Princeton Plasma Physics Laboratory

PPPL-5253

A fully non-linear multi-species Fokker-Planck-Landau collision operator for simulation of fusion plasma

Robert Hager, E.S. Yoon, S. Ku, E.F. D'Azevedo, P.H. Worley, C.S. Chang

June 2016

Prepared for the U.S. Department of Energy under Contract DE-AC02-09CH11466.

Princeton Plasma Physics Laboratory
Report Disclaimers

Full Legal Disclaimer

Trademark Disclaimer

PPPL Report Availability

Princeton Plasma Physics Laboratory:

http://www.pppl.gov/techreports.cfm

Office of Scientific and Technical Information (OSTI):

http://www.osti.gov/scitech/

Related Links:

U.S. Department of Energy

U.S. Department of Energy Office of Science

U.S. Department of Energy Office of Fusion Energy Sciences

A fully non-linear multi-species Fokker-Planck-Landau collision operator for simulation of fusion plasma

Robert Hager[a,*], E. S. Yoon[b], S. Ku[a], E. F. D'Azevedo[c], P. H. Worley[c], C. S. Chang[a]

[a] *Princeton Plasma Physics Laboratory, P.O. Box 451, Princeton, NJ 08543, USA*
[b] *Rensselaer Polytechnic Institute, 110 8th Street, Troy, NY USA 12180, USA*
[c] *Oak Ridge National Laboratory, P.O. Box 2008, Oak Ridge, TN 37831, USA*

Abstract

Fusion edge plasmas can be far from thermal equilibrium and require the use of a non-linear collision operator for accurate numerical simulations. In this article, the non-linear single-species Fokker-Planck-Landau collision operator developed by Yoon and Chang [Phys. Plasmas **21**, 032503 (2014)] is generalized to include multiple particle species. The finite volume discretization used in this work naturally yields exact conservation of mass, momentum, and energy. The implementation of this new non-linear Fokker-Planck-Landau operator in the gyrokinetic particle-in-cell codes XGC1 and XGCa is described and results of a verification study are discussed. Finally, the numerical techniques that make our non-linear collision operator viable on high-performance computing systems are described, including specialized load balancing algorithms and nested OpenMP parallelization. The collision operator's good weak and strong scaling behavior are shown.

Keywords: plasma, fusion, collision operator, XGC, particle-in-cell
PACS: 52.20.Fs, 52.65.Tt, 52.65.Rr

1. Introduction

Coulomb collisions are an essential part of the physics needed to describe the plasmas that commonly occur in nuclear fusion experiments. This is especially true in the low-temperature, highly collisional edge region of tokamak plasmas. Most kinetic codes for the simulation of fusion plasma employ a linearized version of the Fokker-Planck collision operator. The linearization is based on the assumption that the plasma distribution functions can be represented by the sum of a Maxwellian distribution function and a small perturbation, i.e.

$$f = f_M + \delta f, \quad \frac{\delta f}{f_M} \ll 1. \tag{1}$$

Examples of such linearized operators are the collision operators by Hirshman and Sigmar [1], Boozer and Kuo-Petravic [2], Wang et al. [3], and Belli et al. [4].

Linearization of the collision operator is well justified in the core of tokamak plasmas, where radial mixing and transport are small because the ion orbit width and turbulence scales are much smaller than the gradient scale lengths of the background density and temperatures. This parameter regime is often called the *local regime*. However, in the edge region of tokamak plasmas, especially in H-mode discharges with their narrow density and temperature pedestals, physics become non-local as the ion orbit width and the scales of turbulence become comparable to the background scale

*Corresponding author

Email addresses: rhager@pppl.gov (Robert Hager), yoone@rpi.edu (E. S. Yoon), sku@pppl.gov (S. Ku), dazevedoef@ornl.gov (E. F. D'Azevedo), worleyph@ornl.gov (P. H. Worley), cschang@pppl.gov (C. S. Chang)

lengths of the plasma. In the scrape-off layer and around the magnetic separatrix surface, there are neutral ionization and charge-exchange processes, particle loss to the material wall, and radiative energy loss. In this situation, the perturbation δf can be comparable to the background f_M, and the assumptions for linearizing the Fokker-Planck operator are invalid. The Coulomb collision rate is large and becomes one of the dominant physics phenomena. Therefore, a non-linear Fokker-Planck operator is required for accurate results in the tokamak edge.

There are both physical constraints and practical requirements for the development of numerical methods when implementing nonlinear collision operators. Regarding physics, collision operators must conserve mass separately for each species as well as total momentum and energy. In addition, the entropy should increase, and the probability distribution functions of all particle species should relax toward a Maxwellian when approaching thermal equilibrium in a closed system. Furthermore, by definition, the probability distribution function should always be positive. To limit the cost of numerical simulations, the time discretization of the Fokker-Planck equation needs to be implicit in order to circumvent the Courant condition of collisional diffusion in velocity space. Recently, remarkable progress in the development of numerical methods that satisfy all of the above constraints and requirements has been reported. For more details, we refer the reader to articles by Buet and Le Thanh [5, 6] and references therein. Other important contributions have been made by Taitano et al. [7], and Pataki and Greengard [8].

Recently, Yoon and Chang [9] developed a non-linear Fokker-Planck-Landau (FPL) collision operator in the 2-dimensional guiding-center ve-

locity space for single-species, strongly magnetized plasma, which was implemented in the 5-dimensional, full-function gyrokinetic PIC code XGC1 [10]. The purpose of this paper is to extend the work by Yoon and Chang [9] to multiple particle species, and to apply it to the gyrokinetic particle-in-cell (PIC) codes XGC1 and XGCa. This is necessary to capture the non-adiabatic behavior of electrons, especially in the plasma edge. Furthermore, considering that the concentrations of impurities are much higher in the edge region than in the core region, it is imperative to treat non-linear collisions among multiple particle species (i.e. more than two) accurately for more realistic simulations of the plasma edge. As in Ref. [9], finite gyro-radius effects are not considered in the collision operator.

The remainder of this article is organized as follows. In Sec. 2, we derive the discretized multi-species Fokker-Planck-Landau operator and prove the important conservation laws for mass, momentum, and energy in the continuum and in discretized velocity space. We also explain the implementation of the new multi-species collision operator in the gyrokinetic hybrid total-δf PIC codes XGCa [11] and XGC1 [10]. The results of various verification studies to demonstrate the accuracy of this FPL operator are presented in Sec. 3. Since our implementation of the FPL operator is intended for the use in high-performance computing, we demonstrate the scalability of our approach with the code XGCa in Sec. 4. Our conclusions are presented in Sec. 5.

2. Multi-species Fokker-Planck-Landau operator and its implementation in particle-in-cell codes

Some of the material in this section is rather fundamental and well-known. Some of the material has already been explained in Ref. [9]. This basic material is included here for the sake of completeness and for the general audience.

2.1. The Fokker-Planck-Landau collision operator

The Fokker-Planck collision operator in Landau form (FPL) for multiple particle species is given by

$$\begin{aligned}\left.\frac{\mathrm{d}f_a}{\mathrm{d}t}\right|_{col} &= \sum_b C_{ab}(f_a, f_b') \\ &= -\sum_b \frac{e_a^2 e_b^2 \ln\Lambda_{ab}}{8\pi\epsilon_0^2 m_a} \nabla\cdot\left[\int \mathrm{d}^3v'\, \underline{\boldsymbol{U}}\cdot\left(\frac{f_a}{m_b}\nabla' f_b' - \frac{f_b'}{m_a}\nabla f_a\right)\right] \\ &= -\sum_b \nabla\cdot(\boldsymbol{E}_{ab} f_a + \underline{\boldsymbol{D}}_{ab}\cdot\nabla f_a) = -\sum_b \nabla\cdot\boldsymbol{J}_{ab}. \end{aligned} \tag{2}$$

Here, the subscripts a and b are species indices, $f_a = f_a(\boldsymbol{v})$ and f_b are distribution functions, $f' = f(\boldsymbol{v}')$, $\nabla = \partial/\partial\boldsymbol{v}$, and $\nabla' = \partial/\partial\boldsymbol{v}'$, $e_{a/b}$ are the particle charges, $m_{a,b}$ are the particle masses, $\ln\Lambda_{ab}$ is the Coulomb logarithm for collisions between species a and b, and ϵ_0 is the vacuum permittivity. The tensor $\underline{\boldsymbol{U}}$ is defined as

$$\underline{\boldsymbol{U}} = \frac{u^2\underline{\boldsymbol{I}} - \boldsymbol{u}\boldsymbol{u}}{u^3}, \tag{3}$$

where $\boldsymbol{u} = \boldsymbol{v} - \boldsymbol{v}'$ and $u = |\boldsymbol{u}|$. The right-hand side of Eq. (2) shows that the collision operator can be expressed in the form of a continuity equation. The corresponding flux $\boldsymbol{J}_{ab}$ in velocity space is driven by the drag and diffusion

coefficients $\boldsymbol{E}_{ab}$ and $\underline{\boldsymbol{D}}_{ab}$

$$\boldsymbol{E}_{ab} = \frac{\Gamma_{ab}}{m_b} \int \mathrm{d}^3 v' \, \underline{\boldsymbol{U}} \cdot \nabla' f_b', \tag{4}$$

$$\underline{\boldsymbol{D}}_{ab} = -\frac{\Gamma_{ab}}{m_a} \int \mathrm{d}^3 v' \, \underline{\boldsymbol{U}} f_b', \tag{5}$$

$$\Gamma_{ab} = \frac{e_a^2 e_b^2 \ln \Lambda_{ab}}{8\pi \epsilon_0^2 m_a}. \tag{6}$$

Please note that although $\boldsymbol{J}_{ab}$ is local in velocity space, the interactions driving $\boldsymbol{J}_{ab}$ are non-local in velocity space.

Mass, momentum, and energy conservation are easily demonstrated by evaluating

$$\begin{aligned} \sum_a \int \mathrm{d}^3 v \, \left[\phi_a \sum_b C_{ab}(f_a, f_b') \right] &= -\sum_{a,b} \int \mathrm{d}^3 v \, \phi_a \nabla \cdot \boldsymbol{J}_{ab} \\ &= \sum_{a,b} \int \mathrm{d}^3 v \, \nabla \phi_a \cdot \boldsymbol{J}_{ab}. \end{aligned} \tag{7}$$

for $\phi_a = m_a$ (mass), $\phi_a = m_a \boldsymbol{v}$ (momentum), and $\phi_a = m_a v^2/2$ (energy). The contribution of the mutual collisions between species a and b is

$$\int \mathrm{d}^3 v \int \mathrm{d}^3 v' \, \frac{\Gamma_{ab}}{m_a m_b} \left(\nabla \phi_a - \frac{m_a}{m_b} \nabla' \phi_b' \right) \cdot \underline{\boldsymbol{U}} \cdot \left[m_a f_a \nabla' f_b' - m_b f_b' \nabla f_a \right]. \tag{8}$$

It is obvious that Eq. (8) equates to zero for the mass and momentum moment. For the kinetic energy moment, one needs to recall that $\boldsymbol{u} \cdot \underline{\boldsymbol{U}} = \underline{\boldsymbol{U}} \cdot \boldsymbol{u} = 0$, which reflects the Galilean invariance and rotational symmetry of the FPL operator. Specifically, the fact that $\underline{\boldsymbol{U}}$ is a projection on the plane perpendicular to $\boldsymbol{u}$ ensures that $C(f_a, f_b') = 0$ when f_a and f_b are Maxwellians with the same temperature. It also embodies the fact that Coulomb collisions in a plasma produce small angle scattering, which can be interpreted as giving colliding particles a small kick perpendicular to their differential velocity $\boldsymbol{u}$.

