GKV manual

Shinya Maeyama

Dec 23, 2025

TABLE OF CONTENTS:

CHAPTER

ONE

FORMULATION

* Blue-colored sentences are physical assumptions used in GKV [1-1]. This manual is based on the GKV version gkvp_f0.65.

1.1 Governing equations

One derives gyrokinetic equations based on the following gyrokinetic ordering [1-2],

$$\frac{\tilde{f}}{F} \sim \frac{e\tilde{\phi}}{T} \sim \frac{\tilde{B}}{B} \sim \frac{k_\parallel}{k_\perp} \sim \frac{\omega}{\Omega} \equiv \delta \ll 1. \tag{1.1}$$

GKV follows δf gyrokinetics, where distribution functions are split into equilibrium and perturbed parts $\mathcal{F} = F + \tilde{f}$. Additionally, there are some subsidiary assumptions:

- separation of the equilibrium and perturbed scale lengths $|\nabla F|/F \ll |\nabla \tilde{f}|/f$ decouples neoclassical physics from turbulent dynamics and treats flute-type perturbations
- low β value justifies neglect of compressional magnetosonic waves $\tilde{B}_\parallel$ and higher-order correction in β, but retains shear Alfvénic dynamics $\tilde{A}_\parallel$
- the present version of GKV treat equilibrium $\boldsymbol{E} \times \boldsymbol{B}$ flow shear effect.
- the equilibrium distribution function is to be a local Maxwellian $F = F_\mathrm{M} = n\left(\frac{m}{2\pi T}\right)^{\frac{3}{2}} e^{-\frac{mv_\parallel^2}{2T} - \frac{\mu B}{T}}$
- the equilibrium magnetic field satisfies the MHD equilibrium $\nabla P = \boldsymbol{J} \times \boldsymbol{B}$

Then, the δf gyrokinetic Vlasov-Poisson-Ampère equations are

$$\begin{aligned}
&\frac{\partial \tilde{f}_\mathrm{s}}{\partial t} + \left(\boldsymbol{V}_E + v_\parallel \frac{\boldsymbol{B} + \tilde{\boldsymbol{B}}_\perp}{B} + \tilde{\boldsymbol{v}}_\mathrm{E} + \boldsymbol{v}_\mathrm{sG} + \boldsymbol{v}_\mathrm{sC}\right) \cdot \nabla \left(\tilde{f}_\mathrm{s} + \frac{e_\mathrm{s} F_\mathrm{sM}}{T_\mathrm{s}} J_{0\mathrm{s}} \tilde{\phi}\right) \\
&- \frac{\mu \nabla_\parallel B}{m_\mathrm{s}} \frac{\partial}{\partial v_\parallel} \left(\tilde{f}_\mathrm{s} + \frac{e_\mathrm{s} F_\mathrm{sM}}{T_\mathrm{s}} J_{0\mathrm{s}} \tilde{\phi}\right) \\
&+ \frac{e_\mathrm{s} F_\mathrm{sM}}{T_\mathrm{s}} \left[v_\parallel \frac{\partial J_{0\mathrm{s}} \tilde{A}_\parallel}{\partial t} - \boldsymbol{v}_{\mathrm{s}*} \cdot \nabla J_{0\mathrm{s}} (\tilde{\phi} - v_\parallel \tilde{A}_\parallel)\right] = C_\mathrm{s},
\end{aligned} \tag{1.2}$$

$$\left[\nabla_\perp^2 - \frac{1}{\varepsilon_0} \sum_\mathrm{s} \frac{e_\mathrm{s}^2 n_\mathrm{s}}{T_\mathrm{s}} (1 - \Gamma_{0\mathrm{s}})\right] \tilde{\phi} = -\frac{1}{\varepsilon_0} \sum_\mathrm{s} e_\mathrm{s} \int dv^3 J_{0\mathrm{s}} \tilde{f}_\mathrm{s}, \tag{1.3}$$

$$\nabla_\perp^2 \tilde{A}_\parallel = -\mu_0 \sum_\mathrm{s} e_\mathrm{s} \int dv^3 J_{0\mathrm{s}} v_\parallel \tilde{f}_\mathrm{s}, \tag{1.4}$$

