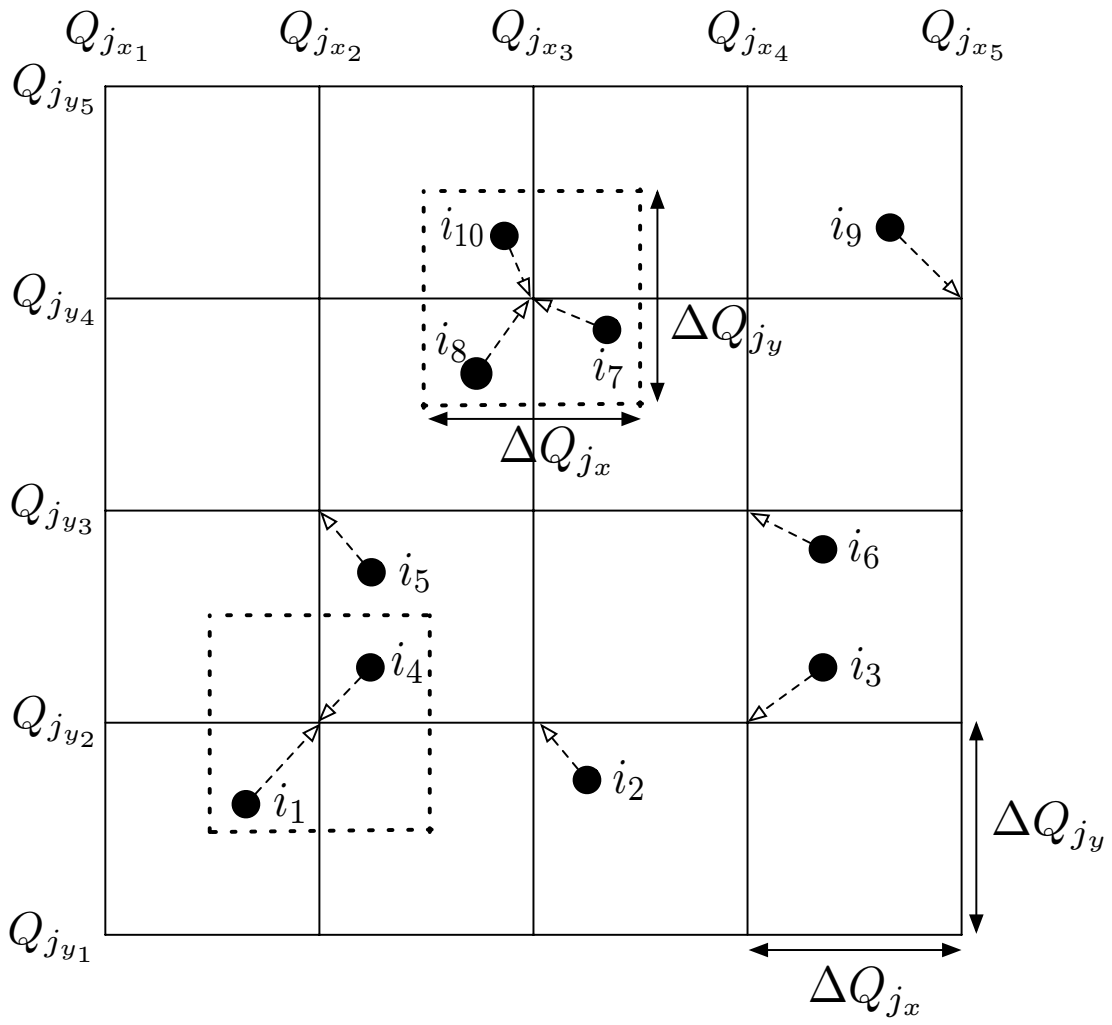

Figure 4.1: This sketch shows a 2-dimensional version of the coarse-graining operation that uses nearest grid-point interpolation. The dotted arrows show onto which grid-point a particle weight gets interpolated. The interpolation volume has the dimension $\Delta Q_{j_x} \times \Delta Q_{j_y}$. Particles i_1, i_4 as well as i_7, i_8, i_{10} are getting interpolated onto the same grid-point and as a consequence their particle weights will be set to their respective average values.

The coarse graining then simply sets the individual particle weights to the average value of those particles that are near the same gridpoint.

In Fig.(4.1) we give a 2-dimensional example for the coarse-graining algorithm that uses nearest-neighbor interpolation. The individual particles are labeled by the indices i_k. The particles i_1 and i_4 fall both into the interpolation volume around the grid-point $(Q_{j_{x_2}}, Q_{j_{y_2}})$. Therefore their particle weights of particle i_1 and i_4 will be set to $w'_{i_1} = w'_{i_4} = \frac{1}{2}(w_{i_1} + w_{i_4})$ after we apply the coarse-graining operator. The particles i_7, i_8 and i_{10} are all within the interpolation volume of grid-point $(Q_{j_{x_3}}, Q_{j_{y_4}})$. So their weights become $w'_{i_7} = w'_{i_8} = w'_{i_{10}} = \frac{1}{3}(w_{i_7} + w_{i_8} + w_{i_{10}})$. All other particle weights will stay unchanged in this example since they do not have one or more neighbor particles that fall inside the same interpolation volume of a grid-point.

The averaging of nearby particle weights introduces strong dissipation in the simulation. We visualize this effect in Fig.(4.2) where we show how we lose structure in the weight distribution by setting all particle weights that fall inside one interpolation volume to the same average value. This means that the equations describing the time evolution of the mean-squared particle weights [Eq.(3.7)] gets an additional term D that must capture the dissipation due to coarse graining:

$$\frac{d}{dt} < w^2 >= 2\left(\frac{1}{L_n} - \frac{3}{2L_T}\right)\Gamma_p + \frac{2}{L_T}\Gamma_e + \frac{2q}{NT_0}\sum_{i=1}^{N} w_i v_{\|_i} < E_{\|_i} >_{\mathbf{R}} - D. \qquad (4.4)$$

The trick is now to balance with the dissipative term D the energy flux term $2\Gamma_e/L_T$ that dominates the growth of the weights. In fact, this scheme introduces strong dissipation, so that Chen and Parker [14] apply this term just every N_s-timesteps (with $N_s = 10$ in a typical run for the Cyclone base case [23]) and suggest that one should run the code in the limit where there are not too many particles

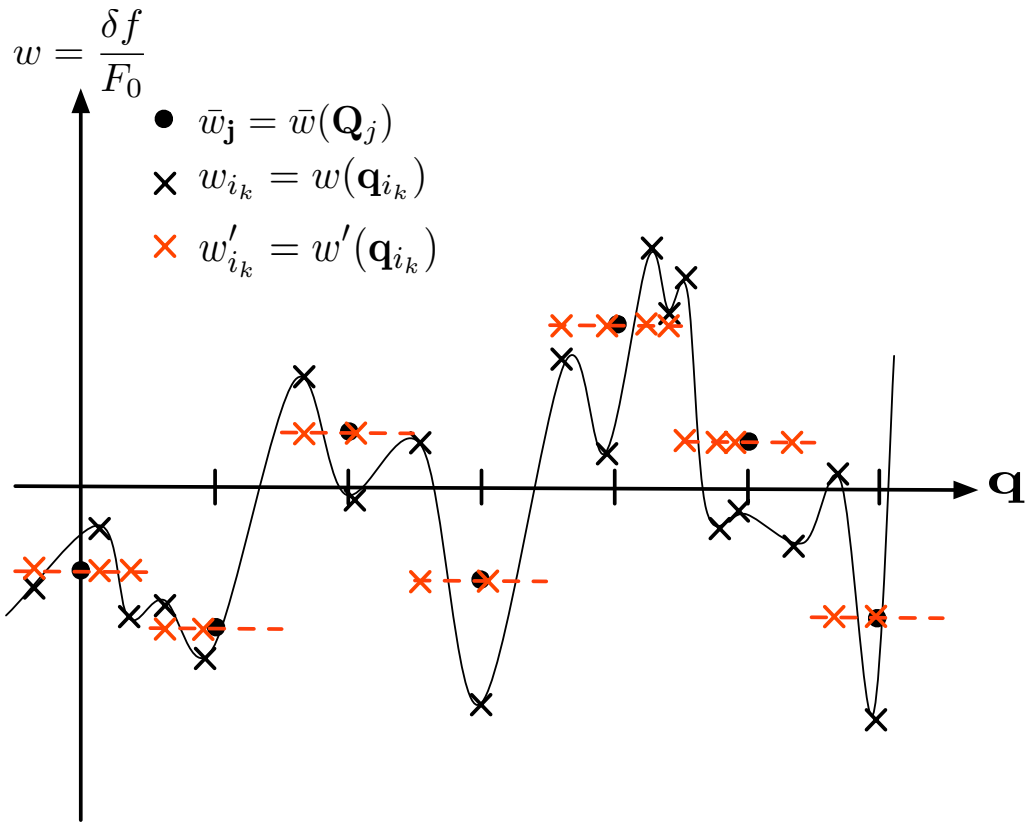

Figure 4.2: This sketch shows how the coarse-graining algorithm that uses the nearest grid-point interpolations changes the the individual particle weights. From this visualization it is recognizable that is method is strongly dissipative.

per 5D-grid cell, so that most particles do not get affected by the coarse graining operator. In addition to those two steps to reduce the dissipation they also introduce the following lag average formula for reassigning the weights:

$$w''_i = (1-\delta)\, w_i + \delta w'_i = (1-\delta)\, w_i + \delta \sum_{j=1}^{N_{5d}} g_{i,\mathbf{j}} \bar{w}_{\mathbf{j}}. \tag{4.5}$$

Chen and Parker recommend to set the parameter $\delta \ll 1$. This allows to apply the coarse-graining operation more often while not making it too dissipative

and therefore one can choose a smaller value for N_s. A large value for N_s together with a larger value for δ has the disadvantage that it introduces a larger discontinuity into the simulation.

4.1.2 Coarse-graining with bilinear interpolation

We can repeat the method that we introduced above but replace the interpolation algorithm from nearest neighbor interpolation to a bilinear interpolation scheme. The scheme is equivalent to the scheme that we introduced in Eq.(4.6), but is now generalized for five dimensions:

$$g^{BLi}_{i,\mathbf{j}} = \frac{G_{Q_{jx},i} G_{Q_{jy},i} G_{Q_{jz},i} G_{Q_{j_E},i} G_{Q_{j_\xi},i}}{\Delta Q_{j_x} \Delta Q_{j_x} \Delta Q_{j_x} \Delta Q_{j_E} \Delta Q_{j_\xi}}, \tag{4.6}$$

$$\text{with:} \quad G_{Q_{j_{(\cdot)},i}} = \begin{cases} Q_{j_{(\cdot)}+1,i} - q_{(\cdot),i} & : \quad Q_{j_{(\cdot)},i} < q_{(\cdot),i} < Q_{j_{(\cdot)}+1,i} \\ q_{(\cdot),i} - Q_{j_{(\cdot)},i} & : \quad Q_{j_{(\cdot)}-1,i} < q_{j(\cdot),i} < Q_{j_{(\cdot)},i} \\ 0 & : \quad \text{else.} \end{cases} \tag{4.7}$$

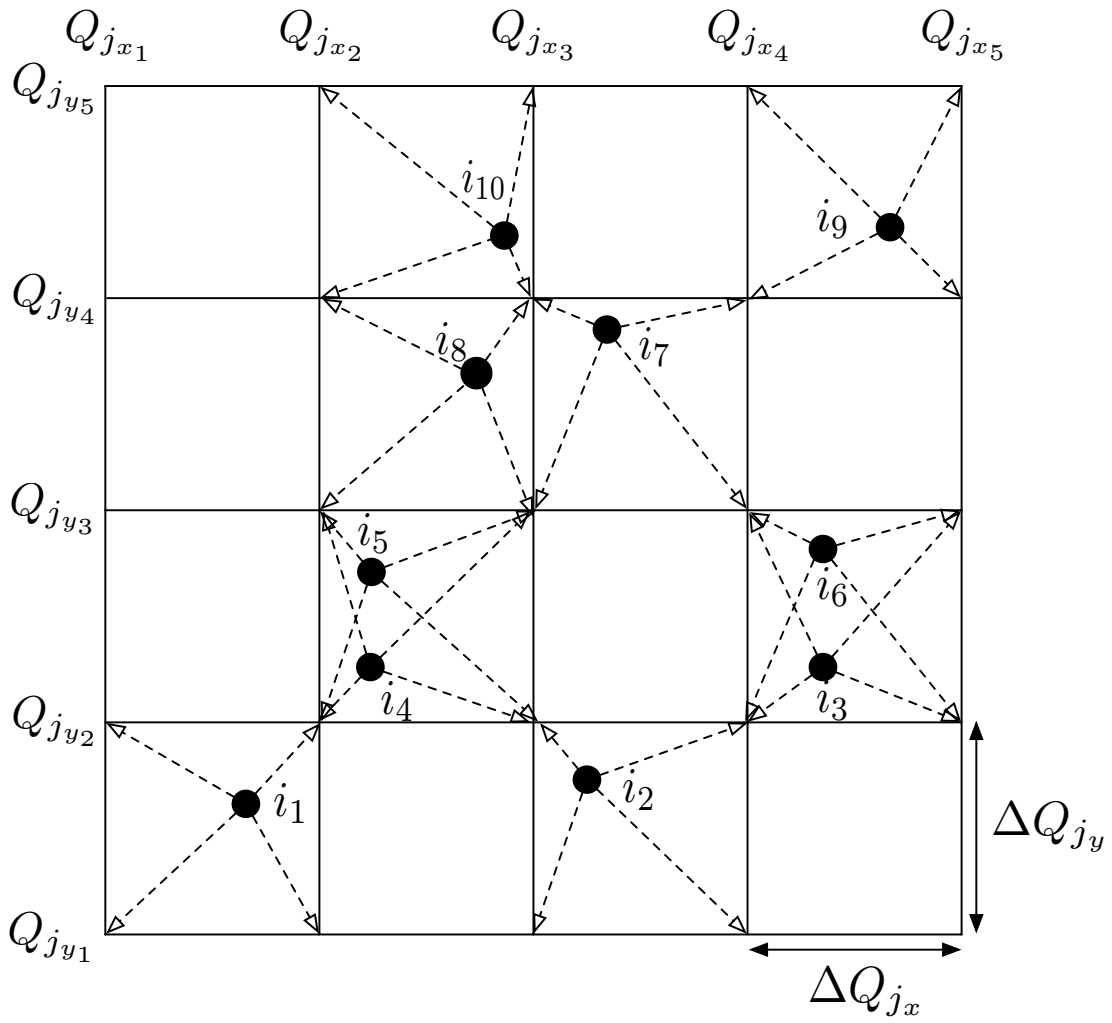

Figure 4.3: This sketch shows a 2-dimensional version of the coarse-graining operation that uses bilinear interpolation. The dotted arrows show onto which grid-points a particle weight gets interpolated. Though we are using the same number of particles in this case as in Fig(4.1) here all ten particle weights will be changed at this time step while for the nearest-neighbor technique just five out of the ten particle weights were affected by the coarse graining.

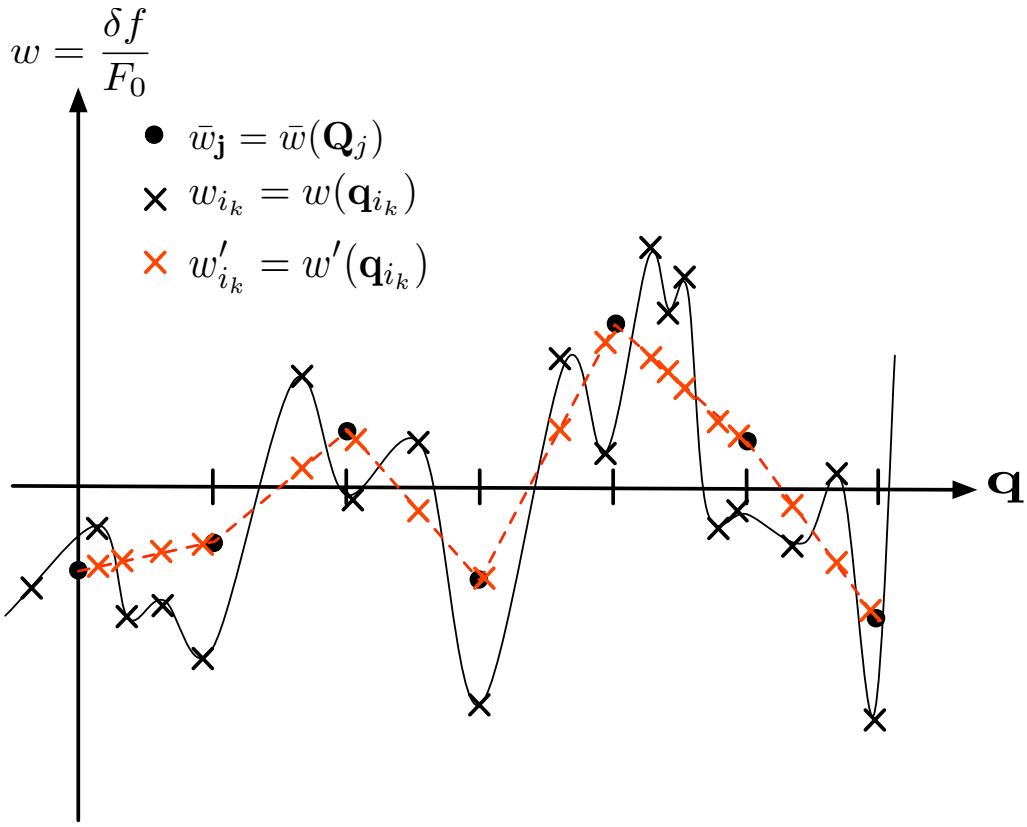

Figure 4.4: This sketch shows how the coarse-graining algorithm that uses the bilinear interpolation technique changes the the individual particle weights. From this visualization it is recognizable that is method is still strongly dissipative though it is less dissipative than the nearest-neighbor interpolation scheme sketched out in Fig(4.2).

Chapter 5

Collision operator

In most physical plasmas, dissipation is present. Its importance for reaching a true steady state in a non-equilibrium system has been discussed by Krommes [41, 42]. Collisions are the physical origin of dissipation in a plasma. Without them, the distribution function could develop infinitesimally small structures in velocity space [3], [58]. Besides being unphysical (since collisions are going to prevent small structures from developing), small structures in velocity space are also a numerical challenge. Therefore the presence of either artificial or physical dissipation in numerical simulations of plasmas is required. The secular growth of particle weights is a direct consequence of treating the plasma as dissipationless in a simulation. Phase-space filamentation is the accumulated result of this effect and leads to fine scales in the distribution function that will not be resolved with a finite number of particles in the code. Since the only physical process to stop the formation of fine scale structures are collisions, purely collisionless simulations of steady-state turbulent systems can be questioned.

Implementations of collisions in δf codes normally involves a Monte Carlo

scattering operator that randomly changes the the particle coordinates in the code [20]. Simulation results of PIC codes that include a Monte Carlo scattering operator have shown an increase in the growth of the mean-squared particle weights compared to standard δf PIC codes [9, 15]. This counterintuitive result, that collisions which are supposed to limit phase-space filamentations and therefore should limit the growth in the weights lead instead to a growth in the weight, is explained by Chen and White [16]. They find that while the δf distribution might be smoothed by a Monte-Carlo collision operator, the distribution of weights across small scales will be broadened in time. This led to additional efforts to try to solve the problem of growing weights in PIC codes. In the previous chapter we already described the coarse-graining method of Chen and Parker [14] that adds artificial dissipation to the simulation to deal with the problem of growing mean-squared weights in a δf particle code. Another approach is to add a Krook operator or a Krook-like term, as the thermostatted δf scheme by Krommes [41]. McMillan *et al.,* have been working on modified Krook operator for a global δf simulation [47]. Their Krook operator is modified to project out the unphysical effects a relaxation operator like the Krook operator has on zonal flows. In this chapter we are going to show that a Krook operator damps away flow profiles in the plasma fast and explain why this is an effect that is an artifact and not physical.

Eulerian codes do not face the challenge that particle codes face with respect to growing weights and how those weights will at late times affect the accuracy of the results. But also in continuum codes fine structures that form in velocity space lead to problems. Not just that those small scale structures may be unphysical, they

also post a challenge to accurately evaluate velocity space integrals in the code. If the small scale structures grow and fill in smaller and smaller scales, eventually the velocity space grid in an Eulerian code will fail to resolve the distribution functions and the results will be questionable. Therefore most current Eulerian gyrokinetic codes employ some form of numerical dissipation [3], [12]. As for particle codes there are several different collision operators in Eulerian codes, most prominently the Krook and the Lorentz pitch-angle scattering operators. Barnes *et al.*[4] most recently have been working on an improved model operator which includes pitch-angle scattering and energy diffusion and has a lot of the desired properties of an accurate collision operator as it conserves particle number, momentum and energy plus satisfies Boltzmann's H-theorem.

Our approach is to add a pitch-angle collision operator to the PIC code. This seems to be the first attempt for a δf PIC code to have an actual collisional operator that acts directly on the distribution function δf. The operator reassigns particle weights accordingly to the changes of δf based on the collisions. Particle positions and particle velocities will not be updated by the operator. By resetting the weights and leaving the phase-space coordinates of the particles fixed our method is similar to the coarse-graining by Chen and Parker [14]. The big difference is that our collision operator comes from of the derivation of gyrokinetics from the Fokker Plank equation while Chen and Parker add an artificial term that is designed to damp the weight growth but is not tied to the underlying equations describing the physics.

5.1 Krook operator

Before discussing in detail about the pitch-angle scattering operator in detail, we briefly sum up some of the main features of a relaxation operator like the Krook operator and show how it is implemented in the code. As we will show later in this chapter the Krook operator can efficiently suppress the growth of the mean squared weight in a PIC simulation but imposes problems to zonal flows as they are strongly damped.

The Krook operator has the following form:

$$C_K(\delta f) \equiv -\nu \delta f. \tag{5.1}$$

The implementation of such an operator to a particle code is rather trivial. The Krook operator resets the values of all particle weights by an amount that is identical in relative terms to the individual particle weight. An explicit scheme leads to the following change in the perturbed distribution function:

$$\begin{aligned} \left(\frac{\partial < \delta f >_{\mathbf{R}}}{\partial t}\right)_C &\approx \frac{< \delta f^{n+1} >_{\mathbf{R}} - < \delta f^* >_{\mathbf{R}}}{\Delta t} = -\nu < \delta f^* >_{\mathbf{R}} \\ \Rightarrow \qquad & < \delta f^{n+1} >_{\mathbf{R}} = (1 - \Delta t \nu) < \delta f^* >_{\mathbf{R}} \end{aligned} \tag{5.2}$$

Since the particle weights are defined as $w_i = \delta f / F_0|_{\mathbf{x_i},\mathbf{v_i}}$, we can directly reassign the particle weights in the same way the distribution function has been updated in Eq.(5.2):

$$w_i^{n+1} = (1 - \Delta t \nu) w_i^*. \tag{5.3}$$

From Eq.(5.3) it is obvious that the Krook operator leads to an overall relaxation in the particle weights. This relaxation is totally independent of the structure

the perturbed distribution function develops in phase space. A distribution function that is entirely flat and has no spatial or velocity dependence will be damped in the same way as one that is highly oscilliatory. The shape won't be affected just the overall values of the weights are getting reduced. So the amplitude of an oscillatory distribution will be reduced, but not its frequency.

5.2 Pitch-angle collision operator

Landau [43] calculated the effect of small angle Coulomb collisions on a distribution function. The complexity of the Landau operator makes it difficult to use for numerical purposes. As a consequence, simpler model operators have been developed over the years [56, 32, 13].

As before, we are looking at the electrostatic version of the gyrokinetic equation in a slab geometry, described in Chapter 2. But now we do not treat the problem as collisionless any longer. By adding a collision operator, Eq.(2.1) becomes:

$$\frac{\partial}{\partial t}\langle \delta f\rangle_{\mathbf{R}} + v_{\parallel}\mathbf{b}_0\cdot\nabla\langle \delta f\rangle_{\mathbf{R}} + \langle \mathbf{v}_{E\times B}\rangle_{\mathbf{R}}\cdot\nabla\langle \delta f\rangle_{\mathbf{R}} =$$
$$-\langle \mathbf{v}_{E\times B}\rangle_{\mathbf{R}}\cdot\nabla F_0 - v_{\parallel}\frac{q}{T}F_0\ \mathbf{b}_0\cdot\nabla\langle \phi\rangle_{\mathbf{R}} + \langle C_M(\delta f)\rangle_{\mathbf{R}}\,. \tag{5.4}$$

For the collision operator $C_M(\delta f)$ we use in our PIC-code the pitch-angle scattering operator for ion-ion collisions. This operator has the advantage that it can conserve particle number, momentum, and energy. It is constructed by taking the test-particle pitch angle scattering collision operator and correcting it with an additional term that ensures momentum conversation. In this form it was first introduced by Rosenbluth et al. [55], [30].