Due to the linear properties of integration, it is even trivial to show that the conservation properties of the FPL operator remain intact even if we were to cut off the integration over velocity for any species s at $v = Kv_{t,s}$, where $v_{t,s}$ is the thermal velocity of species s, and K is a finite natural number. We illustrate this by introducing such a cut-off for species b. The integral from $v_b = 0$ to $Kv_{t,b}$ is denoted by $\int_c \mathrm{d}^3v$ and the integral from $v_b = Kv_{t,b}$ to infinity by $\int_\infty \mathrm{d}^3v$. The expression in Eq. (8) then splits into the two terms

$$\begin{aligned} &\int \mathrm{d}^3v \int_c \mathrm{d}^3v' \frac{\Gamma_{ab}}{m_a m_b} \left(\nabla\phi_a - \frac{m_a}{m_b} \nabla'\phi'_b \right) \cdot \underline{\boldsymbol{U}} \cdot \left[m_a f_a \nabla' f'_b - m_b f'_b \nabla f_a \right], \\ + &\int \mathrm{d}^3v \int_\infty \mathrm{d}^3v' \frac{\Gamma_{ab}}{m_a m_b} \left(\nabla\phi_a - \frac{m_a}{m_b} \nabla'\phi'_b \right) \cdot \underline{\boldsymbol{U}} \cdot \left[m_a f_a \nabla' f'_b - m_b f'_b \nabla f_a \right], \end{aligned} \tag{9}$$

which vanish independently from each other for the mass, momentum, and energy moments. The generalization to a cut-off for both species is straightforward. This justifies the truncation of the velocity spaces of each species at individual cut-off velocities introduced later in this article for the discretized FPL operator – a highly advantageous feature hidden in the Landau form of the Fokker-Planck operator.

This remarkable property of the FPL operator that the momentum and energy changes due to interactions between each velocity pair $\boldsymbol{v}$ and $\boldsymbol{v}'$ cancel independently from each other is obscured in the Rosenbluth-MacDonald-Judd (RMJ) form of the Fokker-Planck operator [12]. This is due to the representation of the drag and diffusion coefficients in terms of the so-called Rosenbluth potentials, which need to be known in the same velocity range as the distribution function of the colliding species. In order to avoid the use of impractical cut-off velocities in case of RMJ collisions

between species with very different thermal velocities (e.g. ion-electron), Taitano et al. [13] circumvented this constraint by using an asymptotic expansion for the Rosenbluth potentials.

2.2. Gyrophase averaged FPL operator

As in Ref. [9], we assume that the distribution functions of all species are independent of the gyro-phase. We mostly follow the single species procedure pioneered in Ref. [9]. To calculate the gyro-phase averaged FPL operator in cylindrical coordinates $(v_\perp, v_\parallel, \varphi)$ starting from the weak form $\sum_b \int \mathrm{d}^3v\, \phi_a C_{ab}(f_a, f_b')$, one needs to evaluate the gyro-average of terms of the form

$$\int_0^{2\pi}\int_0^{2\pi} \nabla \cdot \underline{\boldsymbol{U}} \cdot \begin{pmatrix} \nabla \\ \nabla' \end{pmatrix} \mathrm{d}\varphi \mathrm{d}\varphi'. \tag{10}$$

The gyro-phase average, which is discussed in detail in Ref. [9], results in the following weak form of the FPL operator

$$\begin{aligned} &2\pi \frac{\partial}{\partial t}\int \mathrm{d}^2v\, \phi_a f_a = \\ &\sum_b \frac{\Gamma_{ab}}{m_a m_b}\int \mathrm{d}^2v \int \mathrm{d}^2v'\, (m_a f_a \nabla\phi_a \cdot \underline{\boldsymbol{U}}_E \cdot \nabla' f_b' - m_b f_b' \nabla\phi_a \cdot \underline{\boldsymbol{U}}_D \cdot \nabla f_a) \\ &= \sum_b \int \mathrm{d}^2v \int \mathrm{d}^2v'\, \nabla\phi_a \cdot \boldsymbol{J}_{ab} = -\sum_b \int \mathrm{d}^2v \int \mathrm{d}^2v'\, \phi_a \nabla \cdot \boldsymbol{J}_{ab}. \end{aligned} \tag{11}$$

In this expression, the gradient in 2-dimensional velocity space is simply $\nabla = \partial_{v_\perp}\hat{\boldsymbol{v}}_\perp + \partial_{v_\parallel}\hat{\boldsymbol{v}}_\parallel$. The tensors $\underline{\boldsymbol{U}}_E$ and $\underline{\boldsymbol{U}}_D$, which define the drag and diffusion coefficients $\boldsymbol{E}_{ab}$ and $\underline{\boldsymbol{D}}_{ab}$, are given by

$$\underline{\boldsymbol{U}}_E = \begin{pmatrix} U_{\perp\perp'} & U_{\perp\parallel} \\ U_{\parallel\perp'} & U_{\parallel\parallel} \end{pmatrix}, \quad \underline{\boldsymbol{U}}_D = \begin{pmatrix} U_{\perp\perp} & U_{\perp\parallel} \\ U_{\perp\parallel} & U_{\parallel\parallel} \end{pmatrix}, \tag{12}$$

where the coefficients U_{xx} are given in Ref. [9]. The gyro-averaged form of the FPL operator, Eq. (11), still conserves mass and energy. However, only the momentum in the direction parallel to the magnetic field is conserved. The perpendicular translational momentum is zero after gyro-averaging (without affecting the magnetic moment). To demonstrate the conservation laws, one again needs to analyze the contributions of the mutual collisions between species a and b,

$$\begin{aligned}
&\int \mathrm{d}^2v \int \mathrm{d}^2v' \, [\phi_a C(f_a, f_b') + \phi_b' C(f_b', f_a)] = \\
&= \int \mathrm{d}^2v \int \mathrm{d}^2v' \left[m_a f_a \left(\nabla\phi_a \cdot \underline{\boldsymbol{U}}_E - \frac{m_a}{m_b} \nabla'\phi_b' \cdot \underline{\boldsymbol{U}}_D' \right) \cdot \nabla' f_b' \right. \\
&\qquad \left. - m_b f_b' \left(\nabla\phi_a \cdot \underline{\boldsymbol{U}}_D - \frac{m_a}{m_b} \nabla'\phi_b' \cdot \underline{\boldsymbol{U}}_E' \right) \cdot \nabla f_a \right]. \qquad (13)
\end{aligned}$$

The expression in Eq. (13) equates to zero for the parallel momentum moment ($\phi_{a,b} = m_{a,b} v_\parallel \hat{\boldsymbol{v}}_\parallel$) and the energy moment ($\phi_{a,b} = (m_{a,b}/2)(v_\perp^2 + v_\parallel^2)$) because the tensors $\underline{\boldsymbol{U}}_E$ and $\underline{\boldsymbol{U}}_D$ fulfill the following identities:

$$\begin{aligned}
&\begin{pmatrix} 0 \\ 1 \end{pmatrix} \cdot (\underline{\boldsymbol{U}}_E - \underline{\boldsymbol{U}}_D') = 0, \quad \begin{pmatrix} 0 \\ 1 \end{pmatrix} \cdot (\underline{\boldsymbol{U}}_D - \underline{\boldsymbol{U}}_E') = 0, \\
&\begin{pmatrix} v_\perp \\ v_\parallel \end{pmatrix} \cdot \underline{\boldsymbol{U}}_E - \begin{pmatrix} v_\perp' \\ v_\parallel' \end{pmatrix} \cdot \underline{\boldsymbol{U}}_D' = 0, \quad \begin{pmatrix} v_\perp \\ v_\parallel \end{pmatrix} \cdot \underline{\boldsymbol{U}}_D - \begin{pmatrix} v_\perp' \\ v_\parallel' \end{pmatrix} \cdot \underline{\boldsymbol{U}}_E' = 0, \qquad (14)
\end{aligned}$$

where $\underline{\boldsymbol{U}}_E'$ and $\underline{\boldsymbol{U}}_D'$ result from $\underline{\boldsymbol{U}}_E(\boldsymbol{v}, \boldsymbol{v}')$ and $\underline{\boldsymbol{U}}_D(\boldsymbol{v}, \boldsymbol{v}')$ after swapping $\boldsymbol{v}$ and $\boldsymbol{v}'$.

2.3. Discretization of the FPL collision operator

For numerical treatment, we discretize the gyro-phase averaged weak form of the FPL operator given in Eq. (11),

$$2\pi \frac{\partial}{\partial t} \int \mathrm{d}^2v \, \phi_a f_a = -\sum_b \int \mathrm{d}^2v \int \mathrm{d}^2v' \, \phi_a \nabla \cdot \boldsymbol{J}_{ab}. \qquad (15)$$

Since the FPL collision operator allows for the use of individual cut-off velocities for each species, we discretize velocity space into uniform rectangular grids such that $-Kv_{t,s} \leq v_{\parallel,s} \leq Kv_{t,s}$ with $\Delta v_\parallel = 2Kv_{t,s}/(N-1)$, and $0 \leq v_{\perp,s} \leq Kv_{t,s}$ with $\Delta v_\perp = Kv_{t,s}/(M-1)$, where K is a small natural number, and N and M are the number of grid points in the parallel and perpendicular direction. The volume element becomes $\Delta V(J) = J\Delta v_\perp^2 \Delta v_\parallel$ with $0 \leq J \leq M$. As in typical fluid simulations, distribution functions are evaluated on the grid points (I, J) with $0 \leq I \leq N$ and $0 \leq J \leq M$, and fluxes $\boldsymbol{J}_{ab}$ are evaluated on a staggered grid $(I \pm 1/2, J \pm 1/2)$. Fluxes at $I - 1/2 < 0$, $I + 1/2 > N$, $J - 1/2 < 0$, and $J + 1/2 > M$ are zero to make the discretized operator conservative by preventing fluxes into and out of the velocity space grid. A sketch of this discrete velocity grid is shown in Fig. 1. The resulting discretized operator becomes

$$2\pi \frac{\partial}{\partial t} \sum_{I=1}^{N} \sum_{J=1}^{M} \Delta V(J) \phi_a(I,J) f_a(I,J) \\ = -\sum_b \sum_{I=1}^{N} \sum_{J=1}^{M} \Delta V(J) \phi_a(I,J) \left(\nabla \cdot \boldsymbol{J}_{ab}\right)(I,J). \tag{16}$$

Since Eq. (16) must hold for arbitrary test functions ϕ_a, the system of equations that need to be solved numerically is

$$2\pi \frac{\partial}{\partial t} f_a(I,J) = -\left(\nabla \cdot \boldsymbol{J}_{ab}\right)(I,J). \tag{17}$$

To preserve Gauss's divergence theorem and, thus, the conservation laws in discretized velocity space, it is necessary to define the discrete divergence