where the gyrophase-average operators $J_{0\mathrm{s}} = \oint (d\xi/2\pi) e^{\boldsymbol{\rho}_\mathrm{s} \cdot \nabla} = \oint (d\xi/2\pi) e^{-\boldsymbol{\rho}_\mathrm{s} \cdot \nabla}$ and $\Gamma_{0\mathrm{s}} = \int dv^3 (F_\mathrm{sM}/n_\mathrm{s}) J_{0\mathrm{s}}^2$ are used with the gyroradius vector $\boldsymbol{\rho}_\mathrm{s} = \boldsymbol{b} \times m_\mathrm{s} \boldsymbol{v}/(e_\mathrm{s} B)$. $\boldsymbol{V}_E = \boldsymbol{b} \times \nabla \Phi / B$ denotes the equilibrium $\boldsymbol{E} \times \boldsymbol{B}$ flow. The

perturbed electric and magnetic fields are $\tilde{\boldsymbol{E}} = -\nabla(J_{0\mathrm{s}}\tilde{\phi}) - \boldsymbol{b}\partial\tilde{A}_\parallel/\partial t$ and $\tilde{\boldsymbol{B}}_\perp = \nabla(J_{0\mathrm{s}}\tilde{A}_\parallel)\times\boldsymbol{b}$. The perturbed $\boldsymbol{E}\times\boldsymbol{B}$, grad-B, curvature, diamagnetic drift velocities are respectively given by $\tilde{\boldsymbol{v}}_\mathrm{E} = \boldsymbol{b}\times\nabla(J_{0\mathrm{s}}\tilde{\phi})/B$, $\boldsymbol{v}_\mathrm{sG} = \boldsymbol{b}\times\mu\nabla B/(e_\mathrm{s}B)$, $\boldsymbol{v}_\mathrm{sC} = \boldsymbol{b}\times m_\mathrm{s}v_\parallel^2\boldsymbol{b}\cdot\nabla\boldsymbol{b}/(e_\mathrm{s}B)$ and $\boldsymbol{v}_{\mathrm{s}*} = \boldsymbol{b}\times[T_\mathrm{s}\nabla\ln n_\mathrm{s} + (m_\mathrm{s}v_\parallel^2/2 + \mu B - 3T_\mathrm{s}/2)\nabla\ln T_\mathrm{s}]/(e_\mathrm{s}B)$. C_s is the linearized collision term on the species s and will be explained in Section *Collision operator*. The nonlinear term in the Vlasov eq. (denoted $\mathcal{N}_\mathrm{s}$ below), which originates from $\boldsymbol{E}\times\boldsymbol{B}$ and $v_\parallel\tilde{\boldsymbol{B}}_\perp/B$ advections of $\tilde{f}$ and $\tilde{\boldsymbol{E}}\cdot\tilde{\boldsymbol{B}}_\perp$ acceleration of F, can be rewritten as,

$$\begin{aligned}\mathcal{N}_\mathrm{s} &= \left(\tilde{\boldsymbol{v}}_\mathrm{E} + v_\parallel\frac{\tilde{\boldsymbol{B}}_\perp}{B}\right)\cdot\nabla\tilde{f}_\mathrm{s} + \frac{e_\mathrm{s}\tilde{\boldsymbol{E}}}{m_\mathrm{s}}\cdot\frac{\tilde{\boldsymbol{B}}_\perp}{B}\left(-\frac{m_\mathrm{s}v_\parallel}{T_\mathrm{s}}F_{\mathrm{M}\mathrm{s}}\right)\\ &= \frac{\boldsymbol{b}}{B}\cdot\nabla\left(J_{0\mathrm{s}}\tilde{\phi} - v_\parallel J_{0\mathrm{s}}\tilde{A}_\parallel\right)\times\nabla\left(\tilde{f}_\mathrm{s} + \frac{e_\mathrm{s}F_{\mathrm{M}\mathrm{s}}}{T_\mathrm{s}}J_{0\mathrm{s}}\tilde{\phi}\right),\end{aligned} \tag{1.5}$$

respectively.

1.2 Geometry and coordinates

The following explanation is conventional flux-tube model without equilibrium flow [1-3]. For the numerical treatment of equilibrium flow shear effects in the rotating flux-tube model, please refer to [1-4].