The pitch-angle scattering operator acts on $h(x, y, z, \xi, E)$, the part of the perturbed distribution function that is the homogenous solution to Eq.(1.12) and was introduced in Chapter 1.3.3. The operator has the following form:

$$C_M(h_s) = \frac{\nu_D^{ss}}{2}\left(\frac{\partial}{\partial\xi}(1-\xi^2)\frac{\partial h_s}{\partial\xi} + \frac{1}{1-\xi^2}\frac{\partial^2 h_s}{\partial\zeta} + \frac{\mathbf{v}\cdot\mathbf{U}[\mathbf{h_s}]}{v_{T_s}}\right), \tag{5.5}$$

with

$$\mathbf{U}[h_s] = \frac{3}{2}\frac{\int d^3\mathbf{v}\ \mathbf{v}\ \nu_D^{ss} h_s}{\int d^3\mathbf{v}(v/v_{T_s})^2\ \nu_D^{ss}\ F_{0_s}(v)}, \tag{5.6}$$

$$\nu_D^{ss} = \nu_{ss}\left(\frac{\sqrt{2}\ v_{T_s}}{v}\right)^3\left(\left(1-\frac{v_{T_s}^2}{v^2}\right)\mathrm{erf}\left(\frac{v}{\sqrt{2}\ v_{T_s}}\right) + \frac{v_{T_s}}{\sqrt{2}\ v}\mathrm{erf}\,'\left(\frac{v}{\sqrt{2}\ v_{T_s}}\right)\right). \tag{5.7}$$

All derivatives are taken at constant particle position $\mathbf{r}$. The gyrokinetic version of this operator was introduced by Catto & Tsang [13], [20] by applying the gyro-average operator to Eq.(5.5).

$$\begin{aligned} < C_M[h_s] >_{\mathbf{R_s}} &= \sum_{\mathbf{k}} e^{i\mathbf{k}\cdot\mathbf{R}}\nu_D^{ss}\left\{\frac{1}{2}\frac{\partial}{\partial\xi}(1-\xi^2)\frac{\partial h_{s\mathbf{k}}}{\partial\xi} - \frac{v^2(1+\xi^2)}{8v_{t_s}^2}k_\perp^2\rho_s^2 h_{s\mathbf{k}}\right\} + \\ &+ \sum_{\mathbf{k}} e^{i\mathbf{k}\cdot\mathbf{R}}\nu_D^{ss}\left\{\frac{v_\perp J_1(a_s)U_\perp[h_{s\mathbf{k}}] + v_\parallel J_0(a_s)U_\parallel[h_{s\mathbf{k}}}{2v_{T_s}}F_{0s}\right\}. \end{aligned} \tag{5.8}$$

with $a_s = k_\perp v_\perp/\Omega_s$, and:

$$U_\perp[h_{s\mathbf{k}}] = \frac{3}{2}\frac{\int d^3\mathbf{v} v_\parallel J_0(a_s)\nu_D^{ss}h_{s\mathbf{k}}(v_\perp, v_\parallel)}{\int d^3(v^2/2v_{t_s})\nu_D^{ss}F_{0s}(v)},\ U_\parallel[h_{s\mathbf{k}}] = \frac{3}{2}\frac{\int d^3\mathbf{v} v_\perp J_1(a_s)\nu_D^{ss}h_{s\mathbf{k}}(v_\perp, v_\parallel)}{\int d^3(v^2/2v_{t_s})\nu_D^{ss}F_{0s}(v)}.$$

The derivatives in velocity space are now taken at constant gyrocenter position $\mathbf{R}_s$. The second term in the first bracket of Eq.(5.8) is diffusive. This diffusion is physical and due to the fact that the ring-averaged collision operator leads to changes in the velocity of a particle which can lead to spatial repositioning of the particle's gyrocenter.

The lowest order form of the pitch-angle collision operator Eq.(5.8) in the limit of $k_\perp \rho_s \ll 1$ is:

$$< C_M[h_s] >_{\mathbf{R}_s} = \tag{5.9}$$
$$\nu_D^{ss}(v) \quad \left\{ \frac{1}{2} \frac{\partial}{\partial \xi} (1-\xi^2) \frac{\partial h_{s\mathbf{k}}}{\partial \xi} + \frac{3 v_\parallel \int d^3\mathbf{v}' v'_\parallel \nu_D^{ss}(v') h_s(v'_\perp, v'_\parallel)}{\int d^3\mathbf{v}' v'^2 \nu_D^{ss}(v') F_{0_s}(v')} F_{0_s} \right\} + O(k_\perp \rho_s^2).$$

Our goal is to implement the collision operator that is given in Eq.(5.8) into our particle code. In the next section we are going to explain how the numerics of our PIC code need to be changed from the collisionless case to do that.

5.3 Numerical implementation of pitch-angle scattering in GSP

When solving Eq.(5.4) with the pitch-angle-scattering operator [Eq.(5.8)] for the term $< C_M(\delta f) >$, we use an implicit scheme to evaluate the collision operator. We are now going to describe the numerical scheme for the version of the code that contains the pitch-angle collision operator. We also give an pictorial overview of the numerical scheme of the collisional code in Fig.(5.1).

As a first step we rewrite Eq.(5.4) in terms of h and introduce operators $\mathcal{A}$ and $\mathcal{C}$. $\mathcal{A}$ captures the rate of change of h for the collisionless part of the gyrokinetic equation and $\mathcal{C}$ represents the rate of change of h due to collisions.

$$\text{Eq.(5.4)} \Leftrightarrow \frac{\partial}{\partial t} h = \mathcal{A}(h) + < \mathcal{C}(h) >_{\mathbf{R}} \tag{5.10}$$
$$= \left(\frac{\partial h}{\partial t} \right)_{\mathcal{A}} + \left(\frac{\partial h}{\partial t} \right)_{\mathcal{C}} \tag{5.11}$$

We now to solve this system and to separate the terms in the code we apply Godunov dimensional splitting [27], which is a scheme that is first order accurate in time:

$$\left(\frac{\partial h}{\partial t}\right)_{\mathcal{A}} \approx \frac{h^* - h^n}{\Delta t} = \mathcal{A}(h^n, h^{n+1}), \tag{5.12}$$

$$\left(\frac{\partial h}{\partial t}\right)_{\mathcal{C}} \approx \frac{h^{n+1} - h^*}{\Delta t} =< \mathcal{C}(h^n, h^{n+1}) >_{\mathbf{R}} . \tag{5.13}$$

Hence, as a first step we find the solution for Eq.(5.12). This is the collisional problem that we described before. So the code is doing the exact same as before in order to find h^*. The code finds the new particle coordinates in phase-space by having the particles follow their trajectories and updates their weights according to Eq.(2.21). Given $q_i(t^*)$ and $w_i(t^*)$ the code determines the perturbed distribution function δf^*.

To evaluate the $\left(\frac{\partial h}{\partial t}\right)_{\mathcal{C}}$ operator, we find h, the non-Boltzmann part of the lowest order perturbed distribution function from the relation: $h =< \delta f >_{\mathbf{R}} + \frac{q<\phi>_{\mathbf{R}}}{T} F_0$. To be able to apply the pitch-angle collision operator we need to find the values of the perturbed distribution function on a grid in phase-space so that we can evaluate the derivatives in Eq.(5.8). We use the same energy-pitch-angle-grid that we describe in chapter 3.2. The complication of having to find the distribution function h and therefore δf respectively can be avoided for the case that we do choose the Krook operator Eq.(5.1) instead of the pitch-angle scattering operator. This means that the Krook operator is numerically less expensive than the pitch-angle scattering operator. The difference it takes the code to perform one time-step is quite large when we compare the version of the code that uses the Krook operator

with the version that uses pitch-angle scattering. When the code runs with the Krook operator instead of the pitch-angle collision operator the time per time-step is reduced by a factor of ~ 3. So the code does exactly the same it did in the

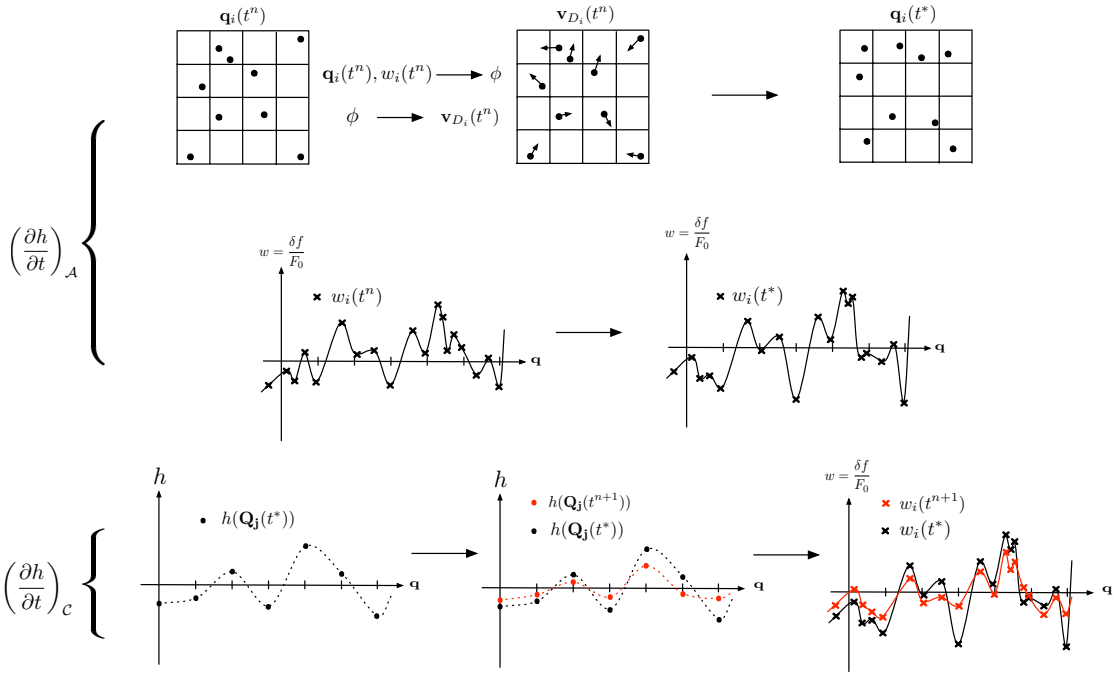

Figure 5.1: Flow-chart that explains the code's numerical scheme that includes pitch-angle collisions. First we solve the operator $\left(\frac{\partial h}{\partial t}\right)_{\mathcal{A}}$ that excludes collisions to find the particles coordinates $\mathbf{q}_i(t^*)$ and particle weights $w_i(t^*)$ at time t^*. The collision operator $\left(\frac{\partial h}{\partial t}\right)_{\mathcal{C}}$ then determines the particle weights $w_i(t^{n+1})$ but leaves the particle spatial and velocity coordinates unchanged ($q_i(t^*) = q_i(t^{n+1})$).

collisionless case to find δf^* given δf^n, the solution at the previous time step n. As in the version of the code with coarse graining, we need to interpolate the perturbed distribution function on the 5D-grid (x, y, z, ξ, E). To be able to solve for $C(h)$ we

need to find the function h given δf. Actually we are not going to use h but (h/F_0) for the collision operator. With the normalization used in the code this means that we just need to add the gyro-averaged potential $\phi(x, y, z)$ to $(\delta f(x, y, z, \xi, E)/F_0)$ to get h/F_0. In the next step we use an explicit scheme to find h at the new time-step $n + 1$ and transform this back to δf by subtracting the gyro-averaged potential ϕ. The following list sketches out the numerical scheme for the code that is solving a system that is described by:

$$\text{Eq.(5.4)} \Leftrightarrow \frac{\partial}{\partial t}\delta f = A(\delta f, F_0) + C(\delta f) \tag{5.14}$$

1. $\frac{\partial}{\partial t}\delta f = A(\delta f, F_0) \Rightarrow \frac{\delta f^* - \delta f^n}{\Delta t} = A(\delta f^n, F_0)$

2. Find δf on a 5D-grid; see chapter 3, Fig.(3.2) and chapter 4 where we use the same 5D version of δf for the coarse-graining operator

3. Determine $h(x, y, x, \xi, E) : \frac{\delta f^*}{F_0} + < \phi_N >= \frac{h^*}{F_0}$

4. Implicitly solve the collision operator: $\frac{\partial}{\partial t}h^* = C(h^{n+1})$

   in matrix notation form this becomes: $\frac{h^{n+1} - h^*}{\Delta t} = \mathbf{C}(h^{n+1})$

5. The code inverts a tri-diagonal matrix to find h^{n+1}:

$$h^{n+1} = (\mathbf{I} - \Delta t\mathbf{C})^{-1}h^*$$

6. Find the distribution function δf at new time step t^{n+1}

$$\frac{\delta f^{n+1}}{F_0} = \frac{h^{n+1}}{F_0} - < \phi^n >$$

7. Re-interpolate δf to the particle positions and change the particle weights accordingly; see Eq. (5.19)-(5.21)

Step 1. to 3. are either trivial or have been explained in the collisionless limit of the code. Step 4. needs further explanation. In order to evaluate the collision operator Eq.(5.8) we need to determine the collision frequency $\nu_D(v)$,

$$\nu_D(v) = \frac{\nu\ 2^{\frac{3}{2}}\left(\mathrm{erf}\left(\frac{v_N}{\sqrt{2}}\right)\left(1-v_N^{-2}\right)+e^{-v_N^2/2}\sqrt{\frac{2}{\pi v_N^2}}\right)}{v_N^3}, \tag{5.15}$$

with $v_N = \frac{v}{v_T}$.

Using energy E and pitch-angle ξ as coordinates for the velocity part of phase-space, the collision frequency $\nu_D(v)$ is a constant factor for a given energy band that consists of a fixed value for v. Because of the denominator v_N^3, the collision frequency falls of rapidly for higher energy values. When we are studying (later in this chapter) the effect that pitch-angle scattering has on the perturbed distribution function δf, we will see that the smoothing gets weak for large energy values. This is a direct consequence of the velocity dependence of ν_D . The parameter ν sets the level of collisionality in the plasma. In the code, for a fixed value of the energy E_j, we have $(2\times j-1)*$ number $\xi = N_j$, grid points. So the pitch-angle scattering operator is solved in the code in the following way:

$$\begin{aligned} C(h(E_j,\xi)) &= \frac{\nu_D(v_j)}{2}\frac{\partial}{\partial\xi}(1-\xi^2)\frac{\partial}{\partial\xi}h \quad \Rightarrow i : i^{th}\xi \text{ - grid-point} \\ &= \frac{\nu_D(v_j)}{2}\frac{\left(1-\xi_{i+\frac{1}{2}}^2\right)\frac{h_{i+1}-h_i}{\xi_{i+1}-\xi_i}-\left(1-\xi_{i-\frac{1}{2}}^2\right)\frac{h_i-h_{i-1}}{\xi_i-\xi_{i-1}}}{\xi_{i+\frac{1}{2}}-\xi_{i-\frac{1}{2}}} \\ &= \frac{\nu_D}{\xi_{i+1}-\xi_{i-1}}\left(h_{i+1}\underbrace{\frac{\left(1-\xi_{i+\frac{1}{2}}^2\right)}{\xi_{i+1}-\xi_i}}_{=\mathbf{c}}+h_{i-1}\underbrace{\frac{\left(1-\xi_{i-\frac{1}{2}}^2\right)}{\xi_i-\xi_{i-1}}}_{=\mathbf{a}}-h_i(\mathbf{c}+\mathbf{a})\right). \end{aligned}$$

Thus we can write the operator $C(h)$ in matrix form:

$$C(h) = \begin{pmatrix} b_1 & c_1 & 0 & \dots & \dots & \dots & \dots & 0 \\ a_2 & b_2 & c_2 & 0 & \dots & \dots & \dots & 0 \\ 0 & a_3 & b_3 & c_3 & 0 & \dots & \dots & 0 \\ 0 & 0 & a_4 & b_4 & c_4 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & \dots & \dots & 0 & a_{N-1} & b_{N-1} & c_{N-1} \\ 0 & \dots & \dots & \dots & \dots & 0 & a_{N-1} & b_N \end{pmatrix} \begin{pmatrix} h_1 \\ h_2 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ h_N \end{pmatrix}, \tag{5.16}$$

$$\text{with} \quad a_i = \frac{\nu_D}{\xi_{i+1} - \xi_{i-1}} \frac{\left(1 - \xi^2_{i-\frac{1}{2}}\right)}{\xi_i - \xi_{i-1}},\ b_i = -(a_i + c_i),\ c_i = \frac{\nu_D}{\xi_{i+1} - \xi_{i-1}} \frac{\left(1 - \xi^2_{i+\frac{1}{2}}\right)}{\xi_{i+1} - \xi_{i-1}}. \tag{5.17}$$

Since we are solving the pitch-angle-scattering collision term implicitly we are dealing with the following equation.

$$\frac{h^{n+1} - h^*}{\Delta t} = \mathbf{C}(h^{n+1}) \Rightarrow h^{n+1} = (\mathbf{I} - \Delta t \mathbf{C})^{-1} h^*$$

To determine h^{n+1} we invert the tri-diagonal matrix $\mathbf{I} - \Delta t \mathbf{C}$ using the standard techniques.

After evaluating the collision operator we find in step 5 the value of $\frac{\delta f}{F_0}$ on the 5D-grid (x, y, z, ξ, E). This gives us the average value of the particle weights on the grid at the new time t^{n+1}.

$$\bar{w}_j^{t+1} = \left(\frac{\delta f^{t+1}}{F_0}\right)_{\mathbf{x}_j^{t+1}, E_j^{t+1}, \xi_j^{t+1}} = \frac{\sum_{i=1}^N g_{i,j}^{t+1} w_i^{t+1}}{\sum_{i=1}^N g_{i,j}^{t+1}}, \tag{5.18}$$

with $g_{i,j}$ the interpolation weight of particle i onto the grid-point $-(\mathbf{x}_j, E_j, \xi_j)$.

We need to re-interpolate these average weights to the particle positions and change the current particle weights. To do this we have a choice to make. We can as we did in coarse-graining introduce an additional dissipative term. This dissipative term has the equivalent origin as it had in the coarse-graining method and is based on adjusting an individual particle weight by a certain fraction toward the average weight on its corresponding grid-points. By corresponding grid-points we mean those eight grid-points which have an interpolation weight on a specific particle that is non-zero. Instead of adding a dissipative term that comes from the interpolation we choose to update the value of the weight of an individual particle relative to the change of the average weight on the grid-points which are due to the pitch-angle collisions. We want to make sure the reassigned particle weights are consistent with the results from the collision operator. Consequently, the reassigned weights w_i^{t+1} should give us $\bar{w}_{\mathbf{j}}^{n+1}$ back if we would interpolate them again onto the grid. By interpolating back and forth between grid-points and particle positions we don't want to introduce any changes to the weights. The following reassignment algorithm has those desired properties:

$$w_i^{n+1} = w_i^* + \sum_{\mathbf{j}} w_{i,j}^{n+1}, \tag{5.19}$$

$$\text{with} \qquad w_{i,j}^{n+1} = g_{i,\mathbf{j}}^* \left(\bar{w}_{\mathbf{j}}^{n+1} - \bar{w}_{\mathbf{j}}^* \right). \tag{5.20}$$

We can modify this weights reassignment to include a lag average as it is defined for the coarse-graining operator. This is done by assigning weight $w_{i,j}^{n+1}$ as:

$$w_{i,j}^{n+1} = g_{i,j}^* \left((1-\gamma)(\bar{w}_{\mathbf{j}}^{n+1} - \bar{w}_{\mathbf{j}}^*) + \gamma \bar{w}^* \right). \tag{5.21}$$

For $\gamma = 0.0$ we get Eq.(5.20) back. The other extreme case is that we choose $\gamma = 1.0$. In that case a particle is assigned the weighted averages $\bar{w}^*_{\mathbf{j}}$ from the eight neighboring grid-points of a particle. This is exactly the coarse-graining operation without the lag averaging term [the case with $\delta = 1$ in Eq.(4.5)] and is therefore completely independent of the effects of pitch-angle collisions. For all results that we are presenting in this thesis we use $\gamma = 0$ in order to make the reassignment procedure of the weights consistent with the results we get from the pitch-angle scattering operator.

Note that unlike in the case for coarse graining, we use the collision operator at each time step and choose not to apply the operator just after a chosen time interval ΔT.

5.4 Testing the pitch-angle-scattering collision operator

Here we show several test cases for the pitch-angle-scattering collision operator. First we test the numerical implementation in the code for a set of test functions. Then we initialize a set of predefined profiles for the perturbed distribution function and check the effects that pitch-angle scattering has on those profiles in comparison to the effects of the Krook operator and to the coarse-graining operation.

5.4.1 Test of the numerical implementation on a known test function

To test the implementation of the pit-angle-scattering operator we set the distribution function $h^*(x, y, z, \xi, E)$ equal to a function for which we can easily determine analytically the form of h^{n+1}. The two functions h_1^* and h_2^* that we tested are:

$$h_1^*(x, y, z, \xi, E) = \xi(1 - \Delta t \nu_D(v)) \tag{5.22}$$

$$\Rightarrow h_1^{n+1} = \xi$$

$$h_2^*(x, y, z, \xi, E) = \cos(\xi) - \Delta t \nu_D(v) \left(2\xi \sin(\xi) - (1 - \xi^2)\cos(\xi)\right) \tag{5.23}$$

$$\Rightarrow h_2^{n+1} = \cos(\xi).$$

The numerical results from the collision operator are compared to the analytic solution for h_1 and h_2 in Fig.(5.2).

When we include more grid points the result become even better and we are able to reduce the error further. The error is most dominant for low values of the energy E, for which we have the fewest grid-points in ξ. [See Fig.(3.2), which explains the velocity grid.]

5.4.2 Test of the pitch-angle operator on sharply peaked profile in velocity space

In this section we study the effect the pitch-angle collision operator has on very sharply peaked profile in velocity space. To generate such a profile we set the

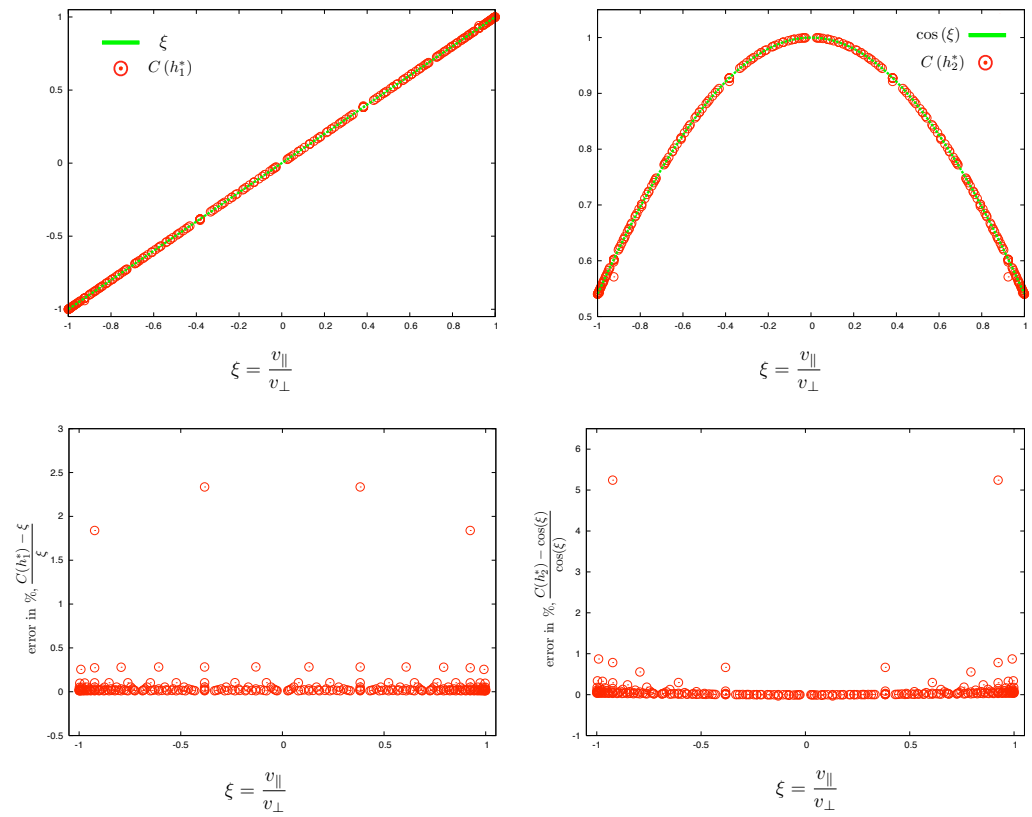

Figure 5.2: a) $C(h_1^*)$ compared to the analytical solution ξ, b) $C(h_2^*)$ compared to the analytical solution $\cos(\xi)$, c) the relative error of $C(h_1^*)$ compared to the analytic solution ξ. Note that the error is largest for the lowest energy-band for which we just have four points in this example, d) the relative error of $C(h_2^*)$ compared to the analytic solution $\cos(\xi)$.

values of the particle weights in a predefined form and run the code without any spatial dependence for the temperature and density profile, no curvature and for a slab geometry. Such a run is linearly stable and the velocity profile of δf is nearly

constant over time when $\nu = 0$. To test the effect of the pitch angle operator on such a sharply peaked profile we run the code in with those linearly stable parameters and initialize the particle weights constant throughout the entire domain but for one cell in velocity space (E, ξ), where we make the weights significantly larger. For the case that we are illustrating in Fig.(5.3) we choose all weights to be equal to unity. The only exception is that all weights that fall into the interpolation volume that is specified by the energy index $j_E = 4$ and the pitch-angle index $j_\xi = 1$ are initially set to have a value of 200. We see in Fig. (5.3) that the pitch-angle collision operator leads to a flattening of the gradient in the velocity profile by eventually distributing the weights uniformly within the energy band ($j_E = 4$) in which we had initialized the peak of the weight profile. The weights for particles with energy values that lie above or below the energy value of the peak will be unaffected by the pitch-angle collision operator since the pitch-angle collision operator does not include any energy diffusion. When we look at Fig.(5.3) we notice that the value for the overall largest weights in the simulation was reduced by two orders of magnitude from the start to the finish of the simulation at which point the pitch-angle dependence of the weights distribution has almost entirely disappeared due to collisions. In addition, we have to point out that the particle momentum is not going to be conserved by the pitch-angle collision operator. To ensure momentum conservation we need to code the additional terms that we introduced earlier, in Eq.(5.6). Currently those terms have not been added to the code but will be in the future to overcome the unphysical result of violating moment conservation.