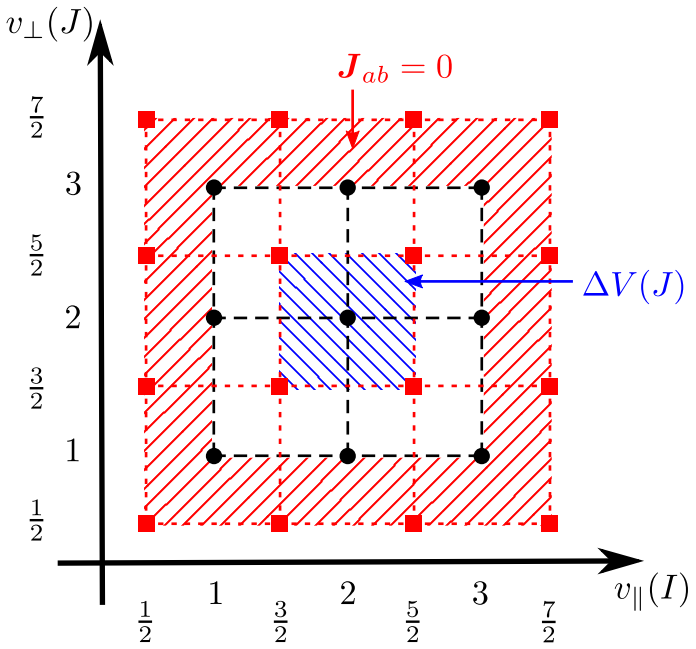

Figure 1: Illustration of the discretized velocity space with 3×3 grid points. Dots mark the velocity grid, squares the corresponding staggered grid. In the volume marked with red stripes (bottom left to top right), the velocity space flux $\boldsymbol{J}_{ab}$ vanishes. The square marked with blue stripes (top left to bottom right) represents a volume element $\Delta V(J)$. Consequently, there are no fluxes into and out of the velocity grid.

operator as

$$
\begin{aligned}
(\nabla \cdot \boldsymbol{J}_{ab})\,(I,J) &= \frac{(J^{\perp}_{ab})(I,J+\frac{1}{2}) - (J^{\perp}_{ab})(I,J-\frac{1}{2})}{\Delta_{v_\perp}} \\
&\quad + \frac{(J^{\parallel}_{ab})(I+\frac{1}{2},J) - (J^{\parallel}_{ab})(I-\frac{1}{2},J)}{\Delta_{v_\parallel}}, \\
\boldsymbol{J}_{ab}(I,J\pm\frac{1}{2}) &= \frac{\Delta V_a(J\pm\frac{1}{2})}{2\Delta V_a(J)}\left[\boldsymbol{J}_{ab}(I+\frac{1}{2},J\pm\frac{1}{2}) + \boldsymbol{J}_{ab}(I-\frac{1}{2},J\pm\frac{1}{2})\right], \\
\boldsymbol{J}_{ab}(I\pm\frac{1}{2},J) &= \frac{1}{2\Delta V_a(J)}\Big[\Delta V_a(J+\frac{1}{2})\boldsymbol{J}_{ab}(I\pm\frac{1}{2},J+\frac{1}{2}) \\
&\quad + \Delta V_a(J-\frac{1}{2})\boldsymbol{J}_{ab}(I\pm\frac{1}{2},J-\frac{1}{2})\Big], \qquad (18)
\end{aligned}
$$

where $J^{\perp}_{ab} = \boldsymbol{J}_{ab}\cdot\hat{\boldsymbol{v}}_\perp$ and $J^{\parallel}_{ab} = \boldsymbol{J}_{ab}\cdot\hat{\boldsymbol{v}}_\parallel$. This makes sense insofar as fluxes at higher J are associated with larger phase space volumes and can be thought to have larger influence on the flux-balance of the volume element (I,J). The flux on the staggered grid is given by

$$
\begin{aligned}
\boldsymbol{J}_{ab}(I\pm\frac{1}{2},J\pm\frac{1}{2}) &= \boldsymbol{E}_{ab}(I\pm\frac{1}{2},J\pm\frac{1}{2})f_a(I\pm\frac{1}{2},J\pm\frac{1}{2}) \\
&\quad + \underline{\boldsymbol{D}}_{ab}(I\pm\frac{1}{2},J\pm\frac{1}{2})\cdot(\nabla f_a)(I\pm\frac{1}{2},J\pm\frac{1}{2}), \\
\boldsymbol{E}_{ab}(I\pm\frac{1}{2},J\pm\frac{1}{2}) &= \frac{\Gamma_{ab}}{m_a m_b}\sum_{i=1}^{N-1}\sum_{j=1}^{M-1}\Delta V_b(j+\frac{1}{2})m_a \\
&\quad \times \underline{\boldsymbol{U}}_E(I\pm\frac{1}{2},J\pm\frac{1}{2},i+\frac{1}{2},j+\frac{1}{2})\cdot(\nabla' f'_b)(i+\frac{1}{2},j+\frac{1}{2}), \\
\underline{\boldsymbol{D}}_{ab}(I\pm\frac{1}{2},J\pm\frac{1}{2}) &= -\frac{\Gamma_{ab}}{m_a m_b}\sum_{i=1}^{N-1}\sum_{j=1}^{M-1}\Delta V_b(j+\frac{1}{2})m_b \\
&\quad \times \underline{\boldsymbol{U}}_D(I\pm\frac{1}{2},J\pm\frac{1}{2},i+\frac{1}{2},j+\frac{1}{2})f'_b(i+\frac{1}{2},j+\frac{1}{2}). \qquad (19)
\end{aligned}
$$

Values of f_a, f_b, and their gradients on the staggered grid are obtained by

simple linear interpolation:

$$
\begin{aligned}
f(i+\frac{1}{2},j) &= \frac{1}{2}\left[f(i+1,j)+f(i,j)\right], \\
f(i+\frac{1}{2},j+\frac{1}{2}) &= \frac{1}{2}\left[f(i+\frac{1}{2},j+1)+f(i+\frac{1}{2},j)\right], \\
(\partial_{v_\perp} f)(i+\frac{1}{2},j+\frac{1}{2}) &= \frac{f(i+\frac{1}{2},j+1)-f(i+\frac{1}{2},j)}{\Delta_\perp}.
\end{aligned} \tag{20}
$$

2.4. Discrete conservation laws

To prove that mass, momentum, and energy are conserved exactly in this discretization, i.e. that Eq. (13) holds in the discretized velocity space, it is useful to show first that Eq. (7) holds in the discretized velocity space. By substituting Eq. (18) into the RHS of Eq. (16), one obtains

$$
\begin{aligned}
&-\sum_{I=1}^{N}\sum_{J=1}^{M}\Delta V_a(J)\phi_a(I,J)\nabla\cdot\boldsymbol{J}_{ab}(I,J) = -\sum_{I=1}^{N}\sum_{J=1}^{M}\phi_a(I,J) \\
&\left\{\frac{1}{2\Delta_{v_\perp}}\left[\Delta V_a(J+\frac{1}{2})\left(J^{\perp}_{ab}(I+\frac{1}{2},J+\frac{1}{2})+J^{\perp}_{ab}(I-\frac{1}{2},J+\frac{1}{2})\right)\right.\right. \\
&\left.-\Delta V_a(J-\frac{1}{2})\left(J^{\perp}_{ab}(I+1,J-\frac{1}{2})+J^{\perp}_{ab}(I-\frac{1}{2},J-\frac{1}{2})\right)\right] \\
&+\frac{1}{2\Delta_{v_\parallel}}\left[\Delta V_a(J+\frac{1}{2})\left(J^{\parallel}_{ab}(I+\frac{1}{2},J+\frac{1}{2})-J^{\parallel}_{ab}(I-\frac{1}{2},J+\frac{1}{2})\right)\right. \\
&\left.\left.+\Delta V_a(J-\frac{1}{2})\left(J^{\parallel}_{ab}(I+\frac{1}{2},J-\frac{1}{2})-J^{\parallel}_{ab}(I-\frac{1}{2},J-\frac{1}{2})\right)\right]\right\}.
\end{aligned} \tag{21}
$$

After reordering the sums using the fact that the flux is zero on the staggered grid outside of the velocity grid boundaries, and after sorting for the flux

on the staggered grid, one finds

$$\sum_{I=1}^{N-1}\sum_{J=1}^{M-1}\Delta V_a(J+\frac{1}{2})\frac{1}{2\Delta_{v_\perp}}J^{\perp}_{ab}(I+\frac{1}{2},J+\frac{1}{2})$$
$$\times[\phi_a(I+1,J+1)+\phi_a(I,J+1)-\phi_a(I+1,J)-\phi_a(I,J)]$$
$$+\frac{1}{2\Delta_{v_\parallel}}J^{\parallel}_{ab}(I+\frac{1}{2},J+\frac{1}{2})$$
$$\times[\phi_a(I+1,J)-\phi_a(I,J)+\phi_a(I+1,J+1)-\phi_a(I,J+1)]$$
$$=\sum_{I=1}^{N-1}\sum_{J=1}^{M-1}\Delta V_a(J+\frac{1}{2})(\nabla\phi_a)(I+\frac{1}{2},J+\frac{1}{2})\cdot\boldsymbol{J}_{ab}(I+\frac{1}{2},J+\frac{1}{2}). \qquad (22)$$

Equations (21) and (22) prove Eq. (7) in the discretized velocity space. The discretized version of Eq. (13) is then obtained by adding the corresponding term resulting from b-a collisions to Eq. (22) and substituting the discrete fluxes $\boldsymbol{J}_{ab}$ and $\boldsymbol{J}_{ba}$ by the definition in Eq. (19):

$$\sum_{I=1}^{N-1}\sum_{J=1}^{M-1}(\nabla\phi_a)(I+\frac{1}{2},J+\frac{1}{2})\cdot\boldsymbol{J}_{ab}(I+\frac{1}{2},J+\frac{1}{2})\Delta V_a(J+\frac{1}{2})$$
$$+\sum_{i=1}^{N-1}\sum_{j=1}^{M-1}(\nabla'\phi'_b)(i+\frac{1}{2},j+\frac{1}{2})\cdot\boldsymbol{J}_{ba}(i+\frac{1}{2},j+\frac{1}{2})\Delta V_b(j+\frac{1}{2})$$
$$=\frac{\Gamma_{ab}}{m_am_b}\sum_{I=1}^{N-1}\sum_{J=1}^{M-1}\sum_{i=1}^{N-1}\sum_{j=1}^{M-1}\Delta V_a\Delta V'_b$$
$$\left[m_af_a\left(\nabla\phi_a\cdot\underline{\boldsymbol{U}}_E-\frac{m_a}{m_b}\nabla'\phi'_b\cdot\underline{\boldsymbol{U}}'_D\right)\cdot\nabla'f'_b\right.$$
$$\left.-m_bf'_b\left(\nabla\phi_a\cdot\underline{\boldsymbol{U}}_D-\frac{m_a}{m_b}\nabla'\phi'_b\cdot\underline{\boldsymbol{U}}'_E\right)\cdot\nabla f_a\right], \qquad (23)$$

where primed quantities depend on the lower case summation indices $(i+1/2, j+1/2)$, unprimed quantities depend on upper case summation indices $(I+1/2, J+1/2)$, and the tensors $\underline{\boldsymbol{U}}^{(\prime)}_E$ and $\underline{\boldsymbol{U}}^{(\prime)}_D$ depend on all summation indices. Thus, mass, parallel momentum, and energy are conserved exactly

by the discretized FPL operator described in this article provided that the finite difference expressions for $\nabla\phi_s$ yield $m_s(0,1)$ and $2m_s(v_\perp, v_\parallel)$ for $\phi_s = m_s\boldsymbol{v}_\parallel$ and $\phi_s = m_s\boldsymbol{v}^2$. This is true for our simple finite difference derivatives:

$$\begin{aligned}
\partial_{v_\parallel} v_\parallel(I) &= \frac{v_\parallel(I+1) - v_\parallel(I-1)}{2\Delta_{v_\parallel}}, \\
\partial_{v_\perp} v_\perp^2 &= \frac{v_\perp^2(I+1) - v_\perp^2(I-1)}{2\Delta_{v_\perp}} \\
&= \frac{(v_\perp(I+1) + v_\perp(I-1))(v_\perp(I+1) - v_\perp(I-1))}{2\Delta_{v_\perp}} \\
&= v_\perp(I+1) + v_\perp(I-1) = 2v_\perp(I). \qquad (24)
\end{aligned}$$

Moreover, the discrete conservation laws in Eq. (23) retain the detailed conservation of the continuum FPL operator. In contrast, the RMJ operator developed by Taitano et al. [7] conserves mass, momentum, and energy only globally. In addition to accurate conservation of mass, energy and momentum, physical solutions require that the overall entropy $S = -\sum_a \int f_a \ln(f_a) \mathrm{d}^3 v$ increases, i.e. that the H-theorem is satisfied. We have not observed a case of decreasing entropy yet in an ordinary tokamak plasma. The relaxation of distorted Maxwellians with different temperatures to the equilibrium Maxwellian presented in Sec. 3.2 demonstrates that the entropy indeed increases, and the H-theorem is satisfied. However, the user has to pay attention to any odd behavior in their special problems.