When an equilibrium magnetic field is known, one can construct a flux coordinate $(\rho_f, \theta_f, \varphi_f)$ such that,

$$\boldsymbol{B} = \nabla\Psi_p(\rho_f)\times\nabla[q(\rho_f)\theta_f - \varphi_f], \tag{1.6}$$

where we use the safety factor $q(\rho_f) = d\Psi_t/d\Psi_p$ and the toroidal and poloidal flux $\Psi_p(\rho_f)$ and $\Psi_t(\rho_f)$. GKV employs Clebsch-type coordinate as

$$\begin{aligned}x &= c_x(\rho_f - \rho_{f0}),\\ y &= c_y[q(\rho_f)\theta_f - \varphi_f],\\ z &= \theta_f,\end{aligned} \tag{1.7}$$

where ρ_{f0}, c_x and c_y are constant. We refer (x, y, z) as the radial, field-line-label, and field-aligned coordinates, respectively. Using this GKV coordinates, the equilibrium magnetic field is represented by

$$\boldsymbol{B} = c_b\nabla x\times\nabla y = \frac{c_b}{\sqrt{g}}\frac{\partial\boldsymbol{r}}{\partial z}, \tag{1.8}$$

where $c_b = (d\Psi_p/d\rho_f)/(c_xc_y)$ and $\sqrt{g} = (\nabla x\cdot\nabla y\times\nabla z)^{-1}$.

Simulation domain of GKV is based on the local flux-tube model [1-3]. Using flute approximation for perturbed quantities $k_\perp \gg k_\parallel$ (consistent with the gyrokinetic ordering Eq. (1.1), vector differential operators in gyrokinetic

Eqs. (1.2)–(1.4) become

$$
\begin{aligned}
\nabla_\parallel \tilde{f} &= \boldsymbol{b}\cdot\nabla\tilde{f} = \frac{c_b}{B\sqrt{g}}\frac{\partial \tilde{f}}{\partial z}, \\
\nabla^2\tilde{f} &= \frac{1}{\sqrt{g}}\frac{\partial}{\partial r^i}\left[\sqrt{g}\left(\frac{\partial\tilde{f}}{\partial r^j}\nabla r^j\right)\cdot\nabla r^i\right] \\
&\simeq g^{xx}\frac{\partial^2\tilde{f}}{\partial x^2} + 2g^{xy}\frac{\partial^2\tilde{f}}{\partial x\partial y} + g^{yy}\frac{\partial^2\tilde{f}}{\partial y^2}, \\
\boldsymbol{b}\times\nabla\tilde{h}\cdot\nabla\tilde{f} &= \boldsymbol{b}\cdot\left(\frac{\partial\tilde{h}}{\partial r^i}\nabla r^i\times\frac{\partial\tilde{f}}{\partial r^j}\nabla r^j\right) \\
&\simeq \frac{B}{c_b}\left(\frac{\partial\tilde{h}}{\partial x}\frac{\partial\tilde{f}}{\partial y} - \frac{\partial\tilde{h}}{\partial y}\frac{\partial\tilde{f}}{\partial x}\right), \\
\boldsymbol{b}\times\nabla H\cdot\nabla\tilde{f} &\simeq \frac{B}{c_b}\left(\frac{\partial H}{\partial x}\frac{\partial\tilde{f}}{\partial y} - \frac{\partial H}{\partial y}\frac{\partial\tilde{f}}{\partial x}\right) \\
&\quad + \frac{\partial H}{\partial z}\left(\frac{g^{xz}g^{yx}-g^{xx}g^{yz}}{B/c_b}\frac{\partial\tilde{f}}{\partial x} + \frac{g^{xz}g^{yy}-g^{xy}g^{yz}}{B/c_b}\frac{\partial\tilde{f}}{\partial y}\right),
\end{aligned}
\tag{1.9}
$$

where $g^{ij} = \nabla r^i\cdot\nabla r^j$ denotes the metric tensor.