The two other operators that we introduced in order to deal with the problem

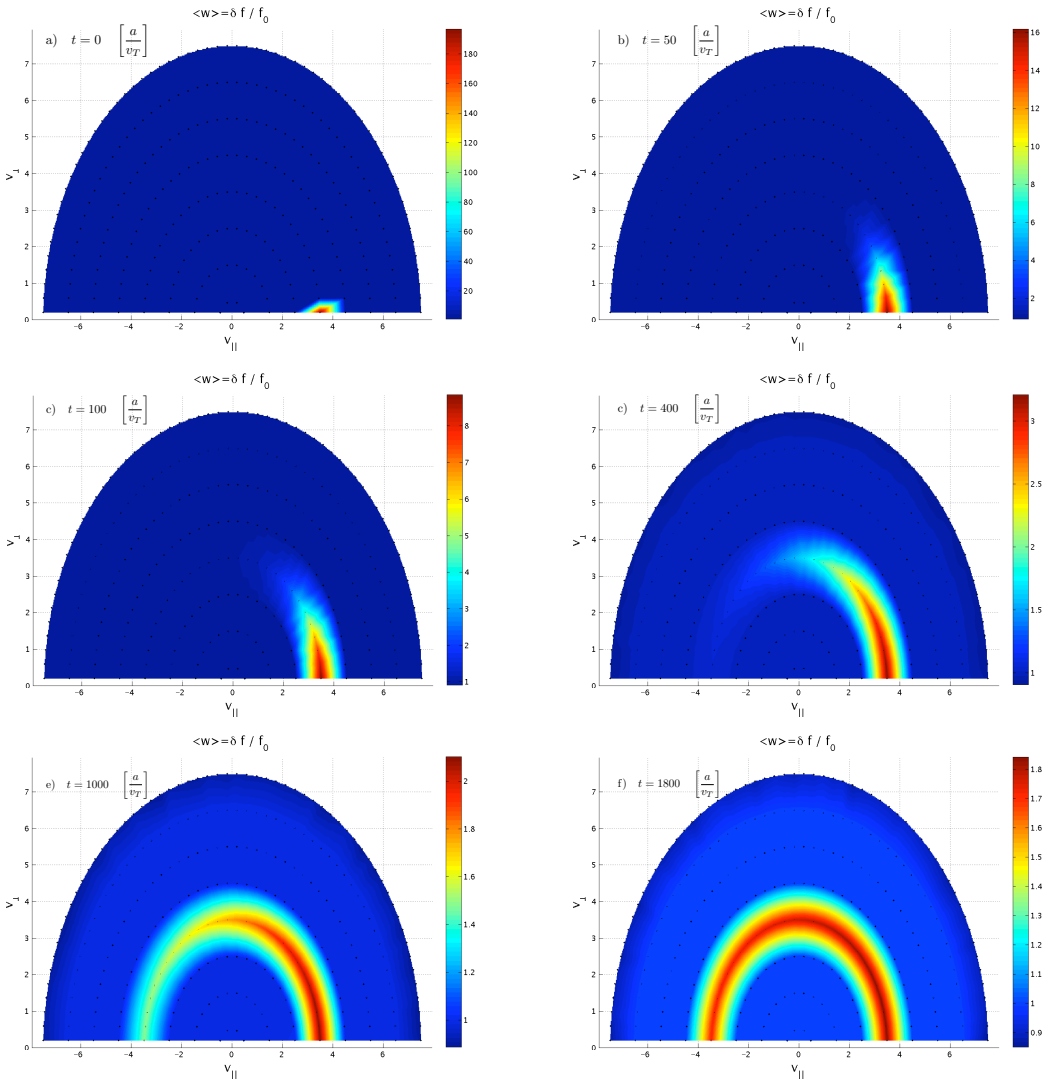

Figure 5.3: Time evaluation of a perturbed distribution function with a sharply peaked velocity profile. The pitch angle collision operator flattens the sharp gradient in the velocity profile of average particle weight $< w >= \frac{\delta f}{F_0}$ and leads to a flat profile within the energy band of the original peak.

of growing weights in gyrokinetic particle code simulations have a different effect on this form of the perturbed distribution. Both the Krook operator as well as coarse-graining will not smooth the velocity dependence of δf. The Krook operator just reduces the overall value of the weights, but since all weights are reduced by an amount that is equal in relative terms, the structure in velocity space will be preserved and just the absolute values of the weights are reduced. This means that the Krook operator deals with the growing weights by artificially reducing the overall values but it fails to identify filamentation issues that occur in velocity space. The coarse-graining operation has the same problem as the Krook operator as it fails to smoothen the sharp gradient in the form of the perturbed distribution function. This is due to the fact that coarse-graining averages the particle weights that fall within one interpolation cell. In our case we initialize the particle weights to be identical within each cell. Furthermore we don't have any spatial dependence of the weight distribution. We are only putting a velocity dependence in δf. Since the particle velocities are not updated along the characteristics and the we do not have a spatial dependence in the problem, coarse graining will neither affect the overall shape of this specific perturbed distribution function nor will it reduce the overall weights.

5.4.3 Collision Operator acting on flow profile

As in the previous section we are going to test the effects of coarse graining, the Krook operator and pitch-angle collisions on a predefined profile of δf. As we

explained before the Krook operator counteracts the growth of the particle weights and reduces therefore the computational noise that is accumulated by the growing weights. But by doing so it fails to distinguish between small scale structures and large scale structures and instead damps all scales at the same rate. Therefore it also damps out wave modes that we wish to simulate with our code. In this section we design a test case in order to study the effects from the three different operators on long lived zonal flow profiles. While it is obvious that the Krook operator will damp those flows at the same rate as it damps all structures in the system a similar claim for the coarse graining and pitch-angle scattering operator is not apparent. Indeed it is desirable property for a good collision model that large scale structures such as zonal flows do not get damped away heavily by the collisions and instead keep their character as a long lived structure.

We initialize the particle weights with a sinusoidal dependence in the $\hat{\mathbf{x}}$-direction. We choose the wavelength for the sinusoidal weight profile to be $L_x/2$. The perturbed distribution function contains the same spatial dependence as the weights and we find that the ϕ-spectrum is dominated by the wavelength that we selected for the initialization of the weights. Such a profile in ϕ with a strong spatial dependence in the perpendicular direction to the magnetic field leads to a strong $E \times B$-drift. Hence we find a dominant flow in our simulation. The parameters that define the strength of the three different collision operator are set to the values that we find in §5.5 to be strong enough to lead to a constant value over time in the nonlinear phase of an ITG-driven turbulence simulation for the sum of the squared particle weights. So the parameters work well in an example in which our goal is to

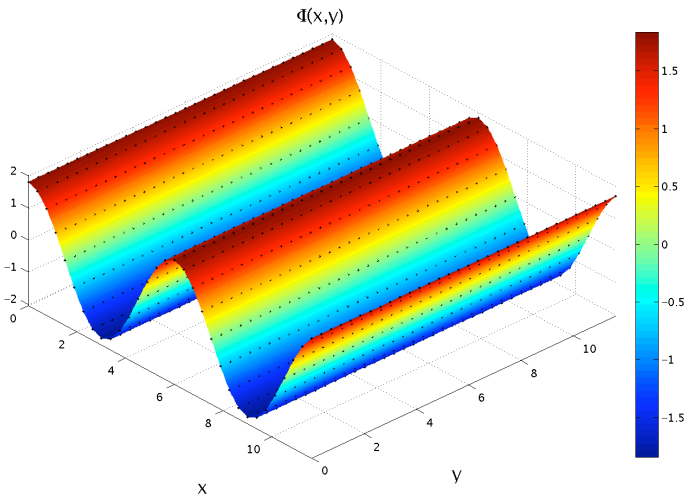

Figure 5.4: Flow-profile initialized to model the effect of different types of collision operators on zonal flows

suppress the weight growth, but at the same time not damp out heavily flow profiles that develop during the simulation.

For the result we are showing Fig. (5.5) we are using the parameters that we identify from the Fig. (5.7) - (5.9) as sufficient to stop the weight growth. Those parameters are summarized in Table 5.1.

From Fig. (5.5) we can conclude that coarse-graining and pitch-angle scattering have for the example that we are looking at the desired property of weakly damping a flow profile while the Krook operator damps the large scale profile in ϕ strongly. These results give a hint that it will be much harder for strong zonal flows to develop when we are including the Krook operator in the simulation compared to the coarse-graining or pitch-angle scattering operator.

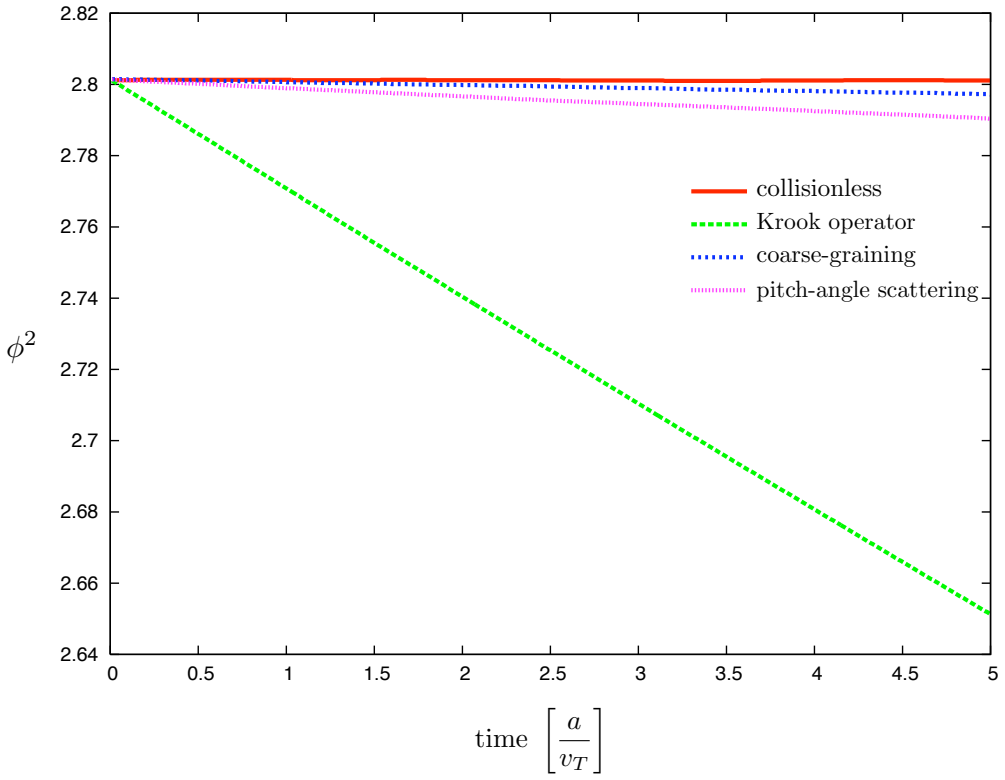

Figure 5.5: Flow-profile initialized to model the effect of different types of collision operators on zonal flows

5.5 ITG base case with different collision operators

While studying plasma turbulence with nonlinear particle codes growing weights pose a problem. As mentioned before the accuracy of the results from particle simulations decreases as the mean-squared particle weights increases. In this section we show the results of electrostatic simulations of Ion-Temperature-Gradient (ITG)

turbulence with adiabatic electrons using our PIC-code GSP . Ion-temperature-gradient-driven turbulence has been studied theoretically [17] and numerically [25] for a long time as it can lead to highly elongated streamers in the plasma which have a strong effect on the transport behavior in the plasma.

ITG driven turbulence is a good candidate for testing the collision operators since it is a well studied and documented problem [23], [50], [52]. We benchmark our results from GSP that we are presenting here with simulation results from the Eulerian code ASTROGK .

We set up an linear instability driven by a temperature gradient. For this run we use a temperature gradient scale length $L_T = 0.1$ and no gradient in the background density ($L_N \to \infty$). GSP finds a linear growth rate for this problem of $\gamma = 0.41$ and a frequency of $\omega = -1.155$. ASTROGK determines the growth rate as $\gamma = 0.408$ and the frequency of this instability as $\omega = -1.202$. So for this case we have overall a pretty good agreement for the linear growth behavior. When we run GSP nonlinearly and long enough for the code to reach a fully non-linear state we observe in the code that the integrated quantity ϕ^2 as well as the heat flux role over after reaching a peak value and start to decay while settling to reach a saturated state. We show the results of the collisionless nonlinear run with GSP in Fig. (5.6). We observe that although the heat flux and ϕ^2 are decreasing in the nonlinear phase the weights stop growing exponentially but continue to grow algebraicy. This is a concern since as the weights keep growing the accuracy of the measurements of physical quantities in the code such as the heat flux will be reduced. Therefore it is obvious that if we keep running the code longer and trying

to obtain measurements for the physical quantities in the system for a non-linear state that reached saturation eventually the code results will be dominated by noise and the measurements' accuracy will be questionable.

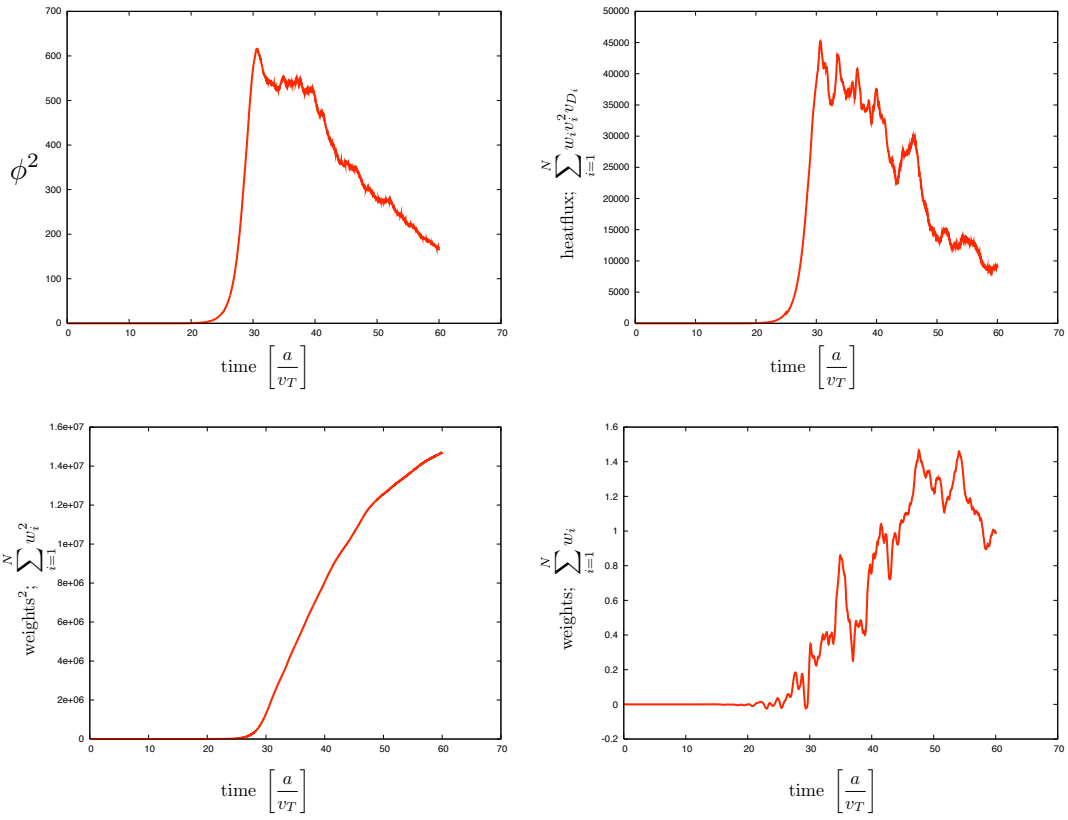

Figure 5.6: GSP results for ITG - turbulence. The results are obtained by running GSP without collisions and with a temperature gradient scale length of $L_T = 0.1$. Shown are the behavior of the integrated quantities for ϕ^2, heat flux, squared weights, and weights versus time.

Therefore we test now the three different operators that we added to the code to see how they can deal with reducing the growth of the weights in the non-

linear phase of the simulations. To find the parameters that are sufficient for the three different operators to succeed in stopping the squared weights from growing in the non-linear phase of the simulation we restart the simulation at time $t = 60\frac{a}{v_T}$ where we ended the collisionless nonlinear simulation before. When we restart the simulation we include one of the three operators to the code and tune its parameters such that the growth of the squared weights discontinues (see Fig. (5.7)-(5.9)). It is important to remember that the growth of the weights is not a problem of the numerical scheme we are using. The growing weights result directly from the set of equations we are modeling and are due to the fact that we left out the collisional term on the right hand side of the gyrokinetic equation, Eq. (1.18). The Krook and the coarse-graining operator are both artificial operators in the sense that they try to correct the fact that the physical collisions were left out in Eq. (2.1). In contrast the pitch-angle scattering operator added to GSP is a physical operator in the sense that it describes the collisional term on the right hand side of Eq. (1.18). It is not an operator that is artificial and tries to make up for the fact that a simplified set of equations is modeled. The pitch-angle scattering operator models the effects of the collisions onto the perturbed distribution function δf and therefore changes the particle weights w in our PIC - code.

We start to examine the effect of the various collision operators on the weight growth in the nonlinear phase of the simulation with the Krook operator. We can choose ν in Eq.(5.1) such that the weights squared stop increasing over time when we restart the collisionless run shown in Fig.(5.6) at time $t = 60\ \frac{a}{v_T}$ with the Krook operator turned on. As documented in Fig.(5.7), we observe that a relaxation

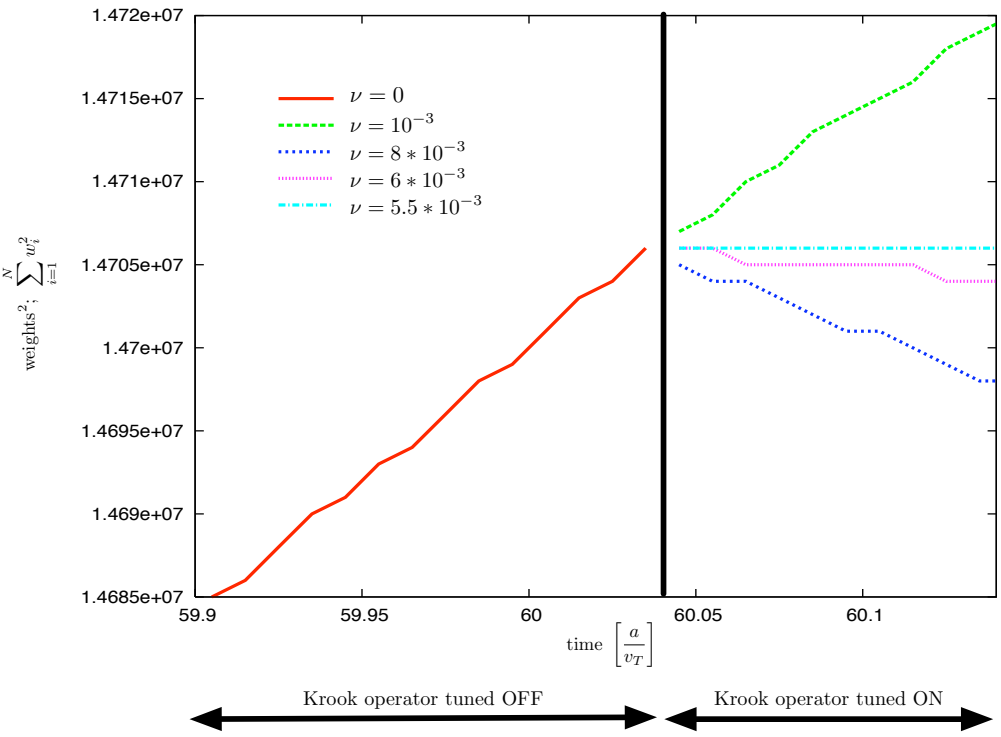

Figure 5.7: Restart of nonlinear run with Krook operator in order to determine the coefficient ν for the Krook operator that is sufficient to cancel out the growth of the sum of the squared particle weights

coefficient of $\nu = 0.0055$ for the Krook operator keeps the squared particle weights constant over time. We then restart the entire run from the beginning with the Krook operator turned on and ν set to be 0.0055. We restart the run from the beginning instead of just restarting it at time $t = 60\ \frac{a}{v_T}$ to rule out that we have to deal with some sort of hysteresis effect from running the code collisionless from time $t = 0$ to $t = 60$. We want to study how the heat flux and the linear growth will be affected by the Krook operator. Those results are presented together with

the results from the other collision operators in Fig. (5.10) - Fig. (5.12).

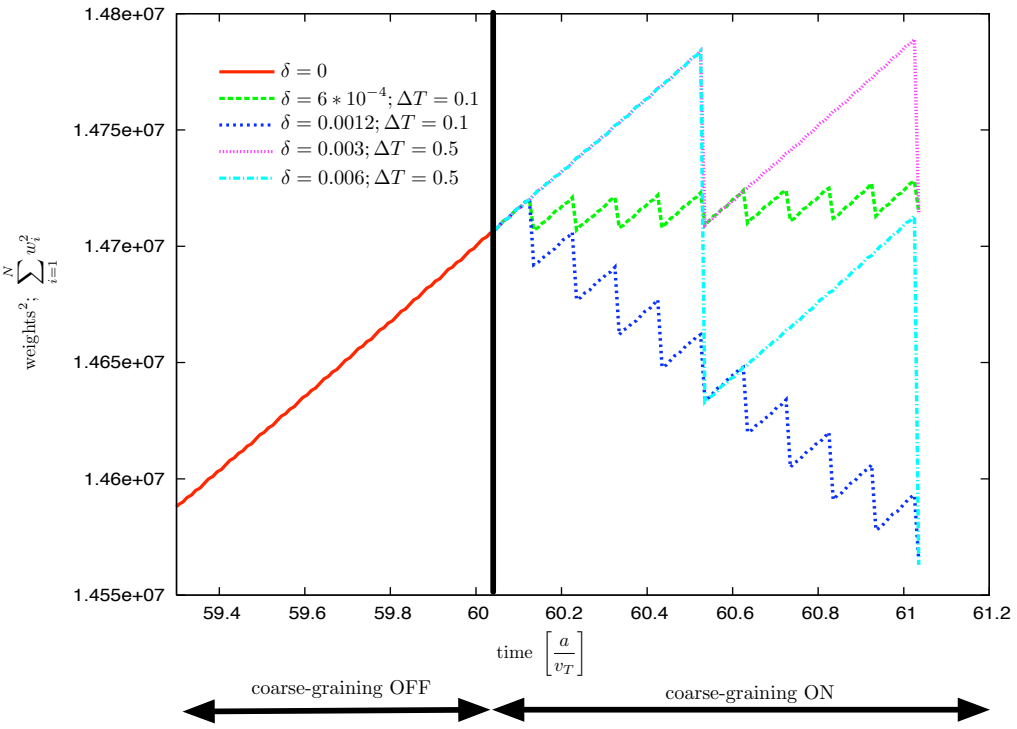

Figure 5.8: Restart of nonlinear run with coarse-graining operator. The damping of the weight growth is linearly proportional to the lag average parameter δ and the time step ΔT that lies between consecutive coarse-graining operations

We choose a similar approach for evaluating the coarse-graining operator. Again we first determine the parameters that will eliminate the weight growth when we restart the nonlinear run at time $t = 60\ \frac{a}{v_T}$ with coarse-graining turned on. A further complication is the fact that the coarse-graining operator has two parameters that we can choose. The first parameter is the factor δ in Eq.(4.5) that determines the fraction to which the individual weight is set to the weight that are found by

interpolation on the 5D grid. The second parameter is the time between coarse graining operations, ΔT. When Parker and Chen [14] are studying the Cyclone base case they are applying the coarse graining operation every 10 time steps and work with the parameter δ in the range between 0.05 and 0.1. Our studies suggest these parameters are way too dissipative for our simulations. The reason for this difference is that we are using a lot more particles in our simulations than Parker and Chen did. For the runs shown in Fig.(5.6) we are using 32 grid points in all three spatial dimensions and have ~ 19 million particles. For the energy/pitch-angle grid we are using a total of 256 grid-points. That means that we have a total of $2^{23} \sim 8.4$ million phase-space grid-points. Since we are using a bilinear interpolation scheme for the spatial components of the phase-space and a nearest-neighbor interpolation in velocity space we end up with interpolation on average ~ 18 particles onto one grid-point in phase-space. This might not seem a lot but is a significantly larger amount than Parker and Chen used. They did not publish their exact numbers, but stated that their number of particles is much smaller than their number of 5D grid-points. Thus, when we identify in Fig. (5.8) the right values for the parameters of the coarse-graining operation we find that $\delta = 6 * 10^{-4}$ and $\Delta T = 0.1$ achieve to keep the weights constant. The amount of dissipation for coarse-graining is proportional to the lag average parameter δ and inversely proportional to the time-step size ΔT. So we can get the same results when we use a δ-parameter that is larger by a factor of 5 and at the same time apply the operator 5 times less often by making ΔT 5 times bigger in size. As long as we are working with parameters that are small enough these linear relationships hold. When restarting the ion-temperature-

gradient-driven simulation shown with the coarse-graining operator turned on in the code we use the $\Delta T = 0.5$ and $\delta = 0.003$ since Fig. (5.8) shows these parameters also control the weights and we can save computational time since we have to determine the full 5-D distribution function 5 times less often compared to the set of parameters that are $\delta = 6 * 10^{-4}$ and $\Delta T = 0.1$.