2.5. Conservation of the thermal equilibrium

The equilibrium conservation scheme used in Ref. [9] for the single-species FPL operator can be applied to the multi-species formulation of the FPL operator as well. From Eq. (19) in Ref. [9], we obtain solutions for

the interpolation weights $\delta_\perp$ and $\delta_\parallel$,

$$\delta_\perp(J+\frac{1}{2}) = \begin{cases} \left\{1-\exp[\frac{1}{2}\Delta_{v_\perp} v_\perp (J+\frac{1}{2})/v_t^2]\right\}^{-1} - \Lambda_\perp^{-1} \\ \left\{1-\exp[\frac{1}{2}\Delta_{v_\perp} v_\perp (J+\frac{1}{2})/v_t^2]\right\}^{-1} \end{cases},$$

$$\delta_\parallel(I+\frac{1}{2}) = \begin{cases} \left\{1-\exp[\frac{1}{2}\Delta_{v_\parallel} v_\parallel (I+\frac{1}{2})/v_t^2]\right\}^{-1} - \Lambda_\parallel^{-1} \\ \left\{1-\exp[\frac{1}{2}\Delta_{v_\parallel} v_\parallel (I+\frac{1}{2})/v_t^2]\right\}^{-1} \end{cases}, \tag{25}$$

where $\Lambda_\perp = -v_\perp(J+\frac{1}{2})\Delta_{v_\perp}/v_t^2$ and $\Lambda_\parallel = -v_\parallel(I+\frac{1}{2})\Delta_{v_\parallel}/v_t^2$ [$v_t = (T/m)^{1/2}$]. One of the two solutions for each, $\delta_\perp$ and $\delta_\parallel$, is always between 0 and 1 and is used as interpolation weight in the calculation of the distribution functions on the staggered grid. With these interpolation weights, Eq. (20) is generalized to

$$\begin{aligned} f(i+\frac{1}{2},j) &= \delta_\parallel(i+\frac{1}{2}) f(i+1,j) + [1-\delta_\parallel(i+\frac{1}{2})] f(i,j), \\ f(i+\frac{1}{2},j+\frac{1}{2}) &= \delta_\perp(j+\frac{1}{2}) f(i+\frac{1}{2},j+1) \\ &\quad + [1-\delta_\perp(j+\frac{1}{2})] f(i+\frac{1}{2},j), \\ (\partial_{v_\perp} f)(i+\frac{1}{2},j+\frac{1}{2}) &= \frac{f(i+\frac{1}{2},j+1) - f(i+\frac{1}{2},j)}{\Delta_\perp}. \end{aligned} \tag{26}$$

Note that the definition of the finite difference derivative does not change except for the definition of the distribution function on the staggered grid.

While the thermal velocity to be used in the calculation of $\delta_\perp$ and $\delta_\parallel$ is unique in single-species simulations, there is more freedom in simulations with multiple particle species. The temperature corresponding to the equilibrium Maxwellian in the multi-species case is $T_{eq} = (\sum_s n_s T_s)/(\sum_s n_s)$, which is simply $T_{eq} = (T_i + T_e)/2$ in case of singly charged ions and electrons. One could use the corresponding thermal velocity $v_{t,eq}$ in the calculation of the interpolation weights for all collision processes. However, the

like-particle collisions seek to drive the distribution functions to the thermal equilibrium of the respective species, and inter-species collisions drive the distribution function to the inter-species equilibrium. This is especially important if the collision frequency of one species is much greater than the collision frequencies of the other species. Therefore, we use $v_t = v_{t,s}$ in like-particle collisions and $v_t = v_{t,ab} = (n_a T_a + n_b T_b)/(n_a + n_b)$ in inter-species collisions between species a and b.

As discussed in Ref. [9], this equilibrium conservation scheme does not guarantee positivity of the distribution function. If a distribution function becomes negative on a specific grid point after a collision time step, it is corrected to be a negligibly small value, thereby allowing for error. This is usually a rare event provided that the noise level in the distribution functions is controlled by using enough marker particles and appropriate resolution in velocity space.

2.6. Implementation in particle-in-cell codes

In order to connect the grid based collision operator described in the previous sections to particle-in-cell codes, one needs to combine the solver for Eq. (17) with a time integrator and with a particle-mesh interpolation for obtaining the plasma distribution functions on the velocity grid.

We implemented our FPL operator in the codes XGC1 [10] and XGCa [11]. Both codes are global, gyrokinetic PIC codes for the simulation of tokamak fusion plasmas and are designed for extreme-scale high-performance computing. The XGC codes calculate the motion of marker particles in a 5-dimensional phase space (3-dimensional configuration space, 2-dimensional velocity space). The specialty of the XGC codes is that the configuration

space includes the whole plasma volume from the magnetic axis to the inner reactor wall. While XGC1 is a gyrokinetic turbulence code, XGCa excludes turbulence by assuming that electric and magnetic fields are axisymmetric and, thus, describes only neoclassical physics.

We use the same backward Euler time discretization and Picard iteration as in Ref. [9]. The generalization of the implicit time integrator described therein is straightforward and leads to

$$\frac{\mathrm{d}f_a}{\mathrm{d}t} = \frac{f_{a,i+1}^{(k)} - f_{a,i}}{\Delta t} = \sum_b C(f_{a,i+1}^{(k)}, f_{b,i+1}^{(k-1)}), \tag{27}$$

where i is the time index, and k is the iteration index of the Picard iteration. The distribution functions of all species need to be advanced together, and the new convergence criterion for the Picard iteration is that the relative errors of the total mass, energy, and parallel momentum are below a certain threshold (usually 10^{-6}). Due the exact conservation laws of the discretized FPL operator, the Picard iteration rapidly converges to a solution that conserves mass, parallel momentum, and energy. Convergence is, however, not guaranteed, e.g. if the initial guess is too far from the actual solution or the collision time step is too large. Therefore, the number of Picard iterations is usually limited to 20 in our simulations, and the collision operation is discarded with an error estimate where convergence is not reached within the target number of iterations. As a rule of thumb, we found that the Picard iteration converges well (within ~ 5 iterations or less) if the time step of the collision operation is at or below the lowest electron collision time of the system $\tau_e \approx T_{e,min}^{3/2}/(5 \cdot 10^{-11} n_{min})$ [14], where $T_{e,min}$ is the minimum of the electron temperature in eV in the global simulation area, and n_{min} is the density in m^{-3} at the corresponding location. Thus, the time step

that needs to be used for the collision operator is effectively determined by the species with the highest collision frequency and is comparable to the time step that needs to be used for the ion particle motion ($\sim 10^{-7}$ s). Since $\tau_e \ll \tau_i$, ion collisions at all energies, and since the collision time becomes longer at higher energy, and electron collisions at intermediate to high energies are sufficiently resolved. The use of an implicit method has the advantage that in order to be stable the time step of the collision operator does not have to be much smaller than the electron collision time as can be expected for a simple explicit method. For very low energy electrons, $mv^2/2 \ll T_e$, however, we admit that there is a possibility that the implicit method compromises accuracy. In the present application of the collision operator, there is only a negligible fraction of low energy electrons so that the use of an implicit method is justified when the implicit time step is $\sim \tau_e$. In some other applications there may be more lower energy electrons, whose collision time is much shorter than the implicit time step, and the use of improved explicit time integration methods such as Refs. [15] and [16] may be considered. Alternative methods for handling the disparate time scales of electron and ion collisions such as the ones discussed in Refs. [17, 18] have not been considered in this work.

In order to obtain the plasma distribution functions before solving the FPL operator, mesh-particle interpolation is necessary in configuration and velocity space. The properties of the mesh-particle interpolation have been discussed in Ref. [9]. It is important to note that mesh-particle interpolation introduces an additional source for errors in the conservation laws of the collision operator. The interpolation errors are studied in Sec. 3.2.

3. Verification

In the following, we present results of various verification studies performed with the code XGCa that demonstrate the functionality of the FPL collision operator discussed in this article.

3.1. Conservation of the thermal equilibrium

The most fundamental test of our implementation of the FPL operator in XGC1 and XGCa is the conservation of the thermal equilibrium. For this purpose, we ran XGCa with two particle species, Deuterium ions ($m_D = 3.34 \cdot 10^{-27}$ kg) and electrons ($m_e = 9.109 \cdot 10^{-31}$ kg), but without evaluating the left-hand side of the gyrokinetic equation, i.e. with deactivated particle motion. Thus, the time evolution of the plasma distribution functions is due to collisional processes alone, and each collision cell in configuration space is completely independent of each other.

In order to demonstrate the effect of the equilibrium conservation scheme described in section 2.5, we ran this conservation test once with dynamic (equilibrium-conserving) interpolation weights $\delta_\parallel$ and $\delta_\perp$ according to Eq. (26) and once with fixed (not equilibrium-conserving) $\delta_\parallel = \delta_\perp = 1/2$. The cut-off velocity for both species is $v_{c,s} = 4v_{t,s}$ with a velocity grid of 41×41 grid points. We initialized the simulation in thermal equilibrium, i.e. each species started with a Maxwellian distribution function at temperature $T = T_{eq} = 300$ eV, and then we tracked the root-mean-square (RMS) of the relative deviation from the initial Maxwellian distribution function over 20000 time steps (corresponding to $\sim$ 16 ion collision times). The distribution functions were diagnosed every 500 time steps. The results are shown in Fig. 2. In both cases, with fixed and dynamic coefficients $\delta_\parallel$ and

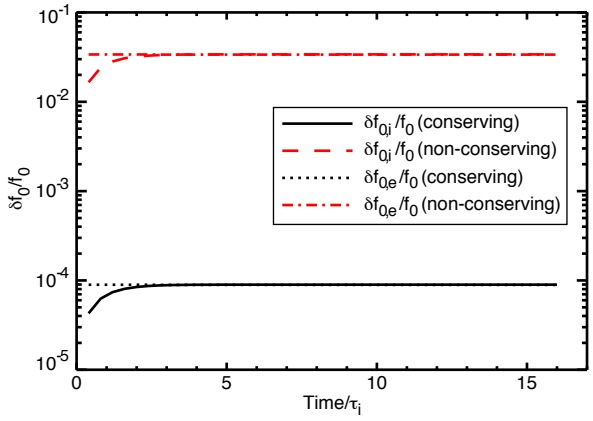

Figure 2: Quality of the conservation of the thermal equilibrium. Ions and electrons are initialized in thermal equilibrium at $T_{eq} = 300$ eV with Maxwellian distribution functions. After 20000 collision time steps, the root-mean-square relative deviation of the distribution functions from the initial Maxwellian is $\sim 10^{-4}$ with activated equilibrium conservation mechanism while it is 3-4% with deactivated equilibrium conservation.