Since the magnetic curvature can be replaced by

$$
\boldsymbol{b}\cdot\nabla\boldsymbol{b} = \frac{\nabla_\perp B}{B} + \frac{\nabla P}{B^2/\mu_0}, \tag{1.10}
$$

when the equilibrium satisfies the MHD equilibrium, $\nabla P = \boldsymbol{J}\times\boldsymbol{B}$ and $\nabla\times\boldsymbol{B} = \mu_0\boldsymbol{J}$,

the magnetic (i.e., grad-B and curvature) drift velocity is given by

$$
\boldsymbol{v}_{\mathrm{sG}} + \boldsymbol{v}_{\mathrm{sC}} = \frac{1}{e_{\mathrm{s}}B}\boldsymbol{b}\times\left(\frac{m_{\mathrm{s}}v_\parallel^2+\mu B}{B}\nabla B + \frac{m_{\mathrm{s}}v_\parallel^2}{B^2/\mu_0}\nabla P\right), \tag{1.11}
$$

and then the magnetic and diamagnetic drift terms are

$$
\begin{aligned}
(\boldsymbol{v}_{\mathrm{sG}} + \boldsymbol{v}_{\mathrm{sC}})\cdot\nabla(J_0\tilde{\phi}) =& \frac{m_{\mathrm{s}}v_\parallel^2+\mu B}{e_{\mathrm{s}}c_b}\left(K_x\frac{\partial J_0\tilde{\phi}}{\partial x} + K_y\frac{\partial J_0\tilde{\phi}}{\partial y}\right) \\
&+ \frac{m_{\mathrm{s}}v_\parallel^2}{e_{\mathrm{s}}c_b}\frac{dP/dx}{B^2/\mu_0}\frac{\partial J_0\tilde{\phi}}{\partial y}, \\
\boldsymbol{v}_{\mathrm{s}*}\cdot\nabla(J_0\tilde{\phi}) =& -\frac{T_{\mathrm{s}}}{e_{\mathrm{s}}c_b}\left[\frac{1}{L_{n\mathrm{s}}} + \left(\frac{m_{\mathrm{s}}v_\parallel^2}{2T_{\mathrm{s}}} + \frac{\mu B}{T_{\mathrm{s}}} - \frac{3}{2}\right)\frac{1}{L_{T\mathrm{s}}}\right]\frac{\partial J_0\tilde{\phi}}{\partial y},
\end{aligned}
\tag{1.12}
$$

where

$$
K_x = -\frac{\partial\ln B}{\partial y} + \frac{g^{xz}g^{yx}-g^{xx}g^{yz}}{B^2/c_b^2}\frac{\partial\ln B}{\partial z}, \tag{1.13}
$$

$$
K_y = \frac{\partial\ln B}{\partial x} + \frac{g^{xz}g^{yy}-g^{xy}g^{yz}}{B^2/c_b^2}\frac{\partial\ln B}{\partial z}, \tag{1.14}
$$

and the density and temperature scale lengths $L_{n\mathrm{s}} = -(d\ln n_{\mathrm{s}}/dx)^{-1}$, $L_{T\mathrm{s}} = -(d\ln T_{\mathrm{s}}/dx)^{-1}$, and total pressure gradient $dP/dx = d(\sum_{\mathrm{s}} n_{\mathrm{s}}T_{\mathrm{s}})/dx = -\sum_{\mathrm{s}} n_{\mathrm{s}}T_{\mathrm{s}}(L_{n\mathrm{s}}^{-1} + L_{T\mathrm{s}}^{-1})$.

1.3 Local approximation

Simulation box $-L_x \leq x < L_x$, $-L_y \leq y < L_y$, $-N_\theta\pi < z < N_\theta\pi$ gives flux-tube domain aligned to the equilibrium magnetic field.

By assuming the perpendicular scale separation of equilibrium and perturbed quantities, the equilibrium quantities can be evaluated by the value at the center of flux-tube domain $x = 0$ or equivalently $\rho_f = \rho_{f0}$. When one considers an axisymmetric equilibrium $\partial_y = 0$, the equilibrium quantities are independent to x and y, i.e., $F = F(z, v_\parallel, \mu)$, $B = B(z)$, and so on. In a non-axisymmetric equilibrium case, one may treat a thin flux-tube domain not only in x but also in y direction and evaluate the equilibrium quantities at $x = 0$ and $y = 0$.