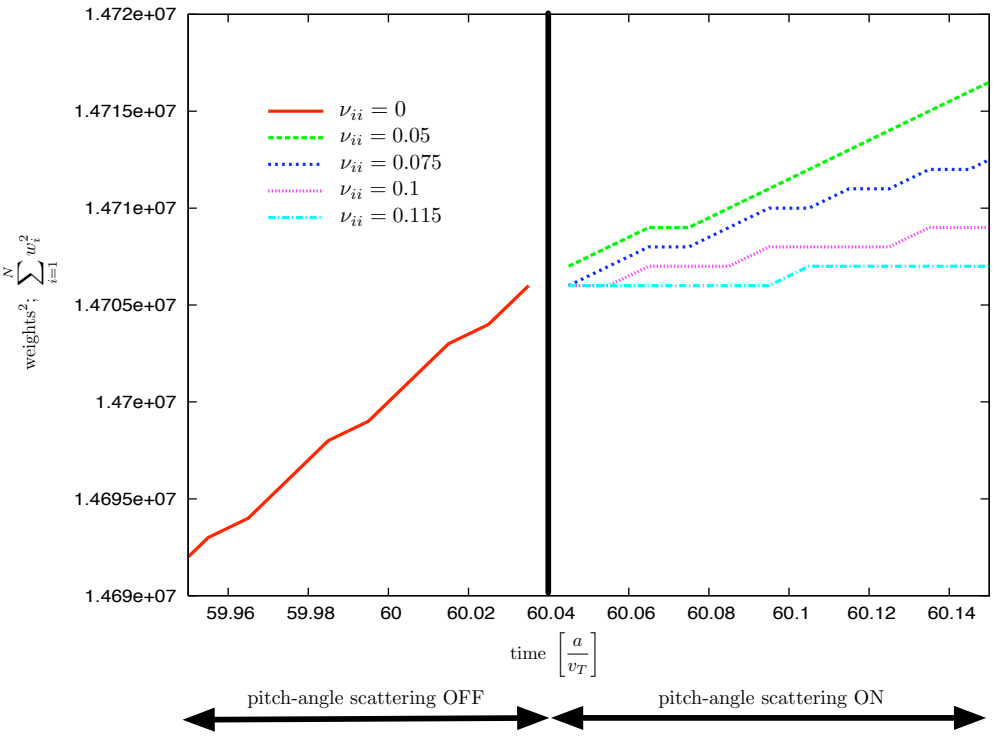

Figure 5.9: Restart of nonlinear run with pitch-angle collision operator in order to determine the coefficient ν_{ii} that produces the right amount of damping to cancel out the growth of the sum of the squared particle weights

For the pitch-angle scattering operator we go through the same parameter finding procedure as we did for the other two collision operator and identify the

collisionality for the ion-ion collisions that is large enough to damp out the weight growth of the particles in the ITG - simulation with the set of parameters used to create Fig.(5.6) . The collision frequency we find to sufficient for canceling the weight increase is $\nu_{ii} = 0.115$ as we show in Fig. (5.9). This value will depend on the resolution of the velocity space grid in general.

Now that we have found the parameters for the Krook, coarse-graining and pitch-angle scattering operator that are large enough to stop the squared weights from linearly increasing, over time we can restart in GSP the same run that was collisionless before but now with those collision operators turned on. That way we can study the effect the operators have on the physical quantities we are measuring with the code. In Fig. (5.10) - (5.12) we show ϕ^2, the heat flux Γ_e, and the sum over squared particle weights for the collisional runs with the Krook-operator, the coarse-graining, and pitch-angle scattering respectively turned on.

In Fig. (5.10) we are comparing the integrated electrostatic potential in the four different runs on a linear and on a logarithmic scale. Note that since we are redoing the runs that we used for Fig. (5.6) we have a temperature gradient scale length of $L_T = 0.1$ and no density gradient in the simulation. The growth rates and frequencies during the linear growth phase become in the four different cases: Furthermore we see from Fig. (5.10) that the overall qualitative behavior for the integrated squared electrostatic potential over time is similar in the four runs. After a linear growth phase the runs enter a non-linear phase around the same time and start to decay towards a saturation level. Since we introduce dissipation with all three collision operators that we have included into the code we see that the overall

Run	**collision parameters**	**growth rate γ**	**frequency ω**
Collisionless	-	0.41	-1.155
Coarse-graining	$\delta = 0.003$, $\Delta T = 0.5$	0.405	-1.150
Krook	$\nu = 0.0055$	0.398	-1.148
Pitch-angle scattering	$\nu_{ii} = 0.115$	0.39	-1.135

Table 5.1: Growth rates for linear phase of ITG-driven simulation and its dependence on the kind of collision operator included to control the weight growth in the nonlinear phase.

level for the electrostatic potential is lower in all three cases with collisions compared to the collisionless case. In addition, we notice that the Krook operator introduces the strongest limitation to the linear growth and the maximal value that ϕ^2 reaches is the lowest for the case that includes the Krook operator.

The levels that we find for the heat fluxes at the end of the simulations are similar for all four runs but lower by a factor of 2 for the case that uses the Krook operator. The qualitative behavior is very similar in all cases but it is hard to draw a final conclusion whether or not the values for the saturated heat fluxes differ in the four cases since we didn't run the code long enough in this example to be sure that we reached a saturated level. Therefore it would be desirable to run the code longer and see how the heat fluxes keep developing over time. We are planning on

doing this test shortly.

In Fig. (5.12) we can see how well we reached our goal to stop the squared weight from growing in the nonlinear phase of the simulation. We can see that the three forms of collisional terms in the code all succeed in reducing the weight growth. For the Krook operator as well as for the coarse-graining operator we observe linear growth during the non-linear phase of the simulation that is a lot lower than for the collisionless case. In those both cases the weights keep increasing, though at a much smaller rate than in the collisionless simulation. The pitch-angle scattering operator is the only one of the three that actually succeeds in stopping the weight growth entirely. Indeed it even leads to a reducing of the squared weights during the linear phase. The pitch-angle operator reaches this while for the collision frequency that we are using for this case the damping of the growth of ϕ doesn't seem to be stronger than in the other two cases.

To be able to interpret the results and their differences we are investigating how the weight distribution in velocity space differ for the three different collisional terms in the simulation. In Fig. (5.13) we plot for one point in physical space the value of the average weights that we find on the energy, pitch-angle grid at different times during the collisionless non-linear simulation. In Fig. (5.14)-(5.16) we show how for the three cases with collisions the weight distribution which is proportional to the perturbed distribution function develops over time under the influence of the different collisional terms in the code.

The first observation that we are making is that there is a remarkable difference for the linear and non-linear part of the simulation. We see that during the

linear phase of all four cases the dominant structures in velocity space have large spatial scales. Those scales in velocity space are broken up into smaller scales during the nonlinear phase of the simulation. For the run with the Krook operator as well as for the run that includes the coarse-graining operation those small scale structures look very similar to the ones in the collisionless case. But both artificial collision operators succeed in reducing the overall amplitude of the structures in velocity space. For the run that includes pitch-angle scattering we see that during the non-linear phase the structures in velocity space for small values of energy E are not present. This is a desired effect from the collision operator. As we showed in Fig. (5.3) pitch-angle collisions smooths out structure in velocity space within one energy band. Since the collisionality drops as the value of energy increases (see Eq. (5.7)) the smoothing due to pitch-angle scattering is strongest for small energy values. In Fig. (5.16) we see that the pitch-angle operator indeed smooths out structures in velocity space most efficiently for the low energy values. When we are evaluating integrals in the code we need to know the perturbed distribution function δf which is defined as the product of the Maxwllian F_0 times the distribution for the weights. So the smoothening of the velocity space due to pitch-angle collisions is there strongest where the values for δf are largest.

We also show snapshots of the time evolution of the electrostatic profile in the x, y-direction in the code for the collisionless case as well as for the three different cases with collisions. These plots show the linear instability that grows the $k_x = 0, k_y = 1$ mode. At times later than $t = 25\,[a/v_T]$ the code becomes nonlinear and

we observe the formation of eddies. Besides an overall damping of the profile in the three collisional cases the results look fairly similar in all four cases. But the simulation with pitch-angle collisions show a profile for ϕ that is smoother than the ones for the other simulations. That the collision operator does not just smooth the velocity-depend part of the distribution function but that the smoothing of the velocity dependence feeds back onto the spatial dependence of the profiles is an expected result. The fact that the pitch-angle collision operator is the only one of the three operators is another reason that makes it seem better suited for being the operator to include in a particle code.

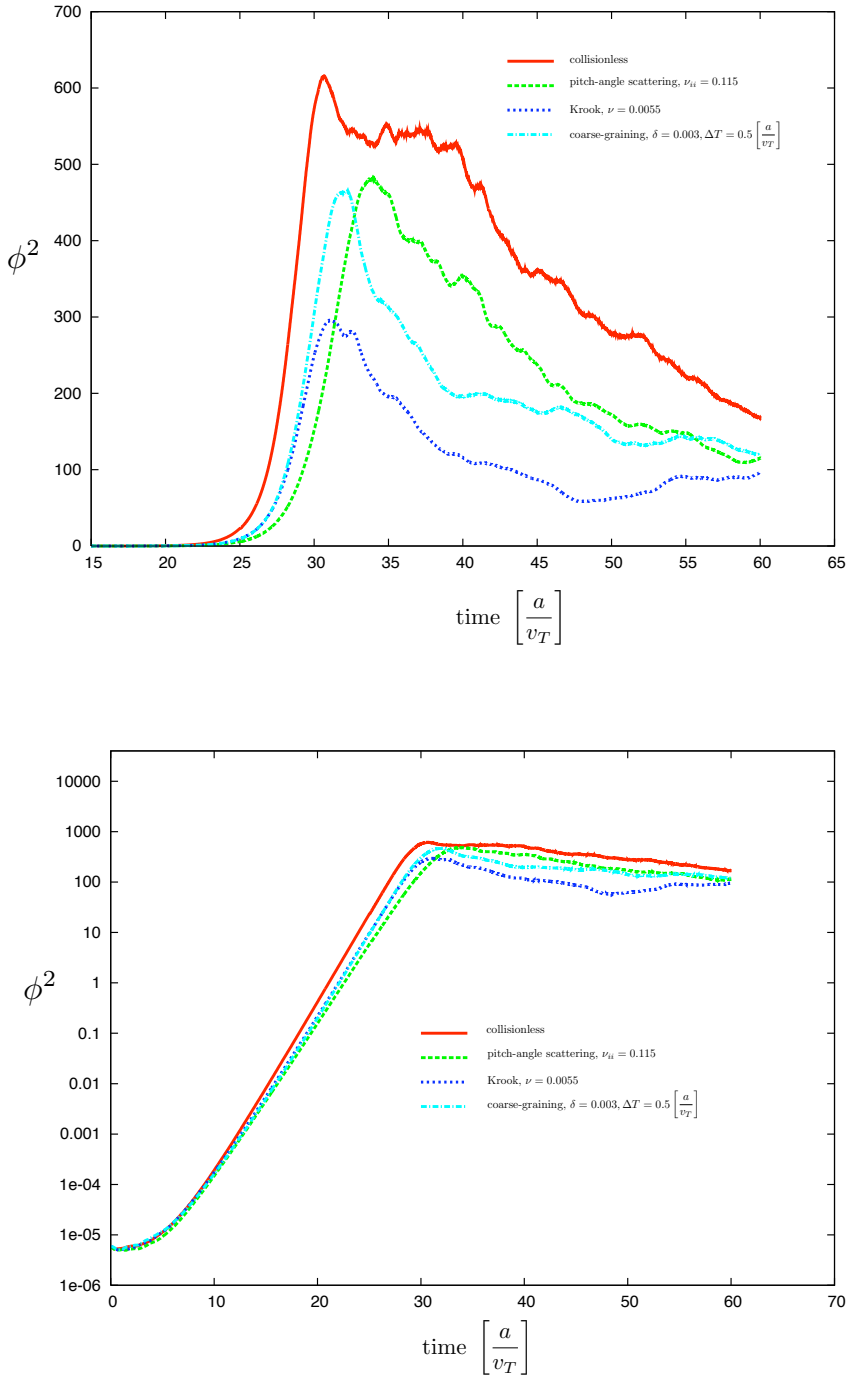

Figure 5.10: Results from Fig. (5.6) compared to the same run with either pitch-angle collision, coarse-graining or the Krook operator included in the code. The linear growth rate is slightly reduced in the collisional cases compared to the collisionless result

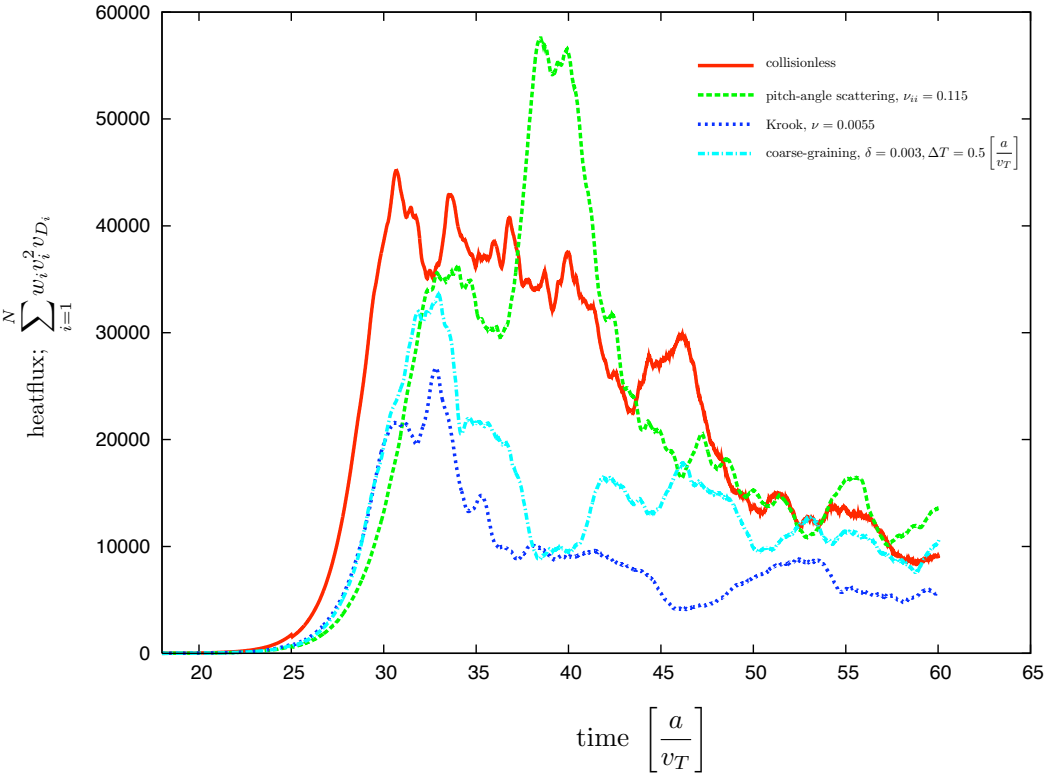

Figure 5.11: Results from Fig. (5.6) compared to the same run with either pitch-angle collision, coarse-graining or the Krook operator included in the code. The heat flux is plotted versus time.

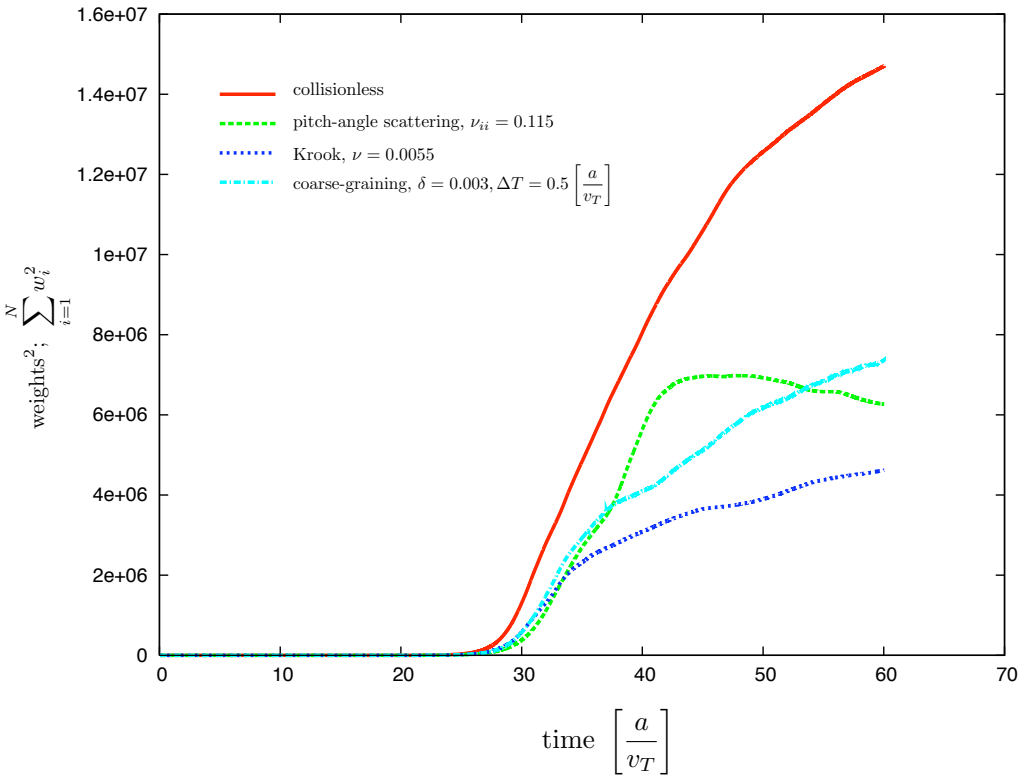

Figure 5.12: Results from Fig. (5.6) for the time time evolution of the sum of the squared particle weights versus time compared to the same run with either pitch-angle collision, coarse-graining or the Krook operator included in the code.

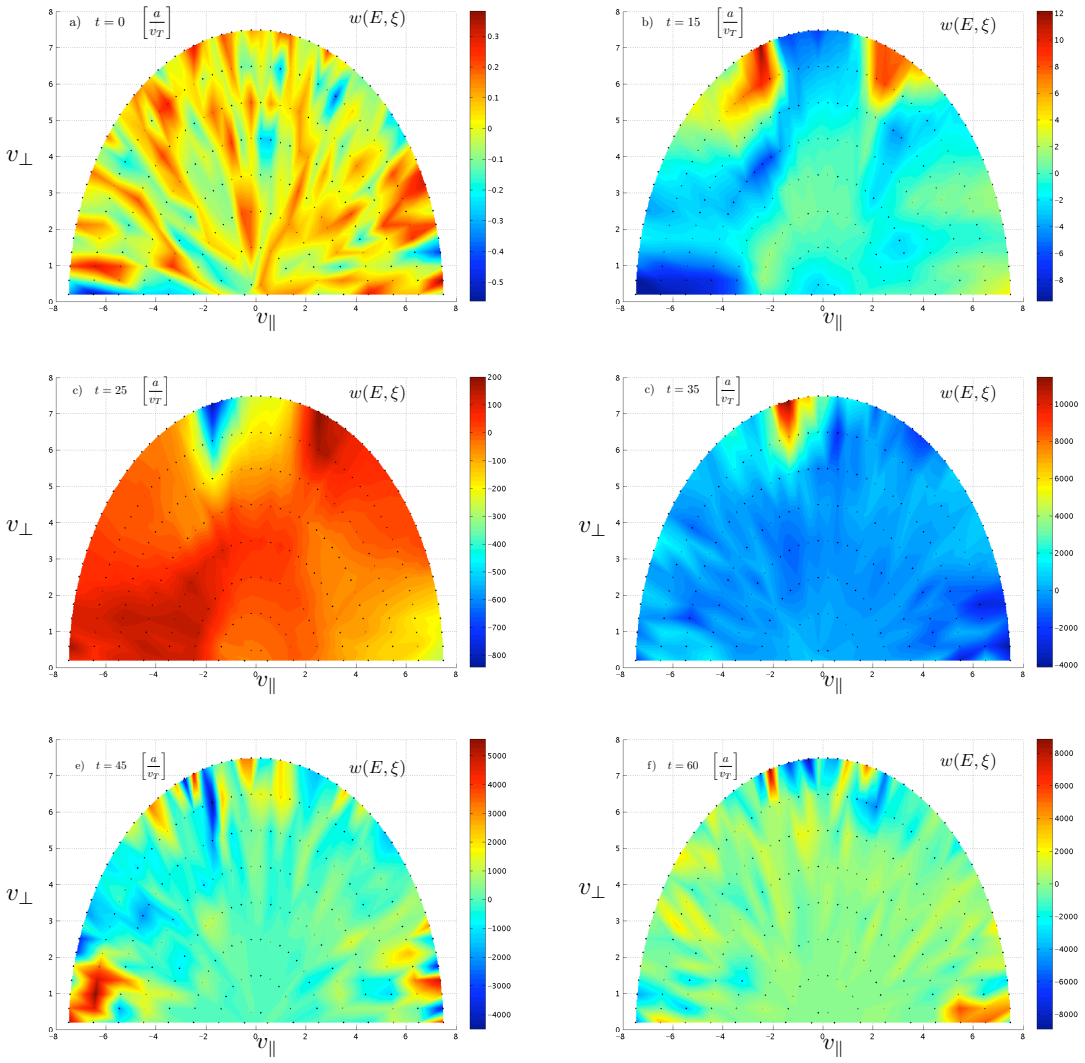

Figure 5.13: Velocity space dependence of particle weight distribution for collisonless ITG - simulation

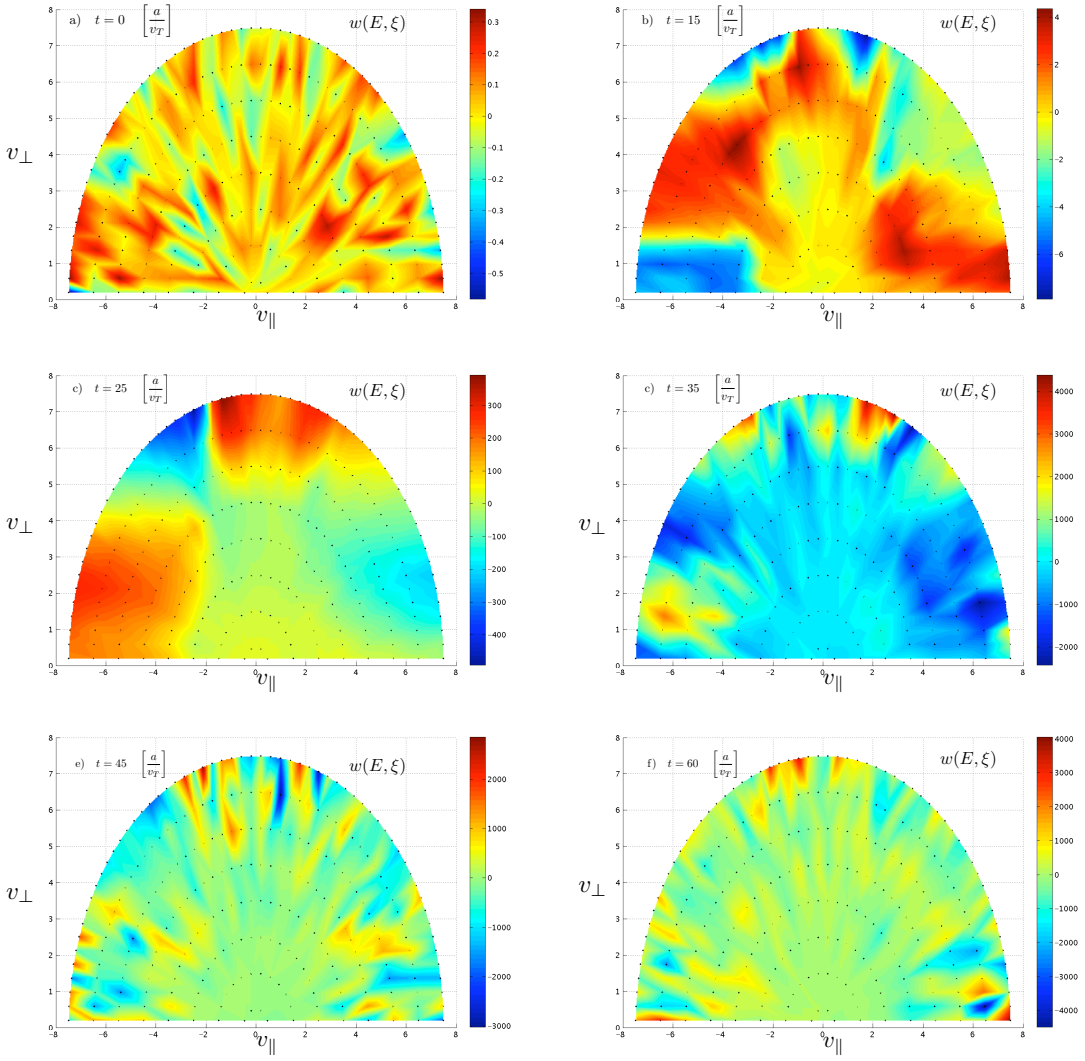

Figure 5.14: Velocity space dependence of particle weight distribution for ITG - simulation with Krook operator ($\nu = 0.0055$).