$\delta_\perp$, the RMS deviation of the ion distribution function grows within the first few ion collision times but is stable afterwards. It is reasonable to assume analogous transient behavior for the electron distribution function on the electron collision time scale, which is, however, not resolved in the diagnostic data.

Despite the similar time evolution, the difference between the simulations with activated (dynamic $\delta_{\parallel/\perp}$) and deactivated (fixed $\delta_{\parallel/\perp}$) equilibrium conservation scheme is drastic. While in case of the former, the RMS of the relative deviation from the equilibrium Maxwellian does not exceed 10^{-4}, the deviation is between 3 and 4% in case of the latter.

3.2. Temperature isotropization and flow relaxation

In this section, we verify that our FPL collision operator leads to the decay of a perturbed plasma to the thermal equilibrium in a simple, but

comprehensive two-species relaxation test performed with XGCa with Deuterium ions and electrons. As in the equilibrium conservation test described in Sec. 3.1, the left-hand side of the gyrokinetic equation is not evaluated. In contrast to the conservation test, however, we initialize ions and electrons with shifted bi-Maxwellians

$$f_s(v_\parallel, v_\perp) = \frac{n}{\alpha(2\pi)^{3/2} v_{t,s}^3} \exp\left[-m_s \frac{(v_\parallel - u_{s,\parallel})^2 + \alpha^{-1} v_\perp^2}{2eT_{s,0}}\right], \qquad (28)$$

where $T_{s,\parallel} = T_{s,0}$ and $T_{s,\perp} = \alpha T_{s,0}$ are the parallel and perpendicular temperatures, m_s is the mass, and $u_{s,\parallel}$ is the parallel fluid flow of species s, and e is the elementary charge. The mean temperature of species s is given by $T_s = (2T_{s,\perp} + T_{s,\parallel})/3$. The thermal velocity $v_{t,s}$ is evaluated for the base temperature $T_{s,0}$. For our test, we chose $T_{0,i} = 200$ eV, $T_{0,e} = 300$ eV, $\alpha = 1.3$, $u_{e,\parallel} = 0.5(m_e/m_i)^{1/2} v_{t,e}$, and $u_{i,\parallel} = 50(m_e/m_i) v_{t,i}$. The initial values of the ion and electron flows have values representative of typical tokamak plasmas, in which ions carry most of the parallel momentum. In thermal equilibrium, $T_{s,\parallel} = T_{s,\perp}$, $T_i = T_e$, and $u_{i,\parallel} = u_{e,\parallel}$. Our choice of the initial state enables us to observe the different time scales of the isotropization of the electron and ion temperatures, the relaxation of the temperature difference between ions and electrons, and the relaxation of ion and electron flows. In addition, the quality of mass, momentum, and energy conservation during the relaxation process can be studied.

Figures 3 and 4 show the time evolution of the ion and electron temperatures and flows from a simulation without mesh particle interaction. Temperatures and flows obtained from the simulation are compared to the analytical result based on the isotropization and energy transfer frequencies given in Ref. [19], and the flow relaxation frequency based on the friction

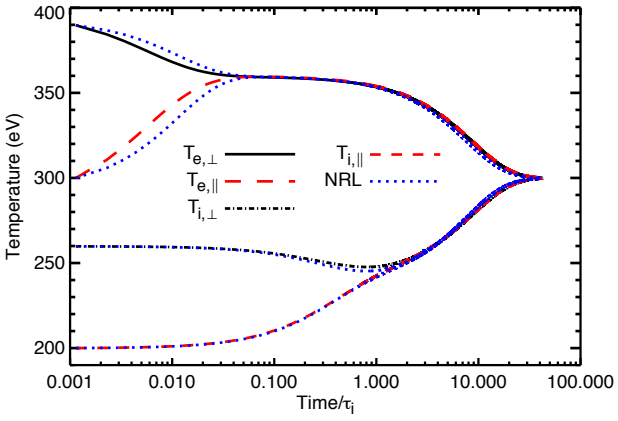

Figure 3: Relaxation of the perpendicular and parallel ion and electron temperatures towards thermal equilibrium. The results of the non-linear FPL operator are compared to the relaxation rates given in the NRL Plasma Formulary [19].

force for large mass ratios given in Eq. (1.17) in Ref. [20],

$$\nu_S = \frac{e^4 n_i \ln \Lambda_{ie}}{3(2\pi)^{3/2}\epsilon_0^2 \sqrt{m_e}(eT_e)^{3/2}}. \tag{29}$$

The relaxation time scales observed in the simulation agree very well with the theoretical predictions. In case of the electron temperature isotropization and the flow relaxation, the theoretical relaxation rates – while having the correct order of magnitude – differ from the numerical results by a factor of approximately 2. These differences are due to the arbitrary assumption made in the calculation of the analytical result that the particle distribution functions are always accurately described by Eq. (28).

Since there are two contributions to the conservation errors in our implementation of the FPL collision operator, we ran two sets of relaxation tests. In the first set of simulations, mesh-particle interpolation was deactivated so that the only error source is the implicit time integration. This error is limited by the convergence criterion applied to the Picard iteration in the

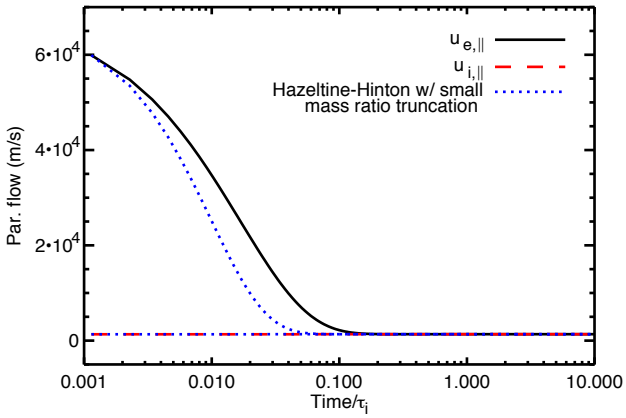

Figure 4: Relaxation of the parallel ion and electron flows. The results of the non-linear FPL operator are compared to Eq. (29), i.e. the ion-electron friction with small mass ratio truncation obtained from Eq. (1.17) in Ref. [20]. Differences between the analytical result and the simulations are due to the fact that the ion and electron distribution functions are not always exactly shifted bi-Maxwellians as defined in Eq. (28).

implicit time stepper (cf. Sec. 2.6). In the second set of simulations, mesh-particle interpolation is activated, which introduces an additional diffusive interpolation error, which can be controlled by the number of particles per collision cell. The accumulated relative errors of momentum and energy with deactivated mesh-particle interpolation are shown in Figs. 5 and 6 for three different cases. We tested with cut-off velocities of 4, 5, and $6v_{t,s}$ and grid sizes of 41×41, 51×51, and 61×61 grid points. The difference between the three cases is marginal.

While the relative error per collision operation cannot exceed the limit set by the convergence criterion of the implicit time stepper in cases without mesh-particle interpolation, the error per time step can be larger than the convergence criterion with activated mesh-particle interpolation. Therefore, we investigated the influence of the number of marker particles per collision

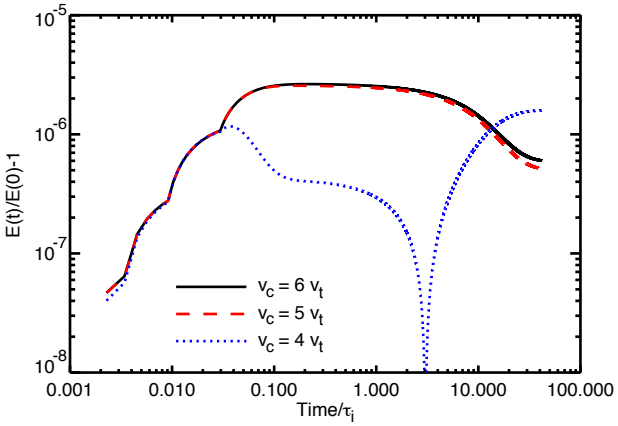

Figure 5: Accumulated absolute value of the relative error of the total kinetic energy in the relaxation test depicted in Figs. 3 and 4 without mesh particle interpolation. The dependence on the cut-off velocity of the velocity grid used by the non-linear FPL operator is marginal.

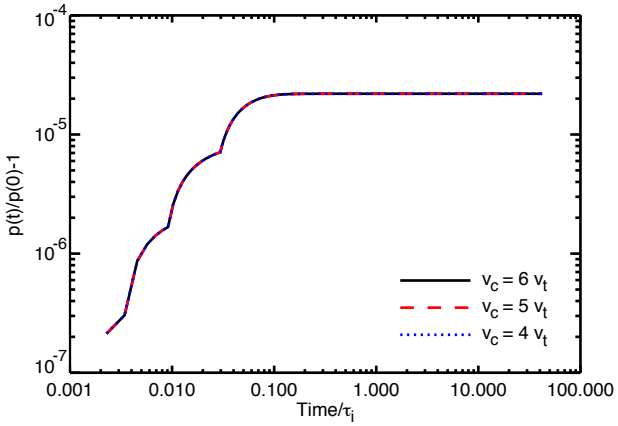

Figure 6: Accumulated absolute value of the relative error of the total parallel momentum in the relaxation test depicted in Figs. 3 and 4 without mesh particle interpolation. There is no dependence on the cut-off velocity of the velocity grid used by the non-linear FPL operator.