1.4 Pseudo-periodic boundary condition along a field line

Since the equilibrium quantities are independent to perpendicular x and y directions, one expand the distribution function and electromagnetic potentials by means of Fourier basis,

$$\begin{aligned}
\tilde{f}_{\rm s}(\boldsymbol{x}, v_\parallel, \mu, t) &= \sum_{k_x}\sum_{k_y} \tilde{f}_{{\rm s}\boldsymbol{k}}(z, v_\parallel, \mu, t) e^{i(k_x x + i k_y y)} \\
\tilde{\phi}(\boldsymbol{x}', t) &= \sum_{k_x}\sum_{k_y} \tilde{\phi}_{\boldsymbol{k}}(z, t) e^{i(k_x x' + i k_y y')} \\
J_{0{\rm s}}\tilde{\phi}(\boldsymbol{x}, \mu, t) &= \sum_{k_x}\sum_{k_y} J_0(k_\perp \rho_{\rm ts}) \tilde{\phi}_{\boldsymbol{k}}(z, t) e^{i(k_x x + i k_y y)}
\end{aligned} \tag{1.15}$$

where $\boldsymbol{x}$ is the gyrocenter coordinates and $\boldsymbol{x}' = \boldsymbol{x} + \boldsymbol{\rho}_{\rm s}$ is the particle-position coordinates.

Additionally, considering the torus periodicity constraint $\tilde{\phi}(\rho_f, \theta_f + 2N_\theta\pi, \varphi_f) = \tilde{\phi}(\rho_f, \theta_f, \varphi_f)$, one finds the pseudo-periodic boundary condition along a field line,

$$\tilde{\phi}_{k_x + \delta k_x, k_y}(z + 2N_\theta\pi) C_{k_y} = \tilde{\phi}_{k_x, k_y}(z), \tag{1.16}$$

where $\delta k_x = -2N_\theta\pi\hat{s}k_y, C_{k_y} = \exp(i2N_\theta\pi k_y c_y q_0)$. This conversion along a field line physically means twisting of the mode by the parallel streaming in the presence of magnetic shear.

1.5 Collision operator

The present version of GKV equips three types of gyrokinetic model collision operators, operating on the non-adiabatic part of the distribution function $\tilde{g}_{{\rm s}\boldsymbol{k}} = \tilde{f}_{{\rm s}\boldsymbol{k}} + \frac{e_{\rm s} F_{\rm Ms}}{T_{\rm s}} J_{0{\rm s}\boldsymbol{k}} \tilde{\phi}_{\boldsymbol{k}}$.

> **Note**
>
> NOTE: Although the Lenard-Bernstein model collision gkvp_f0.48 operates on $\tilde{f}_{{\rm s}\boldsymbol{k}}$ but not on $\tilde{g}_{{\rm s}\boldsymbol{k}}$ due to historical reason, it will be modified near-future update.

Lenard-Bernstein model collision operator

$$\begin{aligned}
C^{\rm LB}_{{\rm a}\boldsymbol{k}} = \nu_{\rm a} \Bigg[& v_{\rm ta}^2 \frac{\partial^2 \tilde{g}_{{\rm a}\boldsymbol{k}}}{\partial v_\parallel^2} + v_{\rm ta}^2 \frac{\partial^2 \tilde{g}_{{\rm a}\boldsymbol{k}}}{\partial v_\perp^2} + v_\parallel \frac{\partial \tilde{g}_{{\rm a}\boldsymbol{k}}}{\partial v_\parallel} \\
& + \left(\frac{v_{\rm ta}^2}{v_\perp} + v_\perp \right) \frac{\partial \tilde{g}_{{\rm a}\boldsymbol{k}}}{\partial v_\perp} + 3\tilde{g}_{{\rm a}\boldsymbol{k}} - k_\perp^2 \rho_{\rm ta}^2 \tilde{g}_{{\rm a}\boldsymbol{k}} \Bigg].
\end{aligned} \tag{1.17}$$