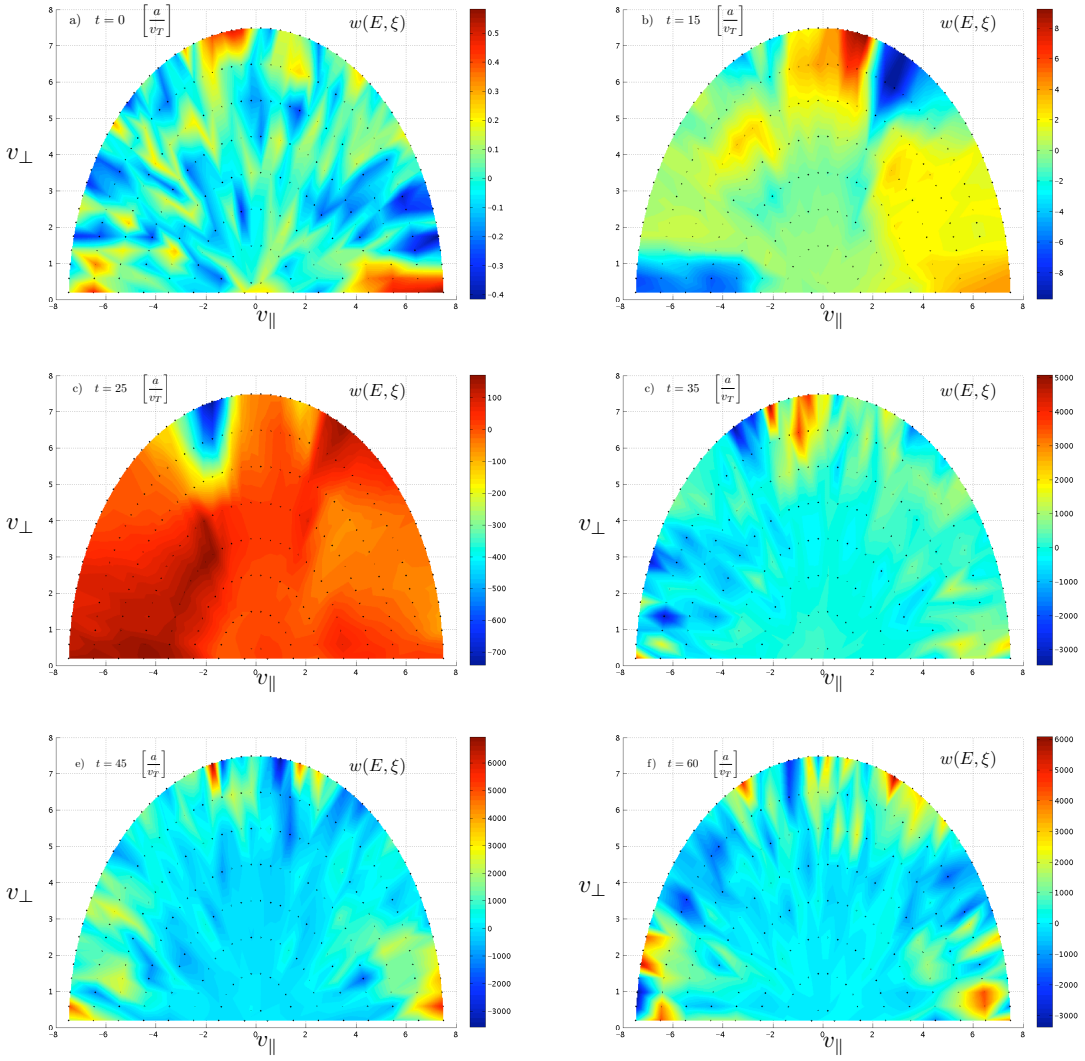

Figure 5.15: Velocity space dependence of particle weight distribution for ITG - simulation with coarse-graining operator ($\delta = 0.003;\ \Delta T = 0.5$)

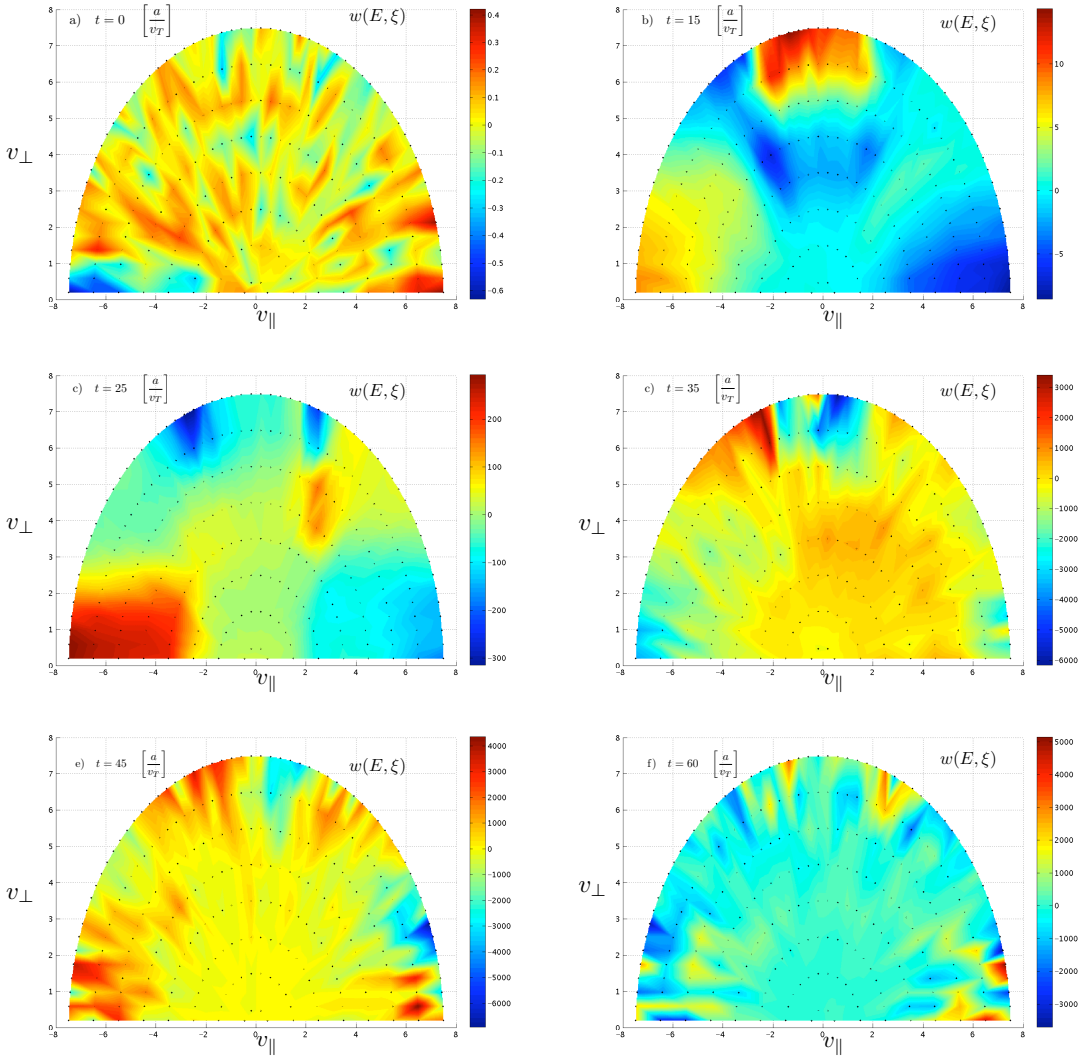

Figure 5.16: Velocity space dependence of particle weight distribution ITG - simulation with pitch-angle collisions ($\nu_{ii} = 0.115$).

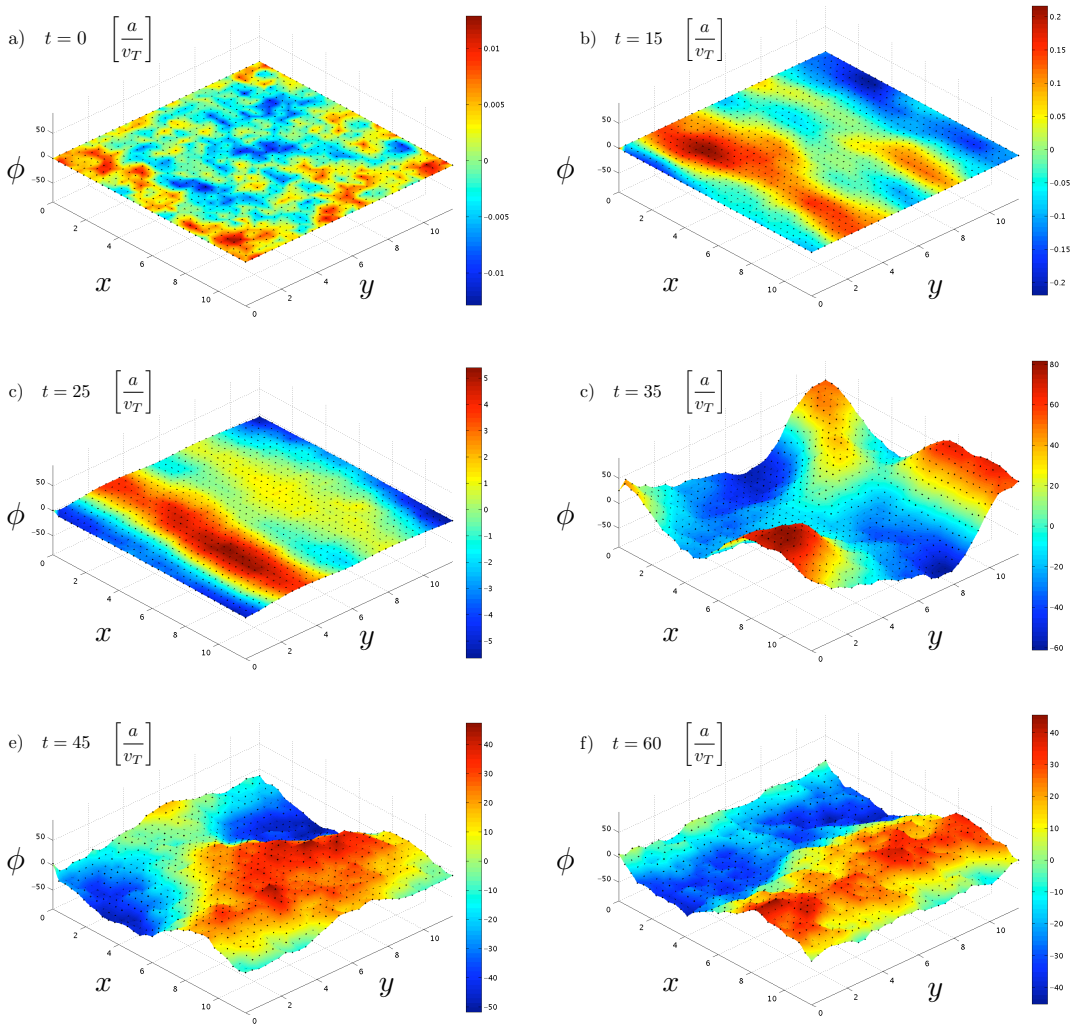

Figure 5.17: Series of snapshots showing the time evolution of the x, y-profile of ϕ for a nonlinear collisionless ITG-driven turbulence simulation with GSP.

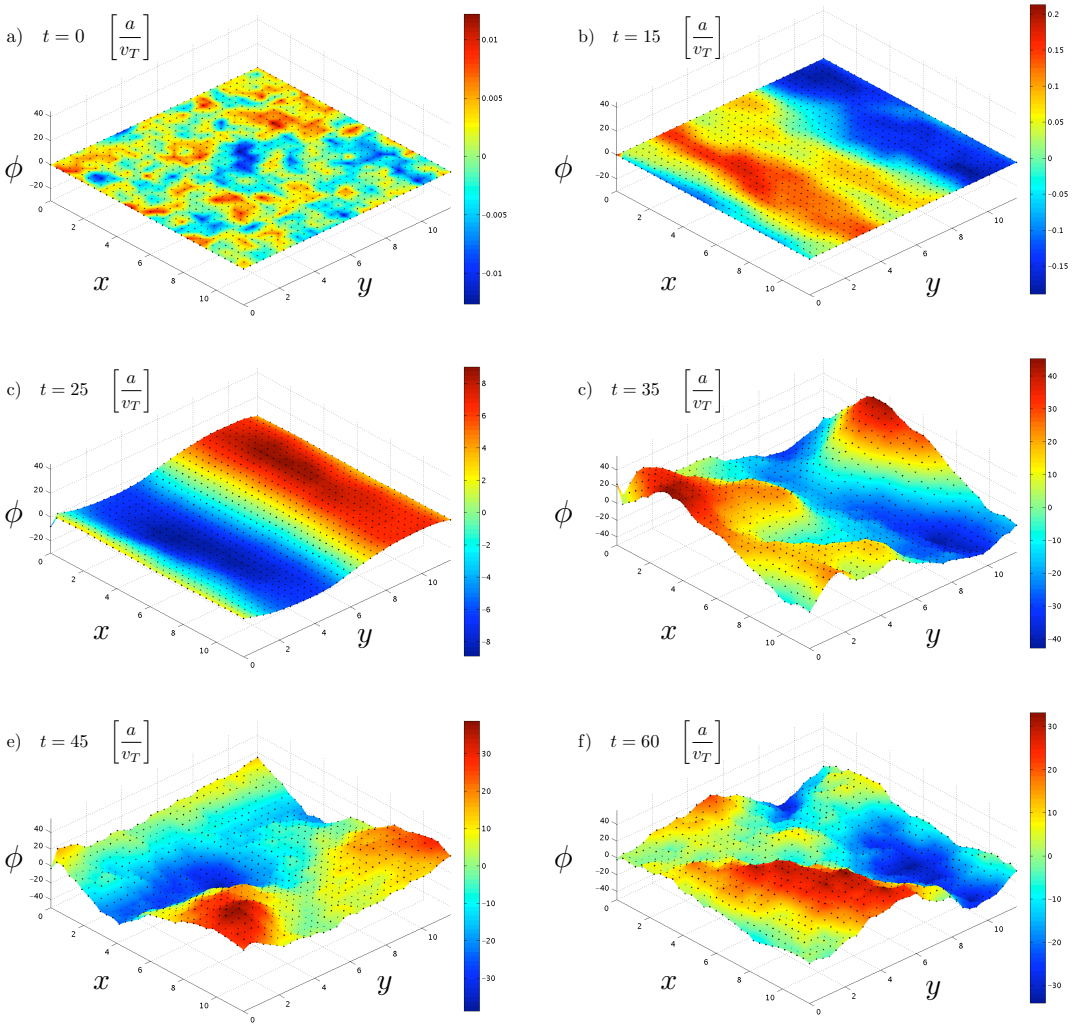

Figure 5.18: Series of snapshots showing the time evolution of the x, y-profile of ϕ for a nonlinear ITG-driven turbulence simulation with GSP that included the Krook operator.

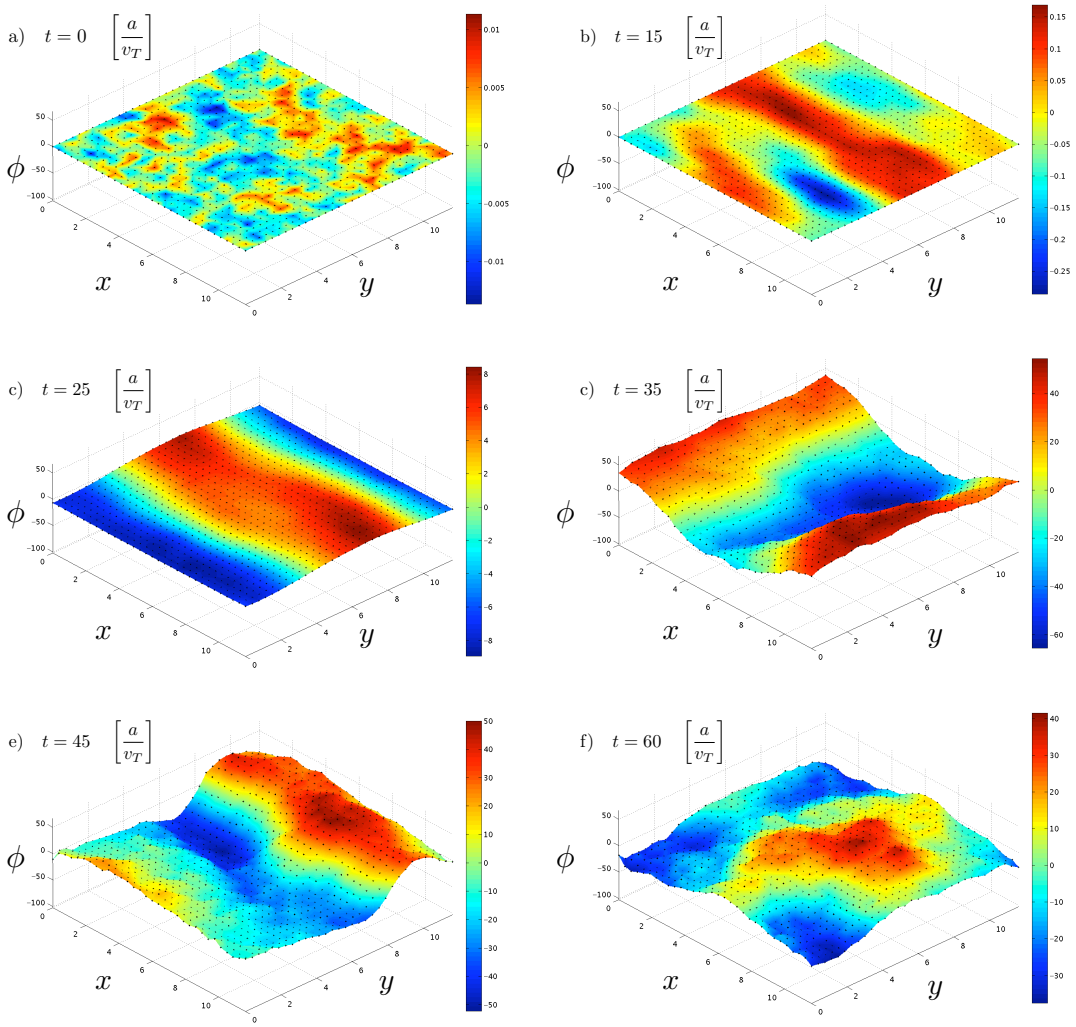

Figure 5.19: Series of snapshots showing the time evolution of the x, y-profile of ϕ for a nonlinear ITG-driven turbulence simulation with GSP that includes coarse-graining

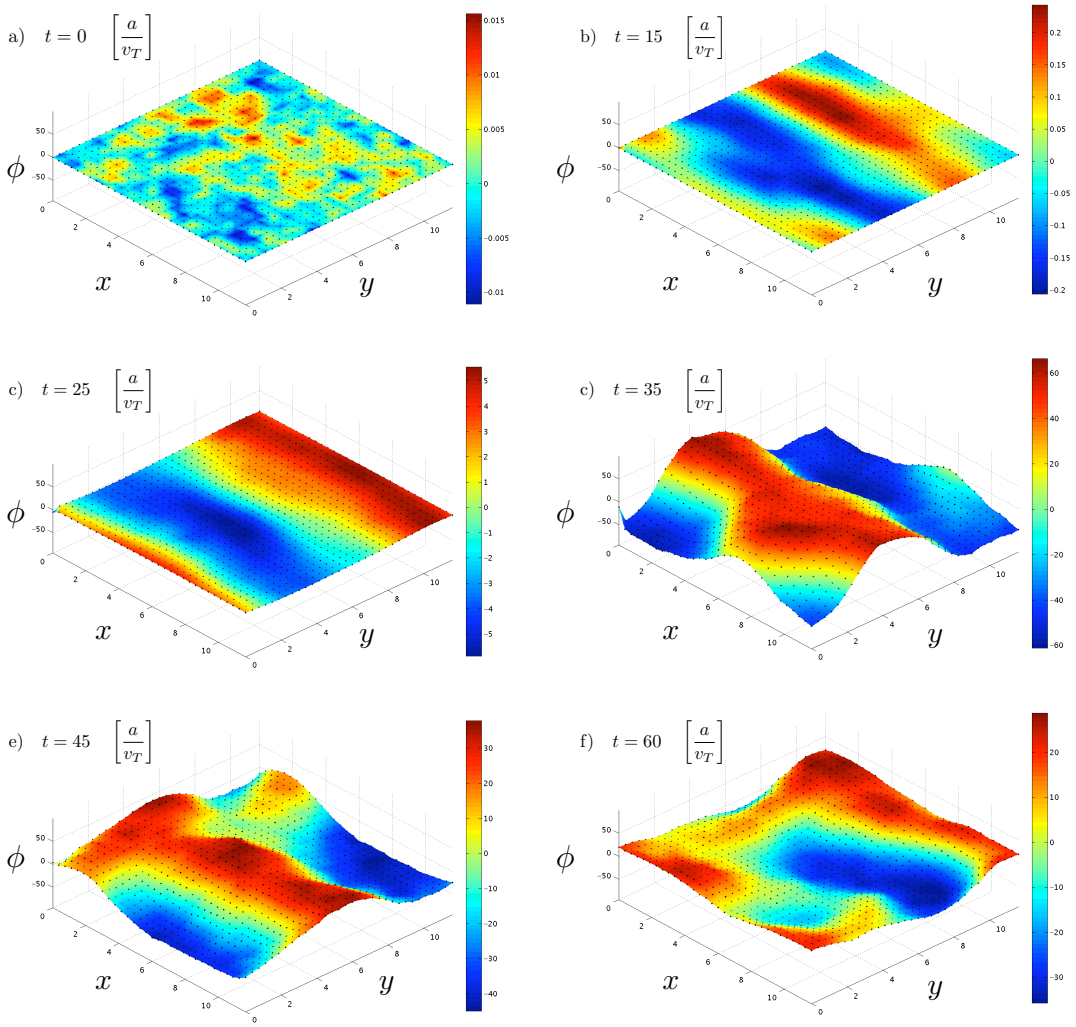

Figure 5.20: Series of snapshots showing the time evolution of the x, y-profile of ϕ for a nonlinear ITG-driven turbulence simulation with GSP that includes the pitch-angle scattering operator

Chapter 6

Z-pinch

In this chapter we are no longer studying a slab geometry, but instead move to a closed-field-line geometry. We are going explain the changes to the gyrokinetic equation based on moving from a slab to the Z-pinch configuration and also will explain how we implemented those changes to the code. The motivation for studying the Z-pinch problem comes from the work done by Ricci *et al.* [54] who studied the small-scale entropy modes in the Z-pinch in a low-β parameter regime. The regime they were investigating was stable to the ideal interchange mode. They were able to find variations in the particle and heat flux as a function of plasma collisionality and the density gradient. The instability they observed is called the entropy mode. Since the regime in which they were making those observations was unstable for large values of $k\rho_i$ it poses a challenge to the standard δf-particle codes, since those codes are restricted to $k\rho_i$ values of order one or smaller. In addition those codes lack a physical collision operator and would struggle to run for long periods of time. Since GSP includes a pitch-angle scattering operator and in addition uses a novel scheme to explicitly evaluate the first order Bessel functions for calculating gyro-

averages it captures all the physics that are needed to benchmark the results that Ricci *et al.* found using the continuum code GS2 .