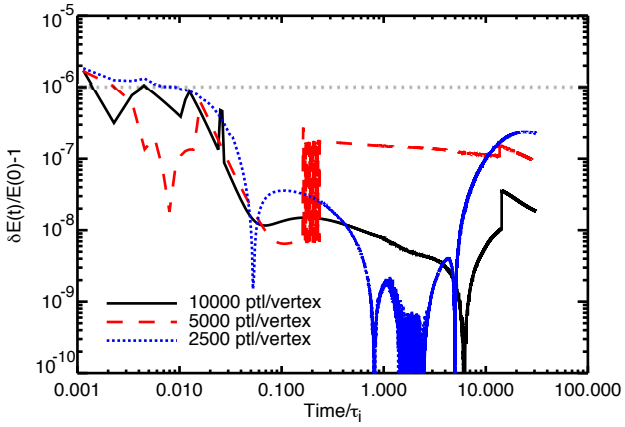

Figure 7: Absolute value of the relative error of the total kinetic energy per time step in the relaxation test depicted in Figs. 3 and 4 with mesh particle interpolation. The additional diffusive error due to mesh particle interpolation in velocity space increases the relative error per time step only slightly above the threshold of 10^{-6} demanded by the implicit time integrator of the non-linear FPL operator. The dependence on the number of particles per collision cell is weak.

cell in the simulations with activated mesh-particle interpolations. We used 2500, 5000, and 10000 particles per collision cell. The latter is a realistic value for production runs of XGC1 and XGCa. Figures 7 and 8 show the relative error of momentum and energy per collision time step compared to the convergence criterion of the implicit time stepper. As expected, the energy and momentum errors can be larger than the convergence criterion due to mesh-particle interaction. In case of the kinetic energy, the error per time step depends only weakly on the number of particles per collision cell. In contrast, the momentum error during the fast processes of electron temperature isotropization and flow relaxation is larger than the convergence criterion by a factor of up to 30 when using only 2500 particles per collision cell. With 10000 particles per collision cell, the momentum error is only

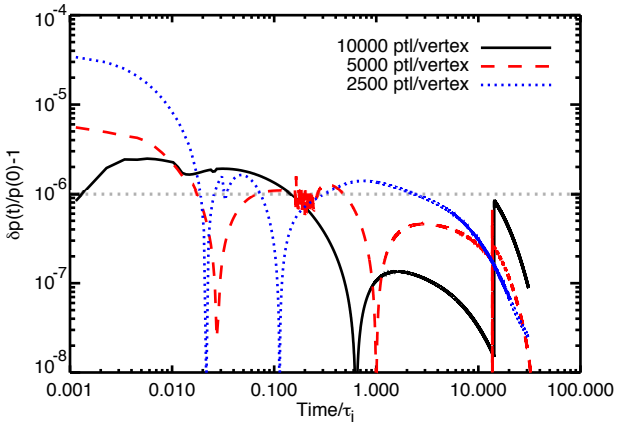

Figure 8: Absolute value of the relative error of the total parallel momentum per time step in the relaxation test depicted in Figs. 3 and 4 with mesh particle interpolation. The additional diffusive error due to mesh particle interpolation in velocity space increases the relative error per time step by a factor of up to 30 above the threshold of 10^{-6} required by the implicit time integrator of the non-linear FPL operator. With 10000 particles per collision cell, the diffusive interpolation error is reduced such that the effective relative error is close to the desired threshold.

slightly enhanced.

Verification studies of full XGCa simulations are beyond the scope of this article and will be published separately. We only want to remark that we found good agreement between the bootstrap currents calculated with XGCa and the neoclassical code NEO [21, 4] in the local regime, for which NEO is constructed, even in diverted geometry [11].

4. Performance considerations

Since the collision operator discussed in this article is intended for use in high-performance computing (HPC) applications, scalability is an important requirement to make this operator viable on today's and on future HPC systems. We employ our version of the FPL (Eulerian) collision operator in the XGC particle-in-cell applications. For these applications, the particle-related work typically exhibits a greater degree of exploitable parallelism than the phase space mesh-related work, and scales better as a result. Defining the mesh size to be the number of vertices in the configuration space mesh, the complexity of the mesh-related work (field solver, collision operator) is close to linear in the mesh size, and, even without parallel overhead inefficiencies, will demonstrate degraded scalability when the number of computational threads is comparable to (or larger than) the number of mesh vertices. Note that for practical grid sizes in XGC1 or XGCa simulations ($N < 4000$), the scaling of the collision operator with the velocity grid size N is between $\mathrm{O}(N)$ and $\mathrm{O}(N^2)$. The reason for this beneficial scaling is that the computing time needed for the calculation of the drag and diffusion coefficients $\boldsymbol{E}$ and $\underline{\boldsymbol{D}}$, which scales quadratically with the size of the velocity grid [$\mathrm{O}(N^2)$] is not yet the dominant computational

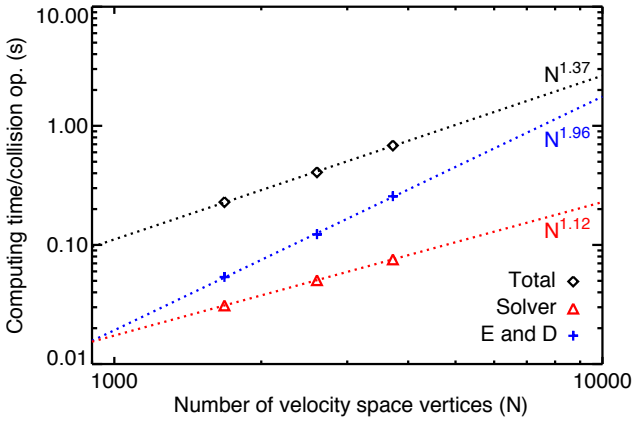

Figure 9: Scaling of the Fokker-Planck-Landau collision operator with the total number of vertices (N) of the velocity space mesh. The dotted lines are fits to the data points. For this test, the collision operator was solved on a single configuration space vertex with 8 OpenMP threads. As expected, the calculation of the drag and diffusion coefficients $\boldsymbol{E}$ and $\underline{\boldsymbol{D}}$ scales quadratically with N. However, the total computing time is not dominated completely by the calculation of the drag and diffusion coefficients for the tested grid sizes. Other operations that scale linearly in N such as the solver in the Picard iteration contribute significantly so that the overall scaling of the collision operator is O($N^{1.37}$) for practical grid sizes used in XGC1 and XGCa. For larger mesh sizes than the ones we tested, the scaling will be closer to N^2.

cost. Operations that scale linearly in N such as the solver contribute significantly to the total cost. This is illustrated in Fig. 9. The particle-related work is independent between particles and the number of particles is typically much larger than the size of the mesh, allowing many more threads to be used. When the cost of the particle-based work is significantly greater than that of the mesh-based work, these applications have demonstrated excellent scalability even beyond the thread count at which the mesh-based cost stops decreasing. (This scalability of the XGC applications is possible because of an effective and low cost particle load balancing scheme.)

However, the cost of the collision operator can be comparable to that of particle-related work in problems of interest, and it is critical that parallelism be exploited efficiently in its implementation. To establish this, we studied the performance characteristics of our collision operator with the gyrokinetic neoclassical particle-in-cell code XGCa.

4.1. Typical problem size

First, we discuss the typical problem size of simulations with XGC1 and XGCa. The meshes for the field solver in an XGC1 simulation need to resolve the gyroradius scale in the poloidal and minor radial direction. In the toroidal direction, at least 32 toroidal modes need to be resolved for accurate description of turbulence. These resolution requirements result in configuration space meshes with of the order of 10^6 vertices in typical XGC1 simulations. In XGCa, only axisymmetric perturbations of the plasma are considered, and the resolution requirements for the poloidal direction are less strict, which reduces the number of vertices in realistic cases by a factor of approximately 100 compared to XGC1.

Since the implementation of our collision operator in the full-f version of XGC1 required several hundred thousand particles per collision cell [9], the field solver mesh could not be used for the collision operator directly. Instead, a much coarser mesh had to be used for collisions. The hybrid total-δf method employed in XGC1 and XGCa reduces the required number of particles per collision cell significantly. In Sec. 3.2 we showed that 5000-10000 marker particles per collision cell are sufficient for accurate collisions. This allows us to use the same triangular mesh for the field solver and for the collision operator in XGC1 and XGCa. For comparison, we note that

the number of mesh cells in the XGC1 and XGCa meshes is approximately twice the number of mesh vertices. In order to resolve the fine velocity space structures of turbulence, XGC1 requires around 10000 particles per cell.

Thus, $O(10^8)$ particles are required for an XGCa simulation and $O(10^{10})$ particles are required for an XGC1 simulation. Typical production runs of XGCa use between 500 and 1000 compute nodes on the Edison computing system (Cray XC30, NERSC); production runs of XGC1 occupy usually 8192-16384 compute nodes on the Titan system (Cray XK7, ORNL).

4.2. Domain decomposition and load balancing

While the details of the parallelization techniques applied in the XGC codes are beyond the scope of this article and will be presented elsewhere, it is necessary to explain the basic aspects of domain decomposition and load balancing here. The simulation domain in the XGC codes in configuration space is a topological torus. In the toroidal direction, this torus is decomposed into N_φ segments and each segment is assigned to a set of MPI processes. The sources of the electric and magnetic fields (charge density and current) are evaluated through particle-mesh interpolation on triangular meshes on the poloidal cross sections at the segment boundaries. The triangular meshes used for the XGC codes are (approximately) field-aligned meshes with optimizations applied around the X-point of the magnetic separatrix. From the magnetic axis to a (arbitrary) flux-surface in the open field-line region, the mesh is (mostly) flux-surface aligned and one-element-deep. Between the outmost flux-surface of the mesh and the wall of the vacuum vessel, the mesh is truly unstructured. A detailed description of these meshes, which are identical in each segment, can be found in Refs.

[22, 23].

The vertices of the solver mesh in each segment are then distributed among the MPI processes assigned to that segment. The decomposition of the solver mesh is also identical in each segment. Each MPI process handles the distribution function data of the mesh vertices assigned to it as well as the data of the marker particles located in the patch of the mesh assigned to the process. Each MPI process evaluates the collision operator in its patch of the mesh and the motion of the particles therein, which facilitates data locality.

In case of XGCa, the plasma distribution functions are axisymmetric, which allows for another level of MPI parallelization of the collision operator. Due to the fact that the solver mesh and its decomposition is identical in each toroidal segment, each MPI process needs to evaluate the collision operator only on part of the mesh vertices of the patch assigned to the process. After each process assigned to the same patch finishes the evaluation of the collision operator on its part of the patch, results are gathered.

While each toroidal segment is assigned the same number of processes, the decomposition of the solver mesh on the poloidal cross sections is dynamic and needs to be optimized with the goal of a balanced workload on all processes. Assuming for the moment that the computing time needed for the collision operation is the same for each collision cell (i.e. mesh vertex), the performance of the collision operator is likely to be optimal if collision cells are distributed evenly among the MPI processes available to the application. However, although the collision operator can be expected to need a considerable fraction of the computing time for one simulation, the evaluation of particle motion (particle push) is often the dominant cost

in realistic use cases. The performance of the particle push is optimal if the simulation domain is decomposed such that each MPI process is assigned the same amount of marker particles. Unfortunately, the density of marker particles in configuration space is not uniform most of the time.

Hence, a domain decomposition that provides optimal load balance for the collision operator will exhibit load imbalance in the particle push step and vice versa. Therefore, we had do develop a flexible dynamic domain decomposition algorithm to find the optimal balance between collision operator and particle push. This turned out to be challenging because the number of mesh vertices assigned to an MPI process is not always a reliable measure for the actual computing time needed to evaluate the collision operator. Depending on the local physical parameters like the collision frequency, the Picard iteration used for implicit time integration converges faster on some vertices than on others. In contrast, the load imbalance in the particle distribution is a reliable predictor of the load imbalance in the cost of particle-related work. Therefore, we measure both the actual time spent in the collision operator on each vertex and the actual total run time per time step and use this to determine how much particle imbalance can be accepted when optimizing the mesh decomposition in regular intervals.