Lorentz model collision operator

$$C_{\mathrm{a}\boldsymbol{k}}^{\mathrm{Lorentz}} = \nu_{\mathrm{D}}^{\mathrm{ab}} \left[\frac{v_\perp^2}{2} \frac{\partial^2 \tilde{g}_{\mathrm{a}\boldsymbol{k}}}{\partial v_\parallel^2} + \frac{v_\parallel^2}{2} \frac{\partial^2 \tilde{g}_{\mathrm{a}\boldsymbol{k}}}{\partial v_\perp^2} - v_\parallel v_\perp \frac{\partial^2 \tilde{g}_{\mathrm{a}\boldsymbol{k}}}{\partial v_\parallel \partial v_\perp} - v_\parallel \frac{\partial \tilde{g}_{\mathrm{a}\boldsymbol{k}}}{\partial v_\parallel} + \frac{v_\perp}{2} \left(\frac{v_\parallel^2}{v_\perp^2} - 1 \right) \frac{\partial \tilde{g}_{\mathrm{a}\boldsymbol{k}}}{\partial v_\perp} - \frac{k_\perp^2 \rho_{\mathrm{ta}}^2}{4 v_{\mathrm{ta}}^2} (2 v_\parallel^2 + v_\perp^2) \tilde{g}_{\mathrm{a}\boldsymbol{k}} \right]. \tag{1.18}$$

Sugama model collision operator [1-5]

$$C_{\mathrm{a}\boldsymbol{k}}^{\mathrm{Sugama}} = \sum_{\mathrm{b}} \left[C_{\mathrm{ab}}^{\mathrm{V}}(\tilde{g}_{\mathrm{a}\boldsymbol{k}}) + C_{\mathrm{ab}}^{\mathrm{D}}(\tilde{g}_{\mathrm{a}\boldsymbol{k}}) + C_{\mathrm{ab}}^{\mathrm{F}}(\tilde{g}_{\mathrm{b}\boldsymbol{k}}) \right]. \tag{1.19}$$

The test-particle differential term $C_{\mathrm{ab}}^{\mathrm{V}}$, the test-particle non-isothermal term $C_{\mathrm{ab}}^{\mathrm{D}}$, and the field-particle term $C_{\mathrm{ab}}^{\mathrm{F}}$ are given by,

$$\begin{aligned}
C_{\mathrm{ab}}^{\mathrm{V}}(\tilde{g}_{\mathrm{a}\boldsymbol{k}}) &= \frac{\nu_\parallel^{\mathrm{ab}} v_\parallel^2 + \nu_{\mathrm{D}}^{\mathrm{ab}} v_\perp^2}{2} \frac{\partial^2 \tilde{g}_{\mathrm{a}\boldsymbol{k}}}{\partial v_\parallel^2} + \frac{\nu_{\mathrm{D}}^{\mathrm{ab}} v_\parallel^2 + \nu_\parallel^{\mathrm{ab}} v_\perp^2}{2} \frac{\partial^2 \tilde{g}_{\mathrm{a}\boldsymbol{k}}}{\partial v_\perp^2} \\
&\quad + (\nu_\parallel^{\mathrm{ab}} - \nu_{\mathrm{D}}^{\mathrm{ab}}) v_\parallel v_\perp \frac{\partial^2 \tilde{g}_{\mathrm{a}\boldsymbol{k}}}{\partial v_\parallel \partial v_\perp} \\
&\quad + \nu_{\mathrm{g}}^{\mathrm{ab}} v_\parallel \frac{\partial \tilde{g}_{\mathrm{a}\boldsymbol{k}}}{\partial v_\parallel} + \left[\nu_{\mathrm{g}}^{\mathrm{ab}} + \frac{\nu_{\mathrm{D}}^{\mathrm{ab}}}{2} \left(1 + \frac{v_\parallel^2}{v_\perp^2} \right) \right] v_\perp \frac{\partial \tilde{g}_{\mathrm{a}\boldsymbol{k}}}{\partial v_\perp} \\
&\quad + \left[\frac{\nu_{\mathrm{h}}^{\mathrm{ab}} x_{\mathrm{a}}^2}{2} - \frac{k_\perp^2}{4\Omega_{\mathrm{a}}^2} \left\{ \nu_{\mathrm{D}}^{\mathrm{ab}} (2 v_\parallel^2 + v_\perp^2) + \nu_\parallel^{\mathrm{ab}} v_\perp^2 \right\} \right] \tilde{g}_{\mathrm{a}\boldsymbol{k}}, \\
C_{\mathrm{ab}}^{\mathrm{D}}(\tilde{g}_{\mathrm{a}\boldsymbol{k}}) &= \sum_{j=1}^{6} X_j^{\mathrm{ab}} M_j^{\mathrm{ab}}, \\
C_{\mathrm{ab}}^{\mathrm{F}}(\tilde{g}_{\mathrm{b}\boldsymbol{k}}) &= \sum_{j=1}^{6} Y_j^{\mathrm{ab}} M_j^{\mathrm{ba}},
\end{aligned} \tag{1.20}$$