6.1 Particle drifts in the Z-pinch

Before we go into the details of the Z-pinch geometry we explain briefly how to derive drift velocities for plasmas that are moving under the influence of a strong, curved background magnetic field.

We are looking at a particle that is gyrating around a magnetic background field which is pointing in the $\hat{z}$-direction. If the particle is exposed to an additional force that is perpendicular to the magnetic field the motion of the particle will be a spiral in the (x, y)-plane, see Fig. (6.1). To explain this we recall that the gyro-radius is given by $\rho_s = \frac{v_\perp}{\Omega_s}$ and therefore acceleration of the particle leads to a larger gyration radius. The acceleration that is due to the additional force $\mathbf{F}_\perp$ leads to a Lorentz force that is perpendicular to the the particle drift velocity.

We can easily determine the drift velocity by averaging the acceleration over several gyration periods and see that the net acceleration must be zero. So we end up with the following equation,

$$0 = \mathbf{F}_\perp + \frac{q_s}{c}\mathbf{v}_D \times \mathbf{B}_0. \tag{6.1}$$

From this expression we find $\mathbf{v}_D$ by taking the cross product of Eq.(6.1) with $\mathbf{B}_0$,

$$\mathbf{v}_D = \frac{c}{q_s}\frac{\mathbf{F}_\perp \times \mathbf{B}_0}{B_0^2}. \tag{6.2}$$

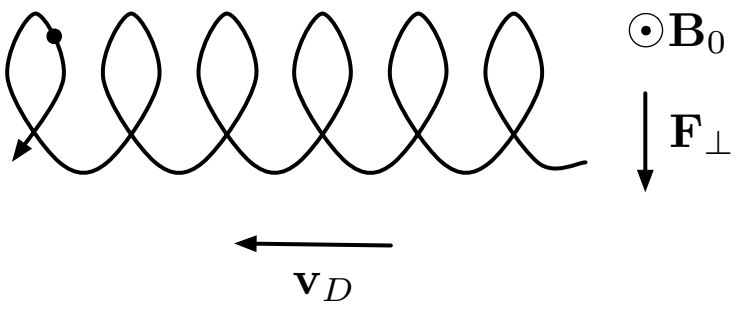

Figure 6.1: Drift of a positive charged ion with a strong background magnetic field and a force perpendicular to the direction of the magnetic field.

6.1.1 ∇B_0 - Drift

For studying the Z-pinch we no longer assume that the background magnetic field is homogenous. Instead we allow a dependence of the field strength on the y-position, $B_0 = B_0(y)$. We then Taylor expand the magnetic field around the guiding center position

$$\mathbf{B} = \mathbf{B}_0 + (\mathbf{r} \cdot \nabla)\mathbf{B}_0$$

and split the particle velocity into a background part $\mathbf{v}_\perp$ and perturbed part $\mathbf{v}_D$. The perturbed part describes the drift associated with the gradient in the magnetic field ($\mathbf{v} = \mathbf{v}_\perp + \mathbf{v}_D$). $\mathbf{B}_0$ is measured at the guiding center position and $\mathbf{r}$ is measured from the guiding center position. By splitting the acceleration of the particle in first and second order terms we get the following two expressions:

$$\frac{d}{dt}\mathbf{v}_\perp = \frac{q_s}{m_s c}\,\mathbf{v}_\perp \times \mathbf{B}_0 \quad \text{gyromotion,} \tag{6.3}$$

$$\frac{d}{dt}\mathbf{v}_D = \frac{q_s}{m_s c}\left(\mathbf{v}_D \times \mathbf{B}_0 + \mathbf{v}_\perp \times (\mathbf{r} \cdot \nabla)\,\mathbf{B}_0\right). \tag{6.4}$$

In order to determine the steady drift velocity $\mathbf{v}_1$ we average Eq.(6.4) over one gyration period which makes the LHS vanish and we are left with:

$$< \mathbf{v}_D \times \mathbf{B}_0 >_{\mathbf{R}} = < \mathbf{v}_\perp \times (\mathbf{r} \cdot \nabla)\mathbf{B}_0 >_{\mathbf{R}} \tag{6.5}$$

$$\Rightarrow \quad \mathbf{v}_D = \frac{1}{B_0^2} < (\mathbf{v}_\perp \times (\mathbf{r} \cdot \nabla)\mathbf{B}_0) \times \mathbf{B}_0 >_{\mathbf{R}} . \tag{6.6}$$

Using the following identities and definitions we can solve for $\mathbf{v}_D$ in a straightforward manner:

$$\mathbf{r} = \left(-\frac{v_\perp}{\Omega}_s \cos(\Omega_s t), \frac{v_\perp}{\Omega_s} \sin(\Omega_s t), 0 \right) \tag{6.7}$$

$$\mathbf{v}_\perp = (v_\perp \sin(\Omega_s t), v_\perp \cos(\Omega_s t), 0) \tag{6.8}$$

$$(\mathbf{r} \cdot \nabla)\, \mathbf{B}_0 = y \frac{\partial B_0}{\partial y} \hat{\mathbf{z}} \tag{6.9}$$

$$< \sin^2(\Omega_s t) >_{\mathbf{R}} = \frac{1}{2} \tag{6.10}$$

and find:

$$\mathbf{v}_D = \frac{v_\perp^2}{2 B_0^2 \Omega_s} (\mathbf{B}_0 \times \nabla B_0) . \tag{6.11}$$

Since Ω_s depends on the charge of the species s the ∇-B-Drift is in opposite directions for electrons and ions.

6.1.2 Curvature Drift

We are now going to look at a toroidal geometry. The magnetic field lines are no longer straight in the $\hat{z}$-direction. Instead the field lines bend to form a ring. When simulating a flux tube in our code the geometry becomes a torus. In Fig.(6.2) we show the transformation of coordinate systems from the slab geometry that is

using a Cartesian coordinate system to the cylindrical coordinate system of the Z-pinch. The background magnetic field is now pointing in the negative $\hat{\phi}$-direction. ($\hat{\mathbf{B}}_0 = -\hat{\phi}$).

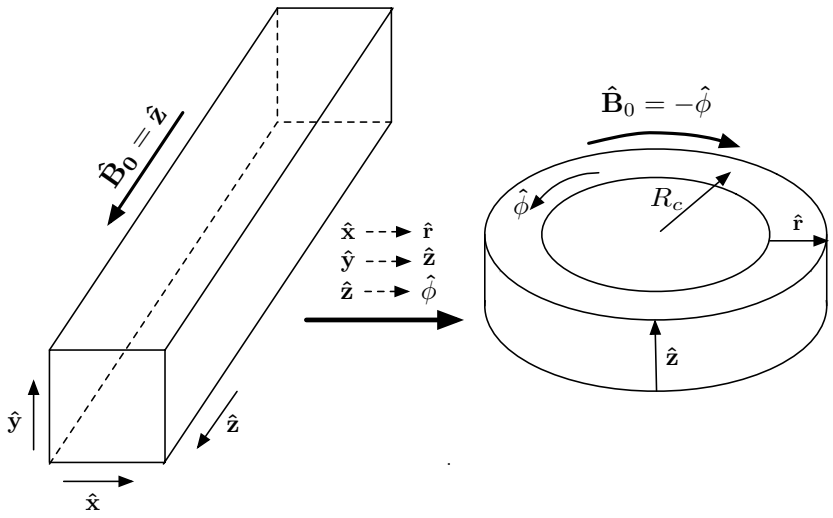

Figure 6.2: Sketch of the change of coordinate system from slab to cylindrical for the Z-pinch problem.

In the code we do not need to implement a complete new coordinate system. As pointed out with the dashed arrows in Fig. (6.2) the old $\hat{\mathbf{x}}$-direction now represents the radial direction, the old $\hat{\mathbf{y}}$-direction describes the $\hat{\mathbf{z}}$-direction in the cylindrical geometry of the Z-pinch and the $\hat{\mathbf{z}}$-direction becomes the new $\hat{\phi}$-direction. So in the code we are still using x, y and z. They now just represent a different coordinate system. We do not need to make an additional changes to the code.

The motion of a particle is still described by a gyration around the field lines

while the particle streams down that field. The particle follows the field lines since the motion perpendicular to the field lines is resisted. But now that we added curvature to the problem the particle will be exposed to a centripetal force while following the field lines. The centripetal force, $\mathbf{F}_c$, points in the $-\hat{\mathbf{r}}$-direction. The force is given by:

$$\mathbf{F}_c = -\frac{m_s v_\parallel^2}{R_c}\hat{\mathbf{r}}. \tag{6.12}$$

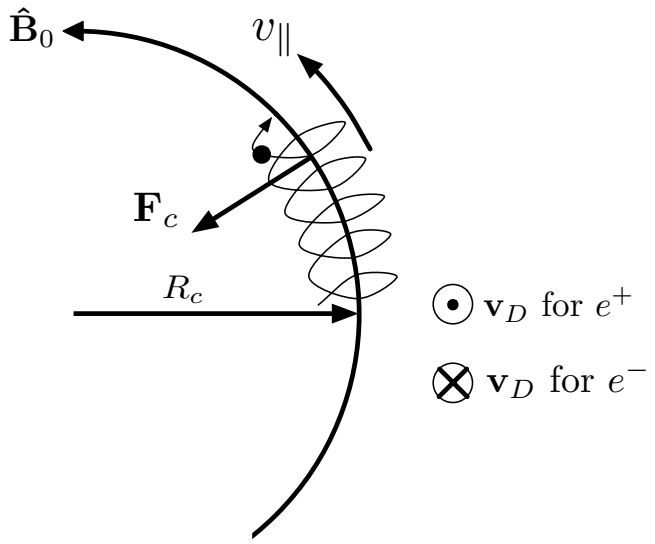

Figure 6.3: Sketch of the centripetal force felt by particles that are streaming along the background magnetic field. This force leads to a curvature drift that is pointing in opposite directions for electrons and ions

Since this force is perpendicular to the magnetic field we can plug it into Eq.(6.2) to find the following drift velocity associated with the curvature of the

magnetic field:

$$\mathbf{v}_d = \frac{v_\parallel^2}{\Omega_s R_c}\left(\hat{\mathbf{r}} \times \hat{\mathbf{B}}_0\right). \tag{6.13}$$

For a cylindrically symmetric vacuum magnetic field the gradient of B_0 turns out to be $\nabla B_0 = -\frac{B_0}{R_c}\hat{\mathbf{r}}$ [49]. Therefore we can add the curvature drift and the $\nabla - B$-drift to find the total drift:

$$\mathbf{v}_d^{tot} = \frac{2v_\parallel^2 + v_\perp^2}{2\Omega_s R_c}\left(\hat{\mathbf{r}} \times \hat{\mathbf{B}}_0\right). \tag{6.14}$$

6.1.3 Gyrokinetics equation with curvature and the new characteristics

When we add curvature to the problem and make the magnetic background field non-uniform we have to update the gyrokinetic equation accordingly. In Chapter 2, we introduced the electrostatic gyrokinetic equation and showed how to determine the characteristics and the time evolution for this system. In the gyrokinetic Eq.(1.18) only the term $d\mathbf{R}/dt$ that was defined in Eq.(1.20) will have to be change when going from the slab to the Z-pinch configuration. To Eq.(1.20) we just need to add the new drift velocity $\mathbf{v}_D^{tot}$ that was defined in Eq.(6.14).

The gyrokinetic equation for the Z-pinch can be written for the non-Boltzmann part h of the perturbed distribution function δf as [3]:

$$\begin{aligned}&\frac{\partial h}{\partial t} + \left(v_\parallel \mathbf{B}_0 + < \mathbf{v}_{E\times B} >_{\mathbf{R}} + \mathbf{v}_D^{tot}\right)\cdot \nabla h = \\ &< C(h) >_{\mathbf{R}} + \frac{qF_0}{T}\frac{\partial < \phi >_{\mathbf{R}}}{\partial t} - < \mathbf{v}_{E\times B} >_{\mathbf{R}} \cdot \nabla\left(F_0 - \frac{q\phi}{T}F_0\right).\end{aligned} \tag{6.15}$$

To find the new characteristics for the system that is described by this equation and to have a new equation that characterizes how the particle weights will behave over time we need to go through the same exercise as in chapter 2. First we write Eq. (6.15) in terms of $\delta f = h - \frac{q\phi}{T}F_0$ and recall that $< h >_{\mathbf{R}} = h$. We then find:

$$\frac{\partial}{\partial t} < \delta f >_{\mathbf{R}} + \left(v_\parallel \mathbf{B}_0 + < \mathbf{v}_{E\times B} >_{\mathbf{R}} + \mathbf{v}_D^{tot}\right) \cdot \nabla < \delta f >_{\mathbf{R}} = < C(h) >_{\mathbf{R}} - < \mathbf{v}_{E\times B} >_{\mathbf{R}} \cdot F_0 - v_\parallel \frac{qF_0}{T}\left(\hat{\mathbf{B}}_0 \cdot \nabla < \phi >_{\mathbf{R}}\right) - \mathbf{v}_d^{tot} \cdot \nabla \left(\frac{q < \phi >_{\mathbf{R}}}{T} F_0\right) \quad (6.16)$$

So using the same technique as introduced in section 2.1 we find the characteristics for this system to be:

$$\frac{d}{dt} R_\parallel = v_\parallel \quad (6.17)$$

$$\frac{d}{dt} \mathbf{R}_\perp = < \mathbf{v}_{E\times B} >_{\mathbf{R}} + \mathbf{v}_d^{tot} \quad (6.18)$$

Along those characteristics the perturbed distribution function δf is defined by the following ODE:

$$\frac{d}{dt}\delta f = \quad (6.19)$$

$$- \quad < \mathbf{v}_{E\times B} >_{\mathbf{R}} \cdot \nabla F_0 - v_\parallel \frac{qF_0}{T} \hat{\mathbf{B}}_0 \cdot \nabla < \phi >_{\mathbf{R}} - \mathbf{v}_d^{tot} \cdot \nabla \left(\frac{q < \phi >_{\mathbf{R}}}{T} F_0\right) \quad (6.20)$$

This leads to an expression for the time dependence of the particle weights. Since the drift velocities are perpendicular to the gradient of F_0 the term $\mathbf{v}_D^{tot} \cdot \nabla F_0 = 0$ in Eq. (6.19). Since the drift velocities point in the $\hat{\phi}$-direction we are left with the following expression for the time evolution of the particle weights:

$$\frac{d}{dt} w_i = - \left(< \mathbf{v}_{E\times B} >_{\mathbf{R}} \cdot \hat{\mathbf{r}} \left(-\frac{1}{L_n} + \frac{3}{2L_T} - \frac{v^2}{2L_T v_T^2}\right) + \frac{q v_\parallel}{T}\, \hat{\mathbf{B}}_0 \cdot < \mathbf{E} >_{\mathbf{R}}\right) + \frac{q}{T} \frac{2v_\parallel^2 + v_\perp^2}{2\Omega R_c}\, \hat{\mathbf{z}} \cdot < \mathbf{E} >_{\mathbf{R}} . \quad (6.21)$$

There are two main differences between the slab problem and the Z-pinch configuration. The first distinction is that the Z-pinch is actually periodic along the magnetic field lines. In the slab we needed to introduce artificially periodic boundary conditions in the z-direction of the code. We then needed to show that we simulate a domain that is several correlation lengths long in z. This was necessary in order to ensure that one end of the box is sufficiently decorrelated from the other end to obviate artificially constraining correlation effects. The second difference is that the magnetic field lines are now curved into circles. The $\hat{\mathbf{z}}$-direction in the slab becomes the $\hat{\phi}$-direction in the cylindrical coordinate system for the Z-pinch. Therefore the field lines are curved around the $\hat{\mathbf{z}}$-axis in the cylindrical coordinate system. The radius of the curvature is given by R_c. The $\hat{\mathbf{x}}$-direction in the slab becomes the $\hat{\mathbf{r}}$-direction in the cylindrical system. In addition, the magnetic field strength is no longer assumed to be spatial homogeneous but it has a profile that depends on the radial position, $\mathbf{B} = B_0(r)\hat{\phi}$. In the slab-geometry the gradient of B_0 points in the negative x-direction. When we transform the slab into the cylindrical system we have a freedom on whether the magnetic field will be pointing in the positive or negative $\hat{\phi}$-direction.

Now we need to look at Poisson' s equation and update it so that we are solving the right system for the Z-pinch. Since the direction of the drift velocities in the Z-pinch depends on the sign of the charge we need to add electrons as simulation particles to the code. Before, when we were studying the slab configuration, it was sufficient to assume that the electrons are adiabatic and we used the Boltzmann description for their density profile. This assumption was important for the way how

we solved Poisson's equation in chapter 2. Note that adding electrons as simulation particles to the code doubles the number of particles in the code and roughly slows down the code by 50%.

So far by treating the electrons as adiabatic we did not need to solve for the perturbed electron distribution function δf_e but instead assumed the solution $\delta f_e = \frac{q_e F_0}{T_e}\phi$. Making the electrons a species that we simulate explicitly in the code we have to perform the same steps that we have been doing for the ions when the used the Boltzmann assumption for the electrons. We need to advance the electron positions over time along their characteristics and update the electron particle weights over time.

The only part that we need to change in the algorithm of the code besides adding the electrons is in the way how we determine ϕ. As before when we ascertained ϕ we start with Poisson's Equation:

$$\nabla^2\phi = -\frac{1}{\epsilon_0}\left(q_i n_i + q_e n_e\right) \tag{6.22}$$

and assume quasineutrality. When writing the electron and ion charges as $q_s = |e|Z_s$ we end up with the following equation that we need to solve for ϕ.

$$Z_i = -Z_e n_e \quad \Leftrightarrow \tag{6.23}$$

$$Z_i \int \left\langle \left(< \delta f_i >_{\mathbf{R}} + \frac{Z_i|e|}{T_i} F_{0_i} \left(< \phi >_{\mathbf{R}} \right) \right) \right\rangle_{\mathbf{R}} d^3v = \tag{6.24}$$

$$-Z_e \int \left\langle \left(< \delta f_e >_{\mathbf{R}} + \frac{Z_e|e|}{T_e} F_{0_e} \left(< \phi >_{\mathbf{R}} \right) \right) \right\rangle_{\mathbf{R}} d^3v \quad \Leftrightarrow \tag{6.25}$$

$$Z_i \int J_0(k_\perp \rho_i) < \delta f_i >_{\mathbf{R}} d^3v + \frac{Z_i^2}{T_i} n_{0_i} \left(\Gamma_0(k_\perp^2 \rho_i^2) - 1 \right) \phi = \tag{6.26}$$

$$-Z_e \int J_0(k_\perp \rho_e) < \delta f_e >_{\mathbf{R}} d^3v + \frac{Z_e^2}{T_e} n_{0_e} \left(\Gamma_0(k_\perp^2 \rho_e^2) - 1 \right) \phi \quad \Leftrightarrow \tag{6.27}$$

$$\phi = \frac{\int \left(Z_i J_0(k_\perp \rho_i) < \delta f_i >_{\mathbf{R}} + Z_e J_0(k_\perp \rho_e) < \delta f_e >_{\mathbf{R}} \right) d^3 v}{\frac{Z_e^2 |e|}{T_e} n_{0_e} \left(1 - \Gamma_0(k_\perp^2 \rho_e^2)\right) + \frac{Z_i^2 |e|}{T_i} n_{0_i} \left(1 - \Gamma_0(k_\perp^2 \rho_i^2)\right)}. \tag{6.28}$$

So for the Z-pinch version of the code we got the following system of equation that we need to solve.

The characteristics are:

- $\frac{d}{dt} x_s = < \mathbf{v}_{E\times B} >_{\mathbf{R_s}} \cdot \hat{\mathbf{x}} = -\frac{\partial}{\partial y} < \phi >_{\mathbf{R}_s} \frac{c}{B}$
- $\frac{d}{dt} y_s = < \mathbf{v}_{E\times B} >_{\mathbf{R_s}} \cdot \hat{\mathbf{y}} + \mathbf{v}_{D_s}^{tot} = \frac{\partial}{\partial x} < \phi >_{\mathbf{R}_s} \frac{c}{B} - \frac{v_\parallel^2 + \frac{1}{2} v_\perp^2}{\Omega_s R_c}$
- $\frac{d}{dt} z_s = v_\parallel$.

The time evolution of the particle weights goes like

$$\begin{aligned} \frac{d}{dt} w_{i_s} &= -\frac{\partial}{\partial y} < \phi >_{R_s} \frac{c}{B} \left[-\frac{1}{L_{n_s}} + \frac{3}{2L_{T_s}} - \frac{v^2}{2L_{T_s} v_{T_s}^2} + \frac{Z_s |e| B}{c T_s} \frac{v_\parallel^2 + \frac{1}{2} v_\perp^2}{\Omega_s R_c} \right] + \\ &+ \frac{Z_s |e| v_\parallel}{T_s} \frac{\partial}{\partial z} < \phi >_{\mathbf{R}_s} . \end{aligned}$$

and Poisson's equation is

$$\phi = \frac{\sum_s \int Z_s J_0(k_\perp \rho_s) < \delta f_s >_{\mathbf{R_s}}}{\sum_s \frac{Z_s^2 |e|}{T_s} n_{0_s} (1 - \Gamma_0(k_\perp^2 \rho_s^2))}.$$

Notice that we wrote the the system of equation here for an arbitrary number of particle species. Indeed we also allow in the code for more than two species. For example it might be of interest to study a plasma that consists of several ion species or one is concerned about how impurities in the plasma are going to affect the plasma dynamics. By simply specifying the number of species and their physical properties (mass, temperature, density, charge, gradient length scales) in the input file to the code it is possible to run GSP with more than two species.

Before we can implement the set of equations that we gave above into the code we need to normalize the equations. But here this is rather straightforward since we already went through this task for the slab geometry version of the code and we have just two new terms that we need to normalize. For all the other terms we can use the normalization that we used before for the slab case.

The new terms are:

$$\frac{d}{dt}y_s = (\ldots) + \frac{v_\parallel^2 + \frac{1}{2}v_\perp^2}{\Omega_s R_c} \Leftrightarrow y_s^1 = y_s^0 + (\ldots) + \Delta t \frac{v_\parallel^2 + \frac{1}{2}v_\perp^2}{\Omega_s R_c} \tag{6.29}$$

$$\frac{d}{dt}w_{i_s} = (\ldots) + \frac{\partial}{\partial y} < \phi >_{R_s} \frac{c}{B}\frac{Z_s|e|}{T_s}\frac{v_\parallel^2 + \frac{1}{2}v_\perp^2}{\Omega_s R_c}$$

$$\Leftrightarrow w_{i_s}^1 = w_{i_s}^0 + (\ldots) + \Delta t \frac{\partial}{\partial y} < \phi >_{R_s} \frac{c}{B}\frac{Z_s|e|}{T_s}\frac{v_\parallel^2 + \frac{1}{2}v_\perp^2}{\Omega_s R_c} \tag{6.30}$$

Using the definitions for normalized quantities in the code given in Appendix B we get for Eq. (6.29) :

$$\begin{aligned} y_{s_N}^1 &= y_{s_N}^0 + (\ldots) + \frac{1}{\rho_{ref}}\Delta t_N \frac{a}{v_{T_{ref}}}\frac{v_{\parallel_N}^2 + \frac{1}{2}v_{\perp_N}^2}{R_{c_N}}\frac{v_{T_s}^2}{a\Omega_s} \\ &= y_{s_N}^0 + (\ldots) + \Delta t_N \frac{v_{\parallel_N}^2 + \frac{1}{2}v_{\perp_N}^2}{R_{c_N}}\frac{v_{T_s}^2\Omega_{ref}}{v_{T_{ref}}^2\Omega_s} \\ &= y_{s_N}^0 + (\ldots) + \Delta t_N \frac{v_{\parallel_N}^2 + \frac{1}{2}v_{\perp_N}^2}{R_{c_N}}\frac{T_{s_N}}{Z_{s_N}} \end{aligned} \tag{6.31}$$

and for Eq. (6.30) :

$$\begin{aligned} & w_{i_{s_N}}^1 = w_{i_{s_N}}^0 + (\ldots) + \\ & \frac{a^2}{\rho_{ref}v_{T_{ref}}}\Delta t_N \frac{1}{\rho_{ref}}\frac{\partial}{\partial y_N} < \phi_N >_{R_s} \frac{T_{ref}\rho_{ref}}{Z_{ref}|e|a}\frac{Z_s|e|}{T_s}\frac{v_{\parallel_N}^2 + \frac{1}{2}v_{\perp_N}^2}{\Omega_s R_{c_N}}\frac{v_{T_s}^2}{a} = \\ = \; & \Delta t_N \frac{\partial}{\partial y_N} < \phi_N >_{\mathbf{R_s}} \frac{v_{\parallel_N}^2 + \frac{1}{2}v_\perp^2}{R_{c_N}} \end{aligned} \tag{6.32}$$

So we are left with a normalized set of five equations that we are solving in the code. The numerical scheme does not need to be change from the slab configuration

that we discussed before. So the flow-chart in Fig. (5.1) that describes the numerical scheme can also be applied to the Z-pinch problem.