4.3. Nested OpenMP parallelism

The XGC codes use mixed MPI and OpenMP parallelization. The OpenMP threads available to each MPI process can be used to accelerate the evaluation of the collision operator on two levels. In the outer level, threads can evaluate the collision operator on several of the mesh vertices assigned to their host MPI process in parallel. In the inner level, threads

can be used to accelerate the evaluation of the collision operator for each collision cell assigned to their host process while collision cells are processed sequentially. The inner level parallelization is somewhat less efficient than the outer in that, for the inner level, the whole collision operator is not parallelized. Only the most computationally expensive loops are threaded, leaving a small but measurable, "serial fraction". However, the outer level parallelization has the drawback of requiring significantly more memory for storing the coefficients U_{xx} appearing in Eq. (12) (approximately 130 MB with 41×41 velocity grid points). The inner level parallelization, on the other hand, does not require additional memory. On computers with little memory per process like Mira (BlueGene Q, ANL), outer level parallelization quickly uses up the available memory. Moreover, nested parallelism, using both outer and inner level parallelization, can help in situations in which more parallel threads are available than there are mesh vertices.

Therefore, the ability to switch between the two OpenMP parallelization approaches described above benefits the portability and performance of the XGC codes. By using nested OpenMP parallelism, available OpenMP threads can be assigned flexibly to the inner and outer level parallelization to optimize the performance of the collision operator based on the specifics, e.g. available memory, of individual HPC systems, compute node counts, and problem sizes.

We tested the efficiency of different thread setups on Edison (two 12-core CPUs per compute node, two hardware threads per core, 1.3GB memory per thread), Titan (one 16-core CPU per compute node, 1GB memory per core), and Mira (one 16-core CPU per compute node, four hardware threads per core, 256 MB memory per thread). The problem size for this test is

$7.2 \cdot 10^7$ marker particles (with $m_e = m_i/100$) on Edison and Titan, and $10.0 \cdot 10^7$ particles on Mira. The configuration space mesh has 20694 vertices and the velocity grid 41×41 grid points. We ran the code for 100 time steps with the collision operator being evaluated every time step. On Edison, we used 100 compute nodes and each compute node hosted 2 MPI processes with 24 OpenMP threads each; on Titan, we used 300 compute nodes with 2 MPI processes per compute node and 8 OpenMP threads per process, and on Mira we used 1024 compute nodes. Note that the Edison and Titan experiments use the same total number of threads (4800), while the Mira experiments used 65536 threads. We have not yet been able to get nested OpenMP parallelism to work in the collision operator on Mira. So, instead of changing the partition of threads in the two levels of OpenMP parallelism, we tested with 1 (64), 2 (32), 4 (16), 8 (8), and 16 (4) MPI processes (OpenMP threads) and assigned all available threads to the inner level parallelization. By increasing the number of MPI processes per compute node, one also increases the number of collision cells that are processed in parallel on a compute node.

The results of this test are shown in Fig. 10. In general, nested OpenMP parallelism works very well because it allows the code to reduce memory use while preserving the overall performance of the collision operator. On Edison, the slowest thread configuration is only approximately 50% slower than the fastest. On Titan, the slowest configuration is around 20% slower than the fastest one. On Mira, the best performance is observed with 4 and 8 MPI processes. When we tried to increase the number of MPI processes per compute node above 16, the system ran out of memory, which demonstrates the scarcity of memory on this type of system. Each of the 64 hardware

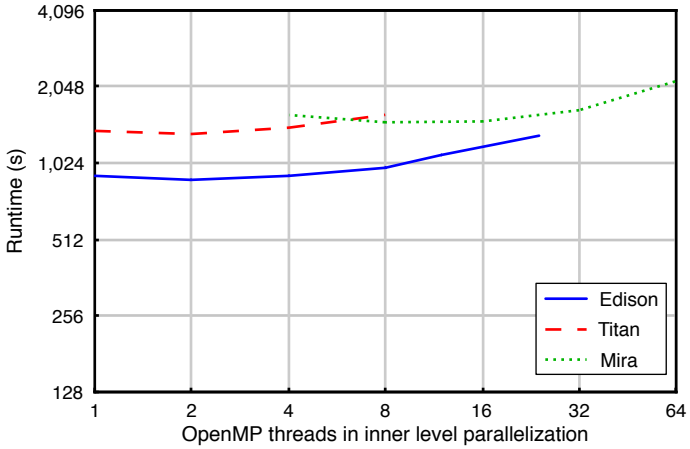

Figure 10: Efficiency of nested OpenMP parallelism in the collision operator in XGCa simulations. The number of OpenMP threads in the inner level OpenMP parallelization of the collision operator was varied. On Edison, the test was run on 100 compute nodes each with 2 MPI processes and 24 OpenMP threads per process. On Titan, 300 compute nodes with 2 MPI processes and 8 OpenMP threads per node were used. On Mira, the test ran on 1024 compute nodes with full hyperthreading (64 hardware threads per node). Since Mira does not support nested OpenMP parallelism, the number of MPI processes per compute node was varied instead with all OpenMP threads used in the inner level OpenMP parallelization.

threads has only 256 MB memory at its disposal. With a velocity grid of 41×41 grid points, each instance of the collision operator alone requires approximately 130 MB of memory.

4.4. Parallelization efficiency

As mentioned in the introduction to this section, the parallel efficiency of approximately linear complexity mesh-based numerical methods necessarily deteriorates when the number of parallel threads becomes comparable to or larger than the number of mesh vertices. The FPL collision operator discussed here is no exception. Therefore, we investigated whether the scalability of XGCa code is affected by the collision operator. We conducted three scaling studies. In each of them, we ran XGCa for 100 time steps with the collision operator being evaluated every time step.

First, we varied the number of marker particles, while keeping the number of compute nodes and the number of mesh vertices fixed. On Edison, we used 256 compute nodes (2 MPI tasks/node, 24 OpenMP threads/process) with a configuration space mesh of 20694 vertices and a velocity grid of 41×41 grid points. The number of marker particles per hardware thread (48 hardware threads/compute node) varied from 2048 to 131072, corresponding to a total of $25 \cdot 10^6$ to $1.6 \cdot 10^9$ marker particles. A total of $2 \cdot 10^8$ markers or approximately 16000 markers per hardware thread would be realistic for this case. We ran the simulations with enhanced electron mass, $m_e = m_i/100$, and realistic electron mass, which increases the computing time spent on the particle motion by a factor of 4. The results of this particle scaling study are shown in Fig. 11. When the number of marker particles is small, the total computing time is dominated by the collision operator and

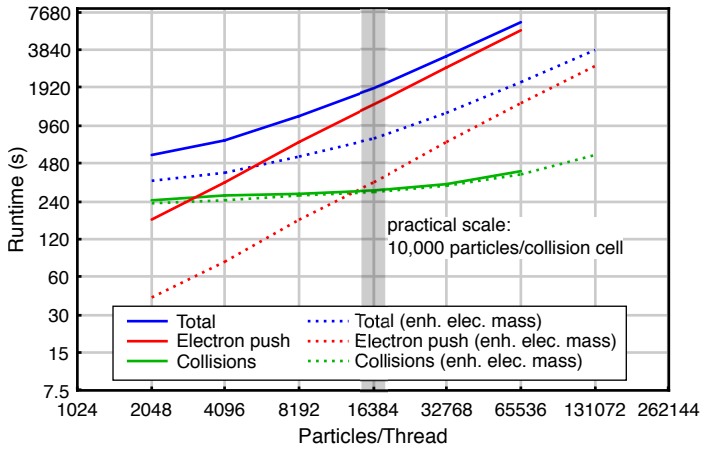

Figure 11: Scaling of XGCa on 256 Edison compute nodes with 2 MPI processes and 24 OpenMP threads per process nodes. The size of the configuration space mesh is fixed at 20694 vertices but the particle count was varied. There are two distinct performance regimes, the collision-dominated and the particle dominated regime. In most realistic cases, XGCa operates in the particle dominated regime.

adding more particles increases the total run time only slightly. With increasing particle count, a transition occurs between the collision-dominated performance regime and the particle-dominated regime, in which the total run time is determined by the computing time spent on the particle push. The results of the particle scaling study suggest that XGCa benefits from the favorable scalability of the particle-in-cell technique when operated in the particle-dominated regime.

We also ran a strong scaling study on Edison using the same mesh as before. In order to relate the strong scaling test to the particle scaling test, we performed two series of simulations, one with $m_e = m_i/100$ and 10^8 marker particles, and the other with realistic electron mass and $2 \cdot 10^8$ particles. The latter is a realistic use case in which the cost for the particle push is dominant. We scaled the resources used on this problem from 16

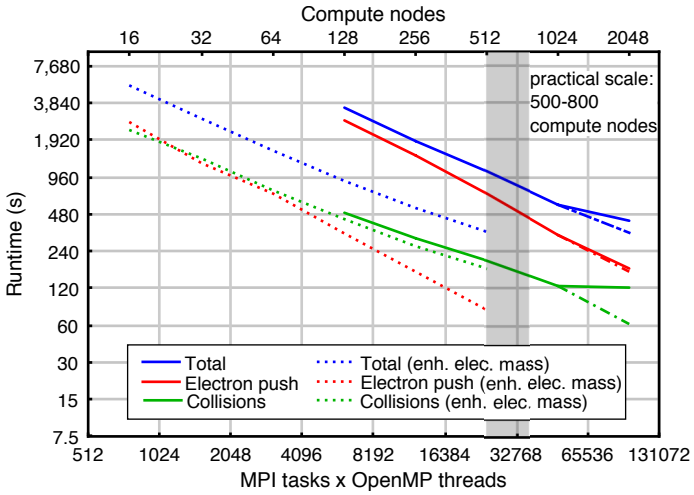

Figure 12: Strong scaling of XGCa on Edison with a configuration space mesh of 20694 vertices. The total number of particles is 10^8 with enhanced electron mass $m_e = m_i/100$ (performance dominated by collisions), and $2 \cdot 10^8$ with realistic electron mass (performance dominated by particles). Each compute node hosted 2 MPI processes with 24 OpenMP threads per process; all OpenMP threads are used in the outer level OpenMP parallelization. Using only 6 threads in the outer and 4 in the inner level OpenMP parallelization improves performance when the number of parallel threads is larger than the number of configuration space vertices (dashed-dotted lines).

compute nodes on Edison to 2048 nodes, which corresponds to 37% of the whole system. The results of the strong scaling study are shown in Fig. 12. In case of the smaller problem with enhanced electron mass, the computing time spent on the collision operator and the particle push is similar. Performance degradation sets in when the product of MPI processes and OpenMP threads (i.e. the total number of hardware threads) is around 20% of the mesh size. The realistic use case, on the other hand scales well with all OpenMP threads in the outer level OpenMP parallelization up to 1024 compute nodes. Then, the computing time spent on collisions stagnates at

a level comparable to the time spent on the particle push. At this point, a significant number of parallel threads is idle during the evaluation of the collision operator. Using the idle threads in the inner level parallelization improves the performance of the collision operator significantly. We verified this by repeating the test on 2048 compute nodes with 6 OpenMP threads in the outer and 4 in the inner level parallelization of the collision operator. A typical production run for the problem size discussed here would use between 500 and 800 compute nodes on Edison.

Finally, we performed a weak scaling study with XGCa on Edison with 10000 particles per mesh vertex. The smallest problem in this study used a mesh with 10188 vertices and was run on 128 compute nodes. The larger problems used meshes with 22109, 40197, and 81936 vertices, and were run on 256, 512, and 1024 compute nodes. The velocity grid in all cases had 41×41 grid points. We tested the weak scaling with enhanced ($m_e = m_i/100$) and realistic electron mass. In the latter case, the particle push was responsible for most of the total run time. Figure 13 summarizes the results. While there is only little performance degradation ($\sim 10\%$) with realistic electron mass, the largest case case with 81936 mesh vertices and enhanced electron mass is 24% slower than the smallest case.