6.2 Entropy mode

When we study the small-scale turbulence in the closed-field-line geometry of the Z-pinch we consider the regime of plasmas with $\beta \ll 1$, in which the dominant instabilities have an electrostatic character and have $k_{\parallel} = 0$ [59]. Since the instabilities have no dependence in the $k_{\perp}$-direction we can run the code with only two spatial dimensions that are both perpendicular to the magnetic field. The dominant instabilities in the Z-pinch problem are driven by a combination of the pressure gradient and the magnetic curvature. Previous studies of this system [29, 38, 39, 59, 60] identified two distinct linear modes that can go unstable. The ideal magnetohydrodynamic (MHD) **interchange mode** and the non-MHD **entropy mode**.

At large pressure gradients the fastest growing mode is the interchange mode with a growth rate of

$$\gamma^2 = \frac{c_s^2}{\frac{2R_c}{L_p} - 7} \tag{6.33}$$

with the sound speed: $c_s \equiv \sqrt{\frac{T_e + T_i}{m_i}}$

and the pressure gradient scale length: $L_p \equiv -\frac{p_0}{\frac{d}{dr}p_0}$.

At weaker gradients ($L_p > \frac{2}{7}R_c$) the interchange mode is stabilized and the only unstable mode left is the entropy mode. This mode leads to perturbations in

both the temperature and the density but does leave the plasma pressure unchanged and therefore changes the specific entropy of the plasma. The entropy mode has been studied numerically by Hassam and Lee [29] using a fluid model and more recently by Kesner [38, 39] and Simakov [59, 60] with gyrokinetic models under the assumption of $k_\perp \rho_i \ll 1$. These gyrokinetic studies found for plasmas with equal ion and electron temperature that the entropy mode growth rate increases linearly in k. Consequently, the strongest growing mode lies outside the regime that could be addressed with their model that was restricted to low values of $k_\perp \rho_i$.

Ricci *et al.* studied the entropy mode using a gyrokinetic model for a collisionless plasma with $T_e = T_i$ and found that the entropy mode is unstable for density gradients that fulfill the relationship: $\frac{2}{7} < L_n/R_c < \frac{\pi}{4}$.

6.3 Linear results

Ricci *et al.* [54] were the first to numerically explore the entropy mode in the regime $k\rho_i \sim 1$, for which it is most unstable. They used the continuum code GS2 [24, 40] to solve the nonlinear gyrokinetic equation for both ions and electrons. Fig. (6.4) shows the growth rates for the entropy mode as a function of $k_\perp \rho_i$ that they found analytically for $T_i = T_e$, $\nu = 0$, and with no temperature gradient in the system. We see that there are two regimes for the entropy mode. Smaller density gradients (larger values for L_n) lead to weakly unstable modes that are just growing at small values of $k_\perp \rho_i$. Large density gradients are unstable for a wide range of values of $k_\perp \rho_i$ values which includes values well above one.

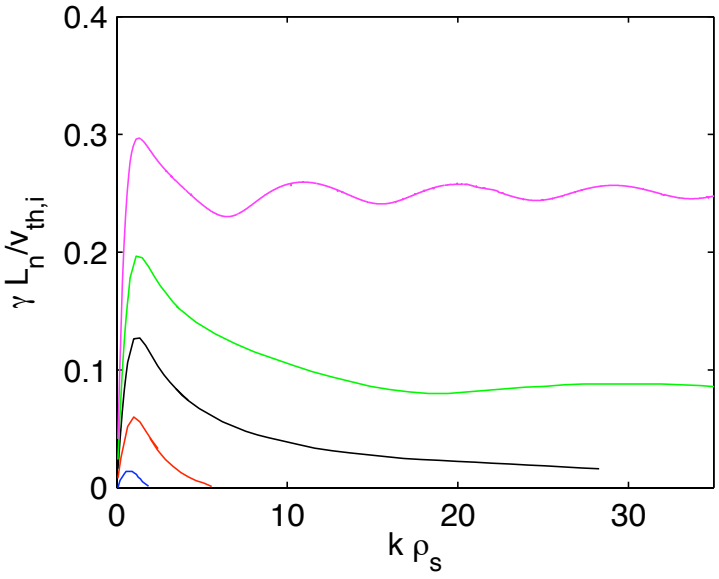

Figure 6.4: γ vs $k\rho_s$ for (bottom to top): $L_n/R_c = 1.25$ (blue), $L_n/R_c = 1(red), L_n/R_c = 0.8$ (black), $L_n/R_c = 0.67$ (green), and $L_n/R_c = 0.5$ (magenta). (reprinted from Ref. [54])

For the high $k\rho_i$ modes the new FLR algorithm that we introduced in GSP and described in Chapter 2 becomes important. We compare our novel scheme to evaluate the Bessel function J_0 explicitly in the code to the 4-point-averaging scheme. We find that in order to have GSP agree within a few percent with results from GS2 we need to use the new FLR algorithm.

We ran GSP with the 4-point averaging scheme and with the new FLR algorithm and compared this to results we obtained from GSP . We did this for two of the cases that are shown in Fig. (6.4). One with strong density gradients so

-	growth rate Case 1	growth rate Case 2
GS2	$\gamma = 0.46$	$\gamma = 0.055$
GSP FULL J_0	$\gamma = 0.465$	$\gamma = 0.05$
GSP 4-POINT-AVERAGING	$\gamma = 0.0$	$\gamma = 0.048$

Table 6.1: Growth rates for two different cases of the entropy mode. Comparison of results from GSP and the continuum code GS2. We show that in the case that high $k_\perp \rho_i$ modes are most unstable our novel FLR scheme is needed to have good agreement with the continuum code.

that the high $k_\perp \rho_i$ modes are unstable and one with weak density gradients, so that low $k_\perp \rho_i$ modes are most unstable. They two cases are:

- Case 1 : $R/L_n = 2,\ T_i = T_e,\ \nu = 0.0,$ calculated mode: $k_\perp \rho_i = 4$
- Case 2 : $R/L_n = 1,\ T_i = T_e,\ \nu = 0.0,$ calculated mode : $k_\perp \rho_i = 0.5$

The growth rates that we find for the different for GS2 and for GSP with and without the novel FLR algorithm are presented in Table 6.1. We see that for case 1 with strong gradients and $k_\perp \rho_i = 4$ the 4-point-averaging scheme fails to reproduce the instability while when we ran GSP with the new FLR algorithm we agreed with GS2 on the growth rate within 10%.

6.4 Nonlinear results

In the previous section we showed the importance of the new gyro-averaging scheme for finding the right linear growth rates for unstable modes with values of $k_\perp \rho_i > 1$. In this section we are going to investigate the importance of collisions on controlling the weight growth which is important to reduce the noise influence on the measurements of physical quantities in the code.

6.4.1 Weakly driven nonlinear entropy mode

We run GSP again for the settings of Case 2 that we described above. As we showed before we find with GSP the right answer for the linear growth rate. But this time we run the code nonlinearly and run it long enough to reach the fully non-linear regime. The box size was chosen to match Ricci, *et al.* In each of the perpendicular directions, 32 grid points were used. The time step was in the range of 0.05 to 0.15 R/v_t. We compare the same run with different number of particles. One run used 4 million particles per species and the other run 20 million. In Fig. (**??**) we show that both runs have a very similar behavior for the exponential growth during the linear growth period of the run. The sum of the squared weights grows at the same rate during that phase for both runs. But the results start to differ a lot when the runs reach the nonlinear phase. For the run with 4 million particles we see that the weight growth becomes catastrophic in the nonlinear phase. We are therefore unable to resolve this run with 4 million particles and the results from this run cannot be trusted. It is likely that this problem represents an interaction

between too few particles and too large of a time step.

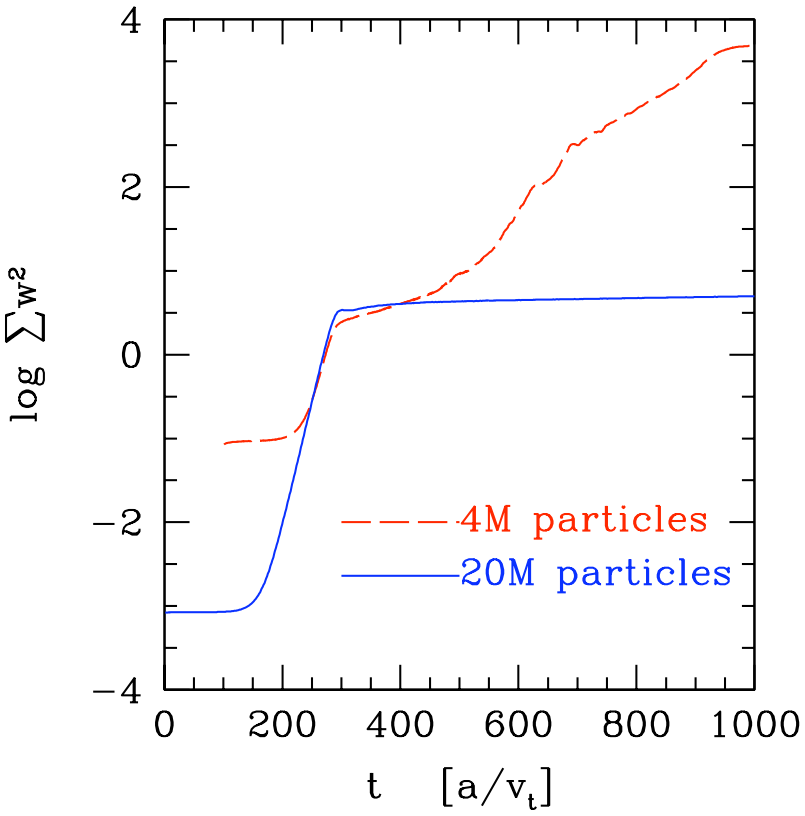

Figure 6.5: Comparison for weight growth for two identical nonlinear weakly driven entropy mode runs with different numbers of particles.

The weight growth for the run with 20 million particles behaves radically different than for the run with 4 million particles when the run enters the nonlinear phase. The squared weights stop growing exponentially and instead we observe a slow algebraic rise. The value of the squared particle weights is reduced by several orders of magnitude at late times of the simulation for the run with 20 million

particles compared to the run with 4 million particles. To understand the origin of this huge difference we look at the particle fluxes for for the two runs.

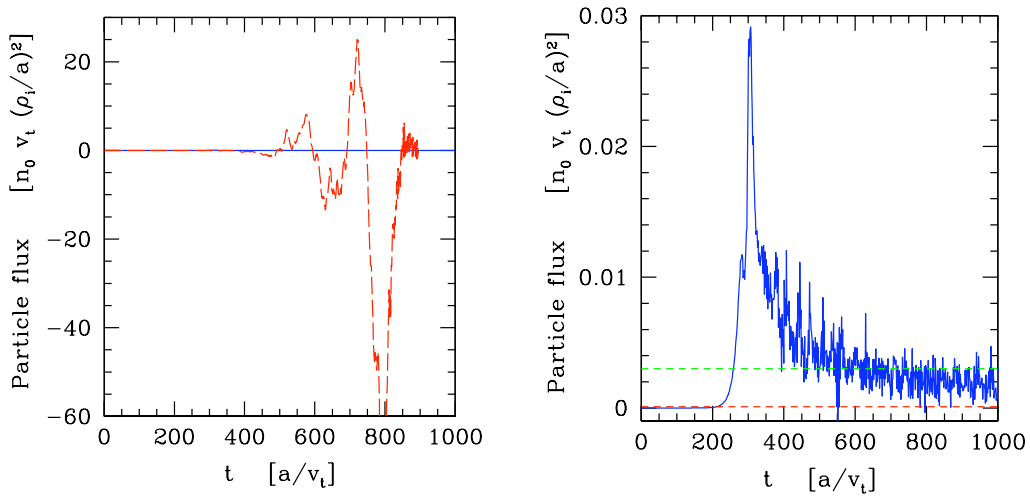

Figure 6.6: Particle flux versus time for two nonlinear entropy mode calculations with GSP with different particle numbers. On the left the particle flux for the run that uses 4 million particles shows a catastrophic behavior when the run becomes fully nonlinear. On the right for the run that uses 20 million particles the particle flux reaches a steady state that is compared to results from GS2. The dashed horizontal lines show the amount of particle flux Ricci *et al.* found using GS2 with (green line) and without collisions (red line).

The particle flux for the case that uses 4 million particles shows the catastrophic behavior that we already observed for the particle weights. The results for this run are shown on the left of Fig. (6.6). The flux is 3 orders of magnitude larger than in the case that uses more particles and doesn't reach a steady state. On the

right hand side of Fig. (6.6) we show the particle flux versus time for the run that uses 20 million particles per species and compare the results with the results from Ricci *et al.* [54]. The level of particle flux that we obtain with GSP when using a number of particles that is sufficient is for the collisional case in good agreement with the answers that Ricci *et al.* found using a continuum code.

We conclude that it is possible in some cases to get reasonable behavior if one uses enough particles, even without employing a collision operator. In the present case, the turbulent flux is very small, so that the growth of the weights is slow.

6.4.2 Strongly driven nonlinear entropy mode

In the last section we argued that by increasing the number of particles in the simulation we could overcome resolution issues and find answers for the particle fluxes that were in agreement with results found using continuum codes. But increasing the number of particles in the simulation is neither always a suitable solution nor does it overcome velocity space filamentation at late times of the simulation. While increasing the number of particles reduces the initial level of noise in the simulation a linear growth in the sum of the squared particle weights during the nonlinear phase will reduce the accuracy and eventually also a run that uses an enormous amount of particles will no longer we resolved.

Here we are comparing two runs with different number of particles. One run uses 1.6 billion particles per species and models a strongly driven entropy mode without collisions while the other run uses 3 million particles per species and has

the pitch-angle collision term turned on. In Fig. (6.7) we compare the behavior of the sum of the squared weights over time for both runs on a log and linear scale. The remarkable finding is that though the collisionless run uses ~ 500 times more particles than the run with pitch-angle collisions it has a stronger weight growth in the nonlinear part of the simulation and ends up having a larger overall value for the sum of the squared weights. So the numerical expense of adding all that particles for the collisionless run was not necessary. With the pitch-angle collision operator the weight growth can be reduced within a calculation that is numerically a lot less expensive.

In Fig. (6.8) we investigate the particle fluxes for those two runs we see that they are initially fairly similar. On the left hand side of Fig. (6.8) we compare the particle fluxes of the collisionless and collisional run. We see that vary on the same time scale around a constant level. For late times the collisionless run the amplitude for the variation of the particle flux keeps increasing and it gets harder to estimate a value for the heat flux. We do not observe the same increase for the variation of the particle flux for the collisionless run. On the right hand side we compare the particle flux of the collisionless run to the particle flux that was found for the identical system using Gs2 . The level around which the particle flux of the collisional run fluctuates is comparable to the finding with Gs2 (red horizontal line on RHS of Fig. (6.8). For future work it would be interesting to study the strong time variation of the particle flux that we are observing in Gsp for this run.

Looking at the weight growth for the run that uses the pitch-angle collision operator we see that though the weight growth is reduced by a substantial amount,

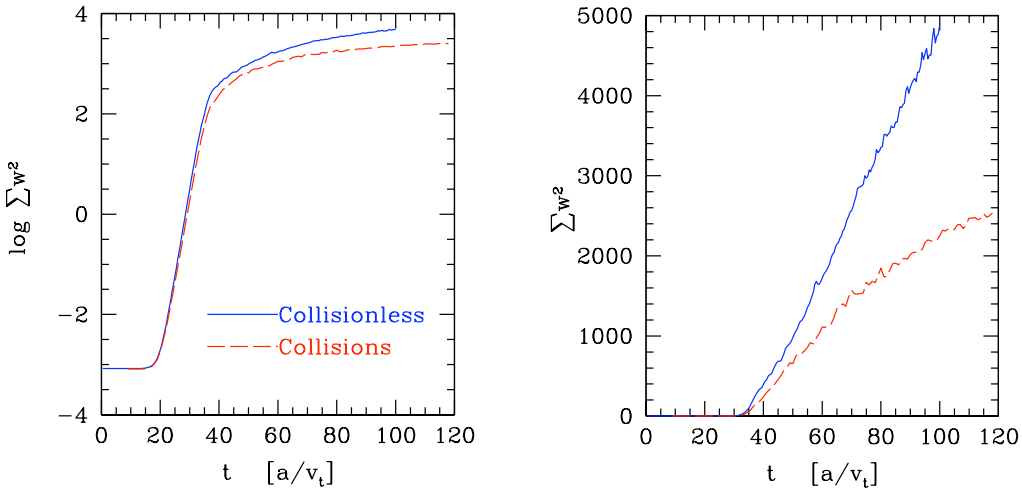

Figure 6.7: Sum of squared weights versus time for two nonlinear entropy mode calculations with GSP. The blue curve shows the results for a run that uses 1.6 billion particles per species and is collisionless while the red curve is for the identical run with particle number reduced to 3 million per species and pitch-angle collisions added. The weight growth for the run that includes pitch-angle collisions is in the nonlinear phase of the simulation a lot slower compared to the collisionless simulation although the coliisionless simulation uses $\sim$ 500 times more particles.

it does not stop entirely. To understand why the weights keep on growing when we include the pitch-angle collisions we visualize the weight distribution in velocity space for this run. In Fig. (6.9) we show snapshots of the weight distribution at different instances late in the calculation well within the nonlinear phase.

We observe that the main dependence of the weight distribution is in energy and not in pitch-angle. That is expected since we showed before in Fig. (5.3) that

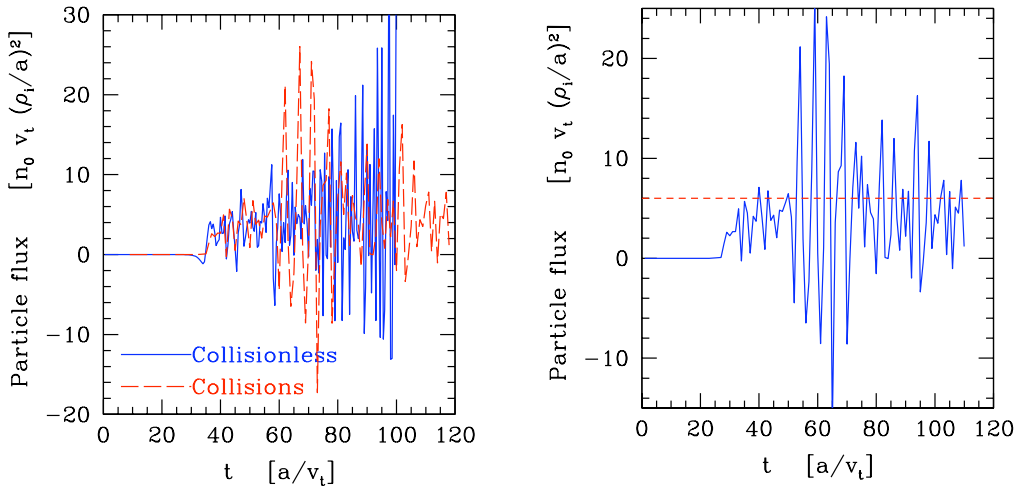

Figure 6.8: Particle fluxes for the two nonlinear entropy mode calculations from Fig. (6.7). On the RHS we compare the results from the run that includes collisions to collisional results from GS2 (red horizontal line).

the pitch-angle collision operator in GSP smooths structures that are pitch-angle dependent efficiently. For future work we think it is important to include an energy-diffusion term to the collision operator to also smooth structures in velocity space that are energy dependent.

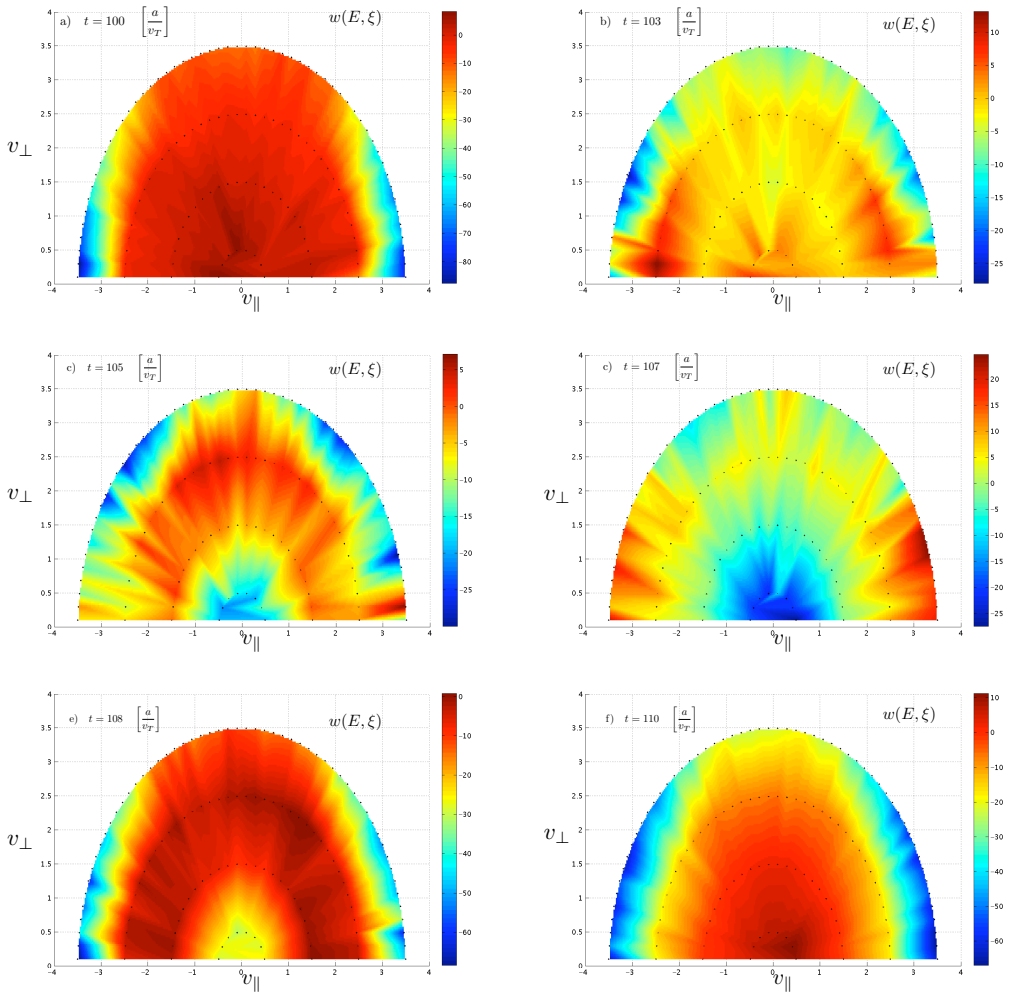

Figure 6.9: Time-series that shows the velocity-space dependence of an entropy mode simulation that uses the pitch-angle scattering operator. At late times in the calculation the structure in velocity space is predominant energy dependent.

Chapter 7

Conclusions

For the research that was conducted as part of this thesis we developed a new particle in cell code that solves the electrostatic version of the gyrokinetic equation. The code has the option of including several particle species and can solve a slab or Z-pinch configuration.

The goal for the new code was to address two challenges that standard particle codes have to deal with:

- Spatial resolution at scales $k_\perp \rho_i > 1$ that cannot be resolved by the 4-point averaging scheme for evaluating gyro-averages.

- Resolution at late times in the nonlinear phase of the simulation that is challenged by growth of the variance of the weights.

As we showed in this thesis we successful addressed those two points in GSP . With the novel algorithm to solve the Bessel function $J_0(k_\perp \rho_i)$ explicitly in this code we were able to obtain the right linear growth rates for the entropy mode, even for $k_{perp}\rho_i = 4$. The 4-point averaging scheme failed to find any linear growth for this case. For the nonlinear regime of the entropy mode we showed the importance of the

collision operator to limit the growth of the weights. The major numerical expense for a particle code to be able to incorporate a collision operator is that the number of simulation particles must be sufficient to resolve velocity space initially so that we can operate with the collision operator on the perturbed distribution function. But once the amount of particles is made large enough to resolve velocity space the collision operator will limit the amount of weight growth during the nonlinear phase of the simulation and therefore the accuracy of the calculation won't be challenged at late times of the simulation. So while the collision operator requires upfront more particles and therefore makes the simulation numerically more expensive it will solve the problem of resolution at late times. This would have been addressed in a conventional code by adding more particles to the simulation and therefore guaranteeing that the code will not be dominated by noise before the code reaches a steady state.

Our results show that if one decides to resolve velocity space adequately with Lagrangian PIC algorithms, the expense as a function of simulated spatial domain size at fixed resolution rises considerably. Nevertheless, with the higher resolution that comes with this decision, it is clear that the growing weight problem can be directly solved. It is also possible to describe fluctuations with wavelengths considerably shorter than the thermal gyroradius with this improved velocity-space resolution.

For future work, it is important to perform more benchmarks, with both Eulerian and Lagrangian codes. It would be useful to determine the optimal number of velocity-space bins for a range of turbulent problems, and also to determine the

number of particles per 5-D bin required to resolve $h(x, y, z, E, \xi)$ adequately. With these findings in hand, it would be possible to compare the efficiency of Eulerian and Lagrangian schemes at fixed accuracy.

It is clear from the results presented in this thesis that while the pitch-angle scattering operator provides physically-motivated smoothing of the distribution function as a function of pitch angle, there is structure formation in $h(E)$ which also tends to progress to the smallest resolved scales in energy. Extension of the basic ideas presented here to include energy diffusion in the collision operator is desirable.

Finally, it would be useful to determine whether the efficiency of the algorithms presented here remains high with more physics included, such as magnetic shear, magnetic fluctuations, and magnetic trapping. GSP moves 640 million guiding centers 1 full time step in 1 wall-clock second on 4096 Cray XT-4 cores (on the jaguar supercomputer at ORNL) currently, but it may well be the case that the performance is degraded as the physics model is broadened. Magnetic trapping, for example, leads to time evolution of $v_\parallel$ and $v_\perp$, which could affect performance.

Appendix A

Plasma Parameters

TABLE 2

Definition of parameters

Plasma Parameter	Definition
$s(= e, i)$	Species (=electron,ion)
q_s	Particle charge
n_{0_s}	Number density
T_s	Temperature
m_s	Particle mass
B_0	Background magnetic field strength
$v_{T_s} \equiv \sqrt{\frac{T_s}{m_s}}$	Thermal velocity
$\lambda_{D_s} \equiv \sqrt{\frac{T_s}{4\pi n_{0_s} q_s}}$	Electron Debye Length
$\Omega_{0_s} \equiv \frac{q_s B_0}{m_s c}$	Equilibrium cyclotron frequency
$\rho_s \equiv \frac{v_{T_s}}{\Omega_s}$	Larmor radius

Appendix B

Normalization

TABLE 3

Definition of normalized quantities in GSP

$q_{s_N} = \frac{Z_s\|e\|}{Z_{ref}\|e\|} = \frac{Z_s}{Z_{ref}}$	Particle charge
$n_{0_{s_N}} = \frac{n_{0_s}}{n_{0_{ref}}}$	Number density
$T_{s_N} = \frac{T_s}{T_{ref}}$	Temperature
$m_{s_N} = \frac{m_s}{m_{ref}}$	Particle mass
$t_N = t\frac{v_{T_{ref}}}{a}$	time
$\phi_N = \frac{Z_{ref}\|e\|a}{T_{ref}\rho_{ref}}\phi$	electrostatic potential
$x_N = \frac{x}{\rho_{ref}}$; $y_N = \frac{y}{\rho_{ref}}$	perpendicular direction
$z_N = \frac{z}{a}$	parallel direction
$R_{c_N} = \frac{R_c}{a}$	Curvature
$L_{T_{s_N}} = \frac{L_{T_s}}{a}$; $L_{n_{s_N}} = \frac{L_{n_s}}{a}$	gradient scale length
$w_{s_N} = w_s\frac{a}{\rho_{ref}}$	particle weight
$\delta f_{s_N} = \frac{a}{n_{0_{ref}}\rho_{ref}}\delta f_s$	perturbed distribution function
$\nu_{s_N} \equiv \frac{A\nu_s}{v_{T_{ref}}}$	collisionality

Appendix C

Input file for GSP

2-dimensional Z-pinch for strongly driven entropy mode ($L_n = 0.5$).

```
&grid_par

Ncell = 500

Nx=32

Ny=32

Nz=1

Nvperp=31

Lx=123.66

Ly=123.66

Lz=6.28

dvperp = 0.125

delta_t = 0.05

nstep= 2000

/

&dia_par

diagnostics_on = F

phi_diagnostics_on = F
```

```
energy_diagnostics_on = F

spectrum_diagnostics_on = T

spectrum_write = 100

/

&dia_new_par

diagnostics_new_on = T

diagnostics_1_on = F

diagnostics_2_on = F

diagnostics_3_on = T

diagnostics_4_on = T

nwrite = 10

/

&init_par

factor_delta = 0.1

num_species = 2

Rinv = 1.

temperature_1 = 1.

temperature_2 = 1.

mass_1 = 1.

mass_2 = 5.4e-4

density_1 = 1.

density_2 = 1.

charge_1 = 1.
```

```
charge_2 = -1.

Ln_1 = 0.5

Ln_2 = 0.5

LT_1 = 1.e8

LT_2 = 1.e8

nonlinear_on = T

time_growth_off = 10000.

nonlinear_phase = 1809000.

save_for_restart = T

init_restart = F

curvature_on = T

/

&dia_traj

diagnostics_traj_on =.FALSE.

Ndisp = 1000

Nvbin = 50

traj_dia_step = 50

/

&collision_par

dE = 0.5

N_Chi = 4

collision_on = F

gamma = 0.0
```

```
time_collision_on = 0.

collision_time = 1

pitch_angle_collision_on = F

nu_coll_1 = 0.

nu_coll_2 = 0.

/

&vel_dia_par

v_dia_x = 4

v_dia_y = 4

v_dia_z = 4

velocity_space_diagnostics_time = 10

velocity_space_diagnostics_on = F

/
```

Appendix D

Parallelization of GSP

Gsp is designed to run efficiently on large parallel clusters. Our parallelization scheme is optimized for large number of particles relative to the three-dimensional spatial grid. Therefore each processor runs a copy of the code with the entire grid using an equal fraction of the total numbers of particles. Before we conduct operations on quantities that are defined on the grid we must reduce those quantities so that the information from all processors and therefore from all simulation particles contributes.

In Fig. (D.1)-(D.2) we show tests for strong and weak scaling of Gsp . For weak scaling we increase the problem size at the same rate as the number of processors. Perfect weak scaling would therefore mean that the runtime stays fixed for all runs. We see in Fig. (D.1) that Gsp looses about 5 % on performance compared to perfect weak scaling.

To test strong scaling we increase the number of processors while keeping the problem size fixed. For perfect strong scaling the runtime multiplied by the number of processors and normalized to the runtime for one processor is a constant. For Gsp we are off by 8% compared to strong scaling when we go from one to 32 processors.

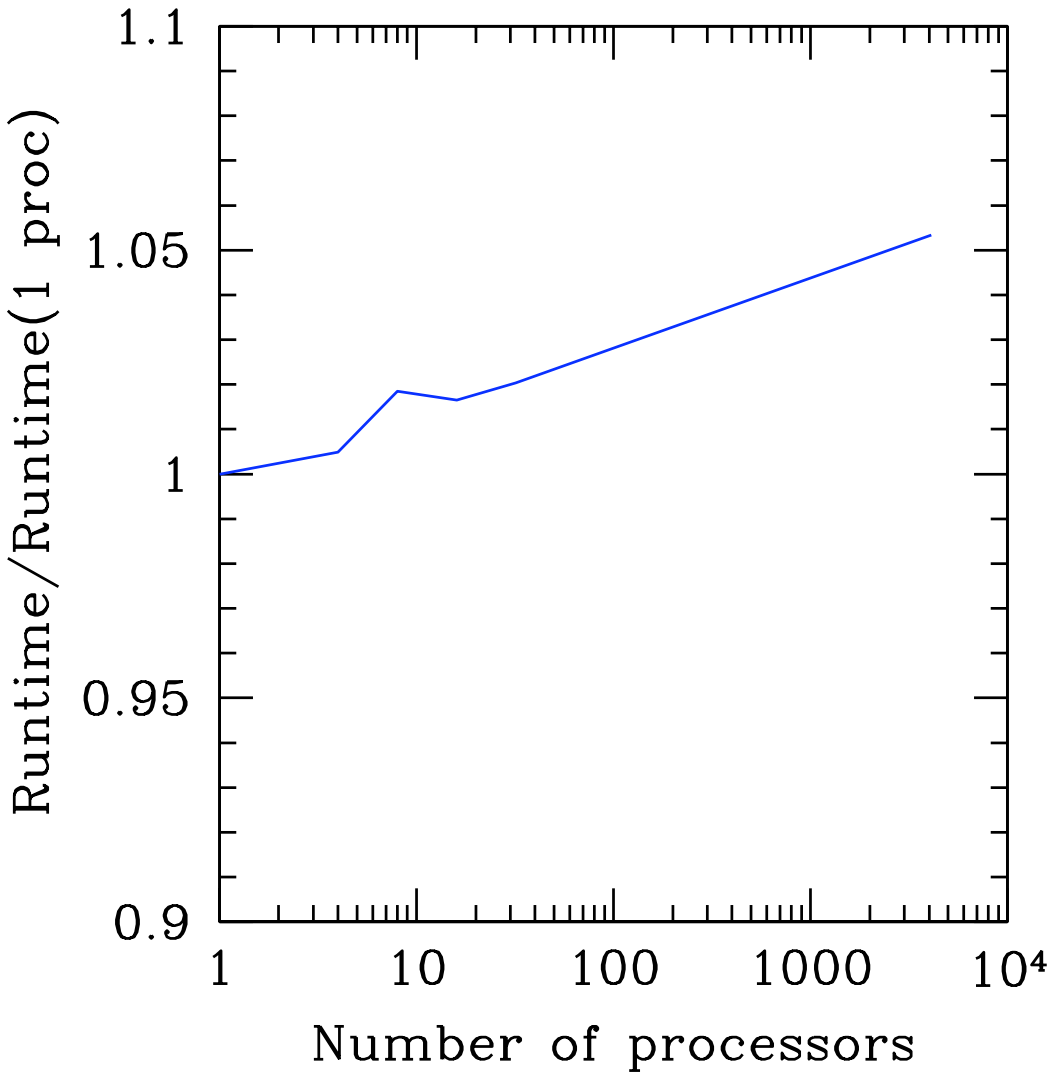

Figure D.1: Weak scaling for GSP. Going from one to 4096 processors we loose $\sim 5\%$ on performance

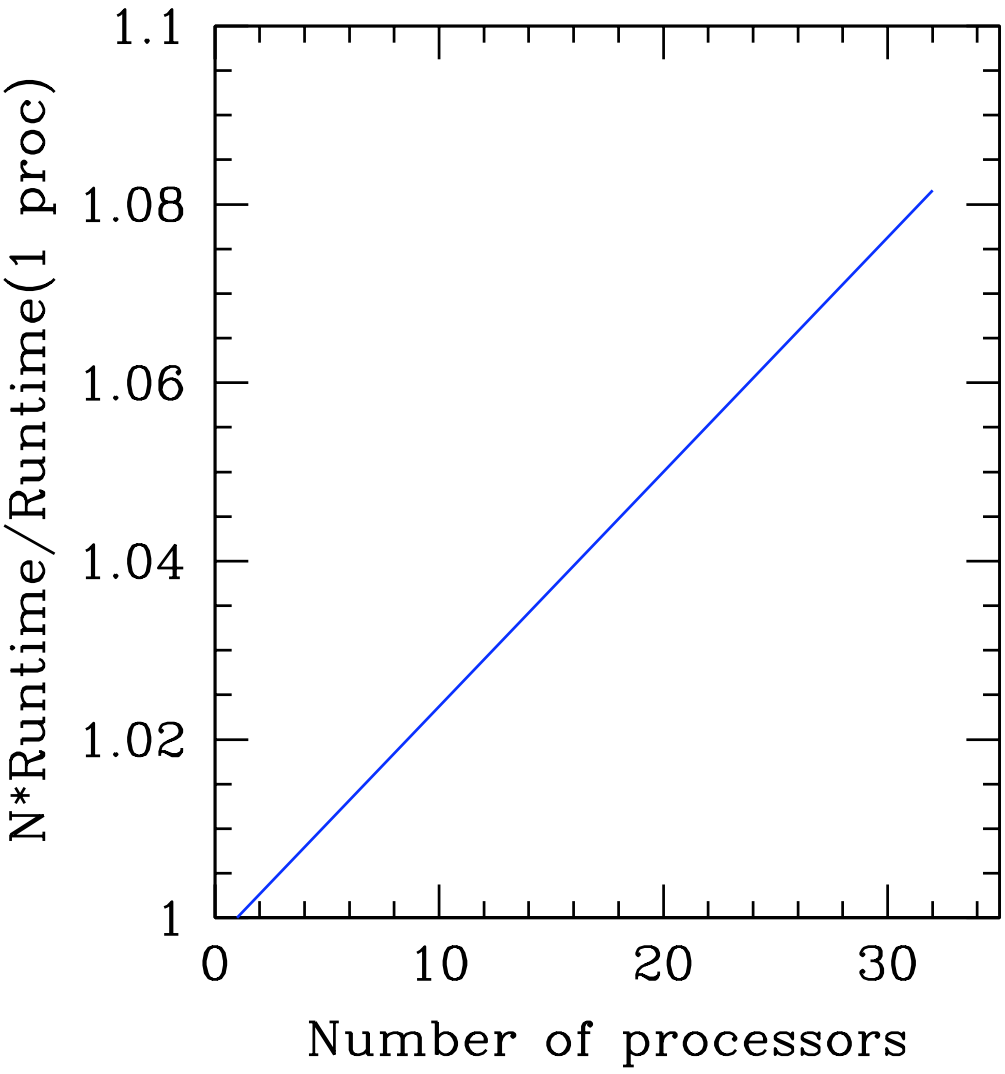

Figure D.2: Strong scaling for GSP. Going from one to 32 processors we loose $\sim 8\%$ on performance compared to perfect strong scaling.

Bibliography

[1] T. M. Antonsen and B. Lane, "Kinetic equations for low frequency instabilities in inhomogeneous plasmas", Phys. Fluids **23**(6) (1980).

[2] A. Y. Aydemir, "A unified Monte Carlo interpretation of particle simulations and applications to non-neutral plasmas" Phys. Plasmas **1**, 822 (1994)

[3] M. A. Barnes and W. Dorland, "Velocity space dynamics in gyrokinetics", Phys. Plasma, (submitted 2008)

[4] M. A. Barnes, W. Dorland, T. Tatsuno, I. G. Abel and A. A. Schekochihin, "Linearised Model Fokker-Planck Collision Operators for Gyrokinetic Simulations: Numerical Implementation", Phys. Plasma, (submitted 2008)

[5] M. A. Beer, S. C. Cowley and G. W. Hammett, "Field-aligned coordinates for nonlinear simulations of tokomak turbulence", Phys. Plasma **2** (7), 2687 (1995)

[6] C. K. Birdsall and A. B. Langdon, "Plasma Physics via Computer Simulation", McGraw-Hill Book Company (1985)

[7] C. K. Birdsall, "Particle-in-Cell Charged Particle Simulations, plus Monte Carlo Collisions with Neutral Atoms, PIC-MCC", IEEE Transactions on Plasma Sciences, (19), 65 (1991).

[8] A. Bottino, A. G. Peeters, R. Hatzky, S. Jolliet, B. F. McMillan, T. M. Tran, and L. Villard, "Nonlinear low noise particle-in-cell simulations of electron temperature gradient driven turbulence", Phys. Plasma **14**, 010701 (2007)

[9] S. Brunner, E. Valeo, J. A. Krommes, "Collisional delta-f scheme with evolving background for transport time-scale simulations", Phys. Plasma **6**, 4504 (1999)

[10] O. Buneman, "Dissipation of current in ionized media", Phys. Rev., (115), 503 (1959)

[11] J. Candy and R. E. Waltz, "An Eulerian gyrokinetic Maxwell solver", J. Comp. Phys. **186**, 545 (2003)

[12] J. Candy and R. E. Waltz, " Velocity-space resolution, entropy production, and upwind dissipation in Eulerian gyrokinetic simulations", Phys Plasma **13**, 032310 (2006)

[13] P. J. Catto and K. T. Tsang, "Linearized gyro-kinetic equation with collisions", Phys. Fluids **20**, 396 (1977)

[14] Y. Chen and S. E. Parker, "Coarse-graining phase space in δf particle-in-cell simulations", Phys. Plasma **14**, 082301 (2007)

[15] Y. Chen and S. E. Parker, "Electromagnetic gyrokinetic δf particle-in-cell turbulence simulation with realistic equilibrium profiles and geometry", J. Comp. Phys. **220**, 839 (2007)

[16] Y. Chen and R. B. White, "Collisional δf method", Phys. Plasma **4** (10), 3591 (1997)

[17] S. C. Cowley, R. M. Kulsrud, and R. Sudan, "Considerations of ion-temperature-gradient-driven turbulence", Phys. Fluids B **3** (10) 2767 (1991)

[18] J. M. Dawson, "Plasma oscillations of a large number of electron beams", Phys. Rev., (118), 381 (1960)

[19] R. E. Denton and M. Kotschenreuther, "δf" Algorithm", J. Comput. Phys. **119**. 283 (1995)

[20] A. M. Dimits and B. I. Cohen, "Collision operator for partially linearized particle simulation codes", Phys. Rev. E **49**, 709 (1994)

[21] A. M. Dimits and W. W. Lee, "Partially linearized algorithms in gyrokinetic particle simulation", J. Comput. Phys. **107**,2 309 (1993)

[22] A. M. Dimits, T. J. Williams, J. A. Byers and B. I. Cohen, "Scalings of Ion-Temperature-Gradient-Driven anomalous transport in Tokamaks" Phys. Rev. Lett. **77**, 71 (1996)

[23] A. M. Dimits, G. Bateman, M. A. Beer, B. I. Cohen, W. Dorland, G. W. Hammett, C. Kim, J. E. Kinsey, M. Kotschenreuther, A. H. Kritz, L. L. Lao, J. Mandrekas, W. M. Nevins, S. E. Parker, A. J. Redd, D. E. Shumaker, R. Sydora, and J. Weiland, "Comparisons and physics basis of tokamak transport models and turbulence simulations", Phys. Plasma **7**, 969 (2000)

[24] W. Dorland and F. Jenko, M. Kotschenreuther, B. N. Rogers, "Electron temperature gradient turbulence", Phys. Rev. Lett. **85**, 5579 (2000)

[25] J. F. Drake, P. N. Guzdar, and A. B. Hassam, "Streamer Formation in Plasma with a Temperature Gradient", Phys. Rev. Lett. **61**, 2205 (1988)

[26] E. A. Frieman and L. Chen, "nonlinear gyrokinetic equations for low-frequency electromagnetic waves in general plasma equilibria", Phys. Fluids **25**, 502 (1982)

[27] S. K. Godunov, "A Difference Scheme for Numerical Solution of Discontinuous Solution of Hydrodynamic Equations", Math. Sbornik **47**, 271 (1959)

[28] P. N. Guzdar, L. Chen, W. M. Tang, and P. H. Rutherford, "Ion-Temperature-Gradient Instability in Toroidal Plasmas", Phys. Fluids **26**, 673 (1983)

[29] A. B. Hassam and Y. C. Lee, "Drift-ideal magnetohydrodynamics", Phys. Fluids **27**, 438 (1984)

[30] P. Helander, and D. J. Sigmar, "Collisional Transport in Magnetized Plasmas" (Cambridge: Cambridge Univ. Press), (2002)

[31] F. L. Hinton, "Simulating Coulomb collisions in a magnetized plasma", Phys. Plasma **15**, 042501 (2008)

[32] S. P. Hirshman and D. J. Sigmar, "Approximate FokkerPlanck collision operator for transport theory applications", Phys. Fluids **19**, 1532 (1976)

[33] R. W. Hockney and J. W. Eastwood, "Computer simulation using particles", McGraw-Hill Book Company (1981)

[34] G. G. Howes, S. C. Cowley, W. Dorland, G. W. Hammett, E. Quataert and A. A. Schekochihin, " Astrophysical gyrokinetics: Basic equations and linear theory", ApJ, 651:590 (2006).

[35] G. G. Howes, T. Tatsuno, W. Dorland, " AstroGK : Astrophysical Gyrokinetic Code, J. Comp. Phys., in preparation (2008)

[36] G. Hu, J. A. Kromes, "Generalized weighting scheme for δf particle-simulation method", Phys. Plasma **4**, 863 (1994)

[37] F. Jenko, "Massively parallel Vlasov simulation of electromagnetic drift-wave turbulence", Comp. Phys. Comm. **125**, 196 (2000)

[38] J. Kesner, "Interchange modes in a collisional plasma", Phys. Plasma **7**, 3837 (2000)

[39] J. Kesner, and R. J. Hastie, "Electrostatic drift modes in a closed filed line configuration" , Phys. Plasma **9**, 395 (2002)

[40] M. Kotschenreuther, G. Rewoldt, W. M. Tang, "Comparison of initial value and eigenvalue codes for kinetic toroidal plasma instabilities", Compt. Phys. Comm. **88**, 128 (1995)

[41] J. A. Krommes, "Thermostated δf, Phys. Plasma **6** (5), 1477 (1999)

[42] J. A. Krommes, G. Hu, "The role of dissipation in the theory and simulations of homogenous plasma turbulence, and resolution of the entropy paradox", Phys. Plasma **1**, 3211 (1994)

[43] L. D. Landau, "Die kinetische Gleichung für den Fall Coulombacher Wechselwirkung", Phis. Z. Sowietunion **10**, 154 (1936)

[44] W. W. Lee, "Gyrokinetic approach in particle simulation", Phys. Fluids **26** (2), 556 (1983)

[45] W. W. Lee, W. M. Tang, "Gyrokinetic particle simultion of ion temperature gradient drift instabilities", Phys. Fluids **31**, 612 (1988)

[46] Z. Lin, T. S. Hahm, W. W. Lee, W. M. Tang, R. B. White, "Turbulent Transport Reduction by Zonal Flows: Massively Parallel Simulations", Science **281**, 1835

[47] B. F. McMillan, S. Joilliet, T. M. Tran, L. Villard, A. Bottino, P. Angelino "Long global gyrokinetic simulations: Source terms and particle noise control", Phys. Plasmas **15**, 052308 (2008)

[48] W. M. Nevins, G. W. Hammett, A. M. Dimits, W. Dorland, D. E. Shumaker, "Discrete particle noise in particle-in-cell simulations of plasma microturbulence", Phys. Plasma **12**, 122305 (2005).

[49] D. R. Nicholson, "Introduction to Plasma Theory", Wiley, New York, 1983

[50] S. E. Parker, W. Dorland, R. A. Santoro, M. A. Beer, Q. P. Liu, W. W. Lee, G. W. Hammett, "Comparison of Gyrofluid and Gyrokinetic Simulations", Phys. Plasma **1**, 1462 (1994)

[51] S. E. Parker, and W. W. Lee, "A fully nonlinear characteristic method for gyrokinetic simulation", Phys. Fluids B **5** (1), 77 (1993)

[52] S. E. Parker, W. W. Lee, and R. A. Santoro, "Gyrokinetic Simulation of Ion Temperature Gradient Driven Turbulence in 3D Toroidal Geometry", Phys. Rev. Lett. **71**, 2042 (1993)

[53] P. Ricci, B. N. Rogers, W. Dorland, and M. A. Barnes, "Gyrokinetic linear theory of the entropy mode in a Z pinch", Phys. Plasma **13**, 062102 (2006)

[54] P. Ricci, B. N. Rogers, and W. Dorland, "Small-Scale Turbulence in a closed-Field-Line Geometry", Phys. Rev. Lett. **97**, 245001 (2006)

[55] M. N. Rosenbluth, R. D. Hazeltine, and F. L. Hinton, "Plasma Transport in Toroidal Confinement Systems", Phys. Fluids **15**, 116 (1972)

[56] P. Rutherford, L. Kovrizhnikh, M. Rosenbluth, and F. Hinton, "Effect of Longitudinal Electric Field on Toroidal Diffusion", Phys. Rev Lett. **25**, 1090 (1970)

[57] S. A. Sarra, "The Method of Characteristics with applications to Conservation Laws", JOMA **3**, (2003)

[58] A. A. Schekochihin, S. C. Cowley, W. Dorland, G. W. Hammett, G. G. Howes, E. Quataert, T. Tatsuno, "Astrophysical gyrokinetics: Kinetic and fluid turbulent ccascades in magnetized weakly collisional plasmas", Astrophys. J. Suppl., submitted (2007)

[59] A. N. Simakov, P. J. Catto, and R. J. Hastie, "Kinetic stability of electrostatic plasma modes in a dipolar magnetic field", Phys. Plasma **8**, 4414 (2001)

[60] A. N. Simakov, R. J. Hastie, and P. J. Catto, "Long mean-free path collisional stability of electromagnetic modes in axisymmetric closed magnetic field configurations", Phys. Plasma **9**, 201 (2002)