From the these tests, we conclude that realistic use cases of XGCa are well within the operating range in which the code's scaling is determined by the more scalable particle-in-cell part and not by the mesh limited scaling of the collision operator. Extrapolating the performance data obtained with XGCa to a large XGC1 simulation with approximately 10^6 mesh vertices and $2 \cdot 10^{10}$ marker particles on 16384 compute nodes on Titan, we also expect XGC1 to be operating in the scalable operating range. Detailed

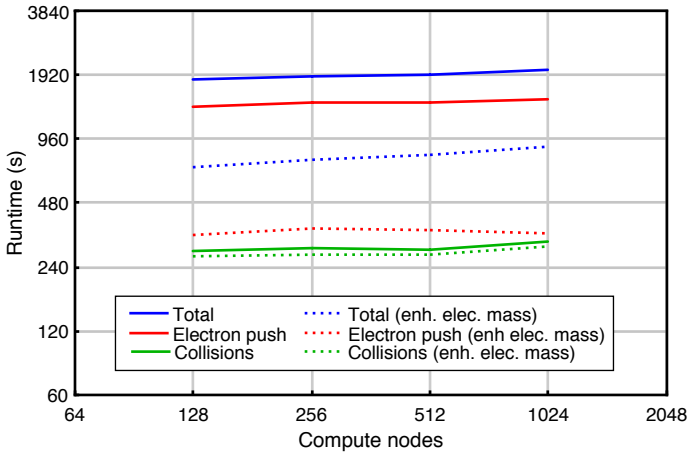

Figure 13: Weak scaling of XGCa on Edison with configuration space meshes of 10188, 22109, 40197 and 81936 vertices, 10000 particles per vertex with enhanced electron mass ($m_e = m_i/100$) and realistic electron mass. With realistic electron mass, the computing time is dominated by the particle push and performance degrades less with increasing number of compute nodes than with enhanced electron mass.

performance studies using XGC1 are currently being carried out and will be published elsewhere.

5. Summary and Conclusions

We generalized the single-species Fokker-Planck-Landau collision operator developed by Yoon and Chang [9] to a multiple-species formulation. Since accuracy is essential for a collision operator, we thoroughly investigated the conservation laws for mass, parallel momentum, and energy. Specifically, we demonstrated that the Landau form of the Fokker-Planck operator has the favorable property that the discrete velocity space meshes do not have to be identical for all species because conservation laws apply independently for each velocity pair $(\boldsymbol{v}, \boldsymbol{v}')$. Therefore, the only prerequi-

site for the discrete velocity grids is that the grids cover those velocities at which the corresponding distribution function is not negligibly small. Compared to the RMJ form of the Fokker-Planck operator, for which the use of different velocity grids requires special treatment [7], this is an important simplification. The RMJ operator, on the other hand, is likely to show better scaling behavior when varying the size of the velocity grid due to the representation of the drag and diffusion coefficients with Rosenbluth potentials. However, due to the relatively small velocity space meshes used in XGC1 and XGCa simulations, the $O(N^2)$ scaling of the calculation of the drag and diffusion coefficients may only impact the overall performance of the XGC codes, if much larger velocity grids ($N \gtrsim 10000$) need to be used (see Fig. 9).

Moreover, we proved that the continuum conservation laws of the FPL operator are exact in the discretized operator. Therefore, the Picard iteration used for the implicit time advance implemented in the XGC codes will always converge to a distribution function that conserves mass, momentum and energy to the desired accuracy without requiring additional complex numerical measures.

In various tests against neoclassical theory and other neoclassical codes, we verified our implementation of the FPL operator. We demonstrated that residual conservation errors can be controlled by adjusting the convergence criterion in the Picard iteration of the implicit time integrator and by the number of marker particles.

Our implementation of the FPL operator is intended for use in extreme-scale high-performance computing applications, where the mixing of PIC and mesh based code modules may limit scalability when the number of

parallel threads becomes comparable to the size of the mesh (the configuration space mesh in the present case). Therefore, we studied scalability for realistic problem sizes with the code XGCa. According to the results of our scaling studies, it is unlikely in realistic cases that the generally good scalability of the PIC part of the XGC codes is diminished significantly by the introduction of the mesh based collision operator. To achieve this good scalability, we needed to implement nested OpenMP parallelism as well as special load balancing algorithms that are able to balance the workload of PIC and mesh components of the XGC codes.

Acknowledgements

Support for this work was provided through the Scientific Discovery through Advanced Computing (SciDAC) program funded by the U.S. Department of Energy Office of Advanced Scientific Computing Research and the Office of Fusion Energy Sciences. The work was performed at Princeton Plasma Physics Laboratory, which is managed by Princeton University under Contract No. DE- AC02-09CH11466, at Oak Ridge National Laboratory, which is managed by UT-Battelle, LLC under Contract No. DE-AC05-00OR22725, and at Rensselaer Polytechnic Institute under Contract No. DE-SC0008449. Awards of computer time was provided by the Innovative and Novel Computational Impact on Theory and Experiment (INCITE) program. This research used resources of the Argonne Leadership Computing Facility, which is a DOE Office of Science User Facility supported under contract DE-AC02-06CH11357. This research also used resources of the Oak Ridge Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC05-00OR22725.

This research also used resources of the National Energy Research Scientific Computing Center, a DOE Office of Science User Facility supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

[1] S. P. Hirshman, D. J. Sigmar, Approximate Fokker-Planck collision operator for transport theory applications, Physics of Fluids (1958-1988) 19 (10) (1976) 1532–1540. doi:http://dx.doi.org/10.1063/1.861356.

[2] A. H. Boozer, G. Kuo-Petravic, Monte Carlo evaluation of transport coefficients, Physics of Fluids (1958-1988) 24 (5) (1981) 851–859. doi:http://dx.doi.org/10.1063/1.863445.

[3] W. X. Wang, N. Nakajima, M. Okamoto, S. Murakami, A new delta-f method for neoclassical transport studies, Plasma Physics and Controlled Fusion 41 (9) (1999) 1091.

[4] E. A. Belli, J. Candy, Full linearized Fokker-Planck collisions in neoclassical transport simulations, Plasma Physics and Controlled Fusion 54 (1) (2012) 015015. doi:http://stacks.iop.org/0741-3335/54/i=1/a=015015.

[5] C. Buet, K.-C. Le Thanh, About positive, energy conservative and equilibrium state preserving schemes for the isotropic Fokker-Planck-Landau equation (Dec 2006).
URL `https://hal.archives-ouvertes.fr/hal-00092543`

[6] C. Buet, K.-C. Le Thanh, Positive, conservative, equilibrium state preserving and implicit difference schemes for the isotropic Fokker-Planck-Landau equation (May 2007).
URL `https://hal.archives-ouvertes.fr/hal-00142408`

[7] W. Taitano, L. Chacón, A. Simakov, K. Molvig, A mass, momentum, and energy conserving, fully implicit, scalable algorithm for the multi-dimensional, multi-species Rosenbluth-Fokker-Planck equation, Journal of Computational Physics 297 (2015) 357 – 380. doi:http://dx.doi.org/10.1016/j.jcp.2015.05.025.

[8] A. Pataki, L. Greengard, Fast elliptic solvers in cylindrical coordinates and the Coulomb collision operator, Journal of Computational Physics 230 (21) (2011) 7840 – 7852. doi:http://dx.doi.org/10.1016/j.jcp.2011.07.005.

[9] E. S. Yoon, C. S. Chang, A Fokker-Planck-Landau collision equation solver on two-dimensional velocity grid and its application to particle-in-cell simulation, Physics of Plasmas 21 (3) (2014) 032503. doi:http://dx.doi.org/10.1063/1.4867359.

[10] S. Ku, C. Chang, P. Diamond, Full-f gyrokinetic particle simulation of centrally heated global ITG turbulence from magnetic axis to edge pedestal top in a realistic tokamak geometry, Nuclear Fusion 49 (11) (2009) 115021.

[11] R. Hager, C. S. Chang, Gyrokinetic neoclassical study of the bootstrap current in the tokamak edge pedestal with fully non-linear coulomb collisions, Physics of Plasmas 23 (4) (2016) 000000. doi:???

[12] M. N. Rosenbluth, W. M. MacDonald, D. L. Judd, Fokker-Planck equation for an inverse-square force, Phys. Rev. 107 (1957) 1–6. doi:10.1103/PhysRev.107.1.

[13] W. T. Taitano, L. Chacón, A. N. Simakov, private communication.

[14] R. J. Goldston, P. H. Rutherford, Introduction to Plasma Physics, 1st Edition, Taylor and Francis, New York, 1995.

[15] O. Larroche, An efficient explicit numerical scheme for diffusion-type equations with a highly inhomogeneous and highly anisotropic diffusion tensor, Journal of Computational Physics 223 (1) (2007) 436 – 450. doi:http://dx.doi.org/10.1016/j.jcp.2006.09.016.

[16] B. Peigney, O. Larroche, V. Tikhonchuk, Fokker-Planck kinetic modeling of suprathermal alpha-particles in a fusion plasma, Journal of Computational Physics 278 (2014) 416 – 444. doi:http://dx.doi.org/10.1016/j.jcp.2014.08.033.

[17] E. Epperlein, G. Rickard, A. Bell, A code for the solution of the Vlasov-Fokker-Planck equation in 1-D or 2-D, Computer Physics Communications 52 (1) (1988) 7 – 13. doi:http://dx.doi.org/10.1016/0010-4655(88)90165-8.

[18] E. Epperlein, Fokker-Planck modeling of electron transport in laser-produced plasmas, Laser and Particle Beams 12 (1994) 257–272. doi:10.1017/S0263034600007722.

[19] J. D. Huba, NRL Plasma Formulary, Naval Research Laboratory, Washington, DC, 2013.

[20] F. Hinton, R. Hazeltine, Theory of plasma transport in toroidal confinement systems, Rev. Mod. Phys. 48 (1976) 239–308. doi:10.1103/RevModPhys.48.239.

[21] E. A. Belli, J. Candy, Kinetic calculation of neoclassical transport including self-

consistent electron and impurity dynamics, Plasma Physics and Controlled Fusion 50 (9) (2008) 095010. doi:http://stacks.iop.org/0741-3335/50/i=9/a=095010.

[22] M. F. Adams, S.-H. Ku, P. Worley, E. D'Azevedo, J. C. Cummings, C. Chang, Scaling to 150k cores: Recent algorithm and performance engineering developments enabling XGC1 to run at scale, in: Journal of Physics: Conference Series, Vol. 180, IOP Publishing, 2009, p. 012036.

[23] F. Zhang, R. Hager, S.-H. Ku, C.-S. Chang, S. C. Jardin, N. M. Ferraro, E. S. Seol, E. Yoon, M. S. Shephard, Mesh generation for confined fusion plasma simulation, Engineering with Computers 32 (2) (2016) 285–293. doi:10.1007/s00366-015-0417-y.

PPPL PRINCETON PLASMA PHYSICS LABORATORY

Princeton Plasma Physics Laboratory
Office of Reports and Publications

Managed by
Princeton University

under contract with the
U.S. Department of Energy
(DE-AC02-09CH11466)

P.O. Box 451, Princeton, NJ 08543
Phone: 609-243-2245
Fax: 609-243-2751

E-mail: publications@pppl.gov

Website: http://www.pppl.gov