where $x_{\mathrm{a}} = v/(\sqrt{2} v_{\mathrm{ta}})$, $\alpha_{\mathrm{ab}} = v_{\mathrm{ta}}/v_{\mathrm{tb}}$, $\nu_{\mathrm{g}}^{\mathrm{ab}} = \nu_\parallel^{\mathrm{ab}} x_{\mathrm{a}}^2 (1 - \alpha_{\mathrm{ab}})$, and $\nu_{\mathrm{h}}^{\mathrm{ab}} = 3\sqrt{\pi} \tau_{\mathrm{ab}}^{-1} \alpha_{\mathrm{ab}} \Phi'(x_{\mathrm{b}})/(4 x_{\mathrm{a}}^2)$. The energy-diffusion and deflection frequencies are respectively given by $\nu_\parallel^{\mathrm{ab}} = 3\sqrt{\pi} \tau_{\mathrm{ab}}^{-1} G(x_{\mathrm{b}})/(2 x_{\mathrm{a}}^3)$ and $\nu_{\mathrm{D}}^{\mathrm{ab}} = 3\sqrt{\pi} \tau_{\mathrm{ab}}^{-1} [\Phi(x_{\mathrm{b}}) - G(x_{\mathrm{b}})]/(4 x_{\mathrm{a}}^3)$ with the error function $\Phi(x) = \mathrm{erf}(x)$ and $G(x) = [\Phi(x) - x\Phi'(x)]/(2x^2)$. Expressions of the other coefficients X_j^{ab} and Y_j^{ab} and of the fluid moments M_j^{ab} are found, e.g., in the literature [1-6].

1.6 Summary of formulation

Finally, one obtains the δf gyrokinetic Vlasov-Poisson-Ampère equations in a local flux-tube model, represented in perpendicular wave-number space,

$$\begin{aligned}
&\frac{\partial \tilde{f}_{\mathrm{s}\boldsymbol{k}}}{\partial t} + \left(v_\parallel \nabla_\parallel + i\boldsymbol{k} \cdot \boldsymbol{v}_{\mathrm{sG}} + i\boldsymbol{k} \cdot \boldsymbol{v}_{\mathrm{sC}} \right) \left(\tilde{f}_{\mathrm{s}\boldsymbol{k}} + \frac{e_{\mathrm{s}} F_{\mathrm{sM}}}{T_{\mathrm{s}}} J_{0\mathrm{s}\boldsymbol{k}} \tilde{\phi}_{\boldsymbol{k}} \right) + N_{\mathrm{s}\boldsymbol{k}} \\
&- \frac{\mu \nabla_\parallel B}{m_{\mathrm{s}}} \frac{\partial}{\partial v_\parallel} \left(\tilde{f}_{\mathrm{s}\boldsymbol{k}} + \frac{e_{\mathrm{s}} F_{\mathrm{sM}}}{T_{\mathrm{s}}} J_{0\mathrm{s}\boldsymbol{k}} \tilde{\phi}_{\boldsymbol{k}} \right) \\
&+ \frac{e_{\mathrm{s}} F_{\mathrm{sM}}}{T_{\mathrm{s}}} \left[v_\parallel \frac{\partial J_{0\mathrm{s}\boldsymbol{k}} \tilde{A}_{\parallel \boldsymbol{k}}}{\partial t} - i\boldsymbol{k} \cdot \boldsymbol{v}_{\mathrm{s}*} J_{0\mathrm{s}\boldsymbol{k}} (\tilde{\phi}_{\boldsymbol{k}} - v_\parallel \tilde{A}_{\parallel \boldsymbol{k}}) \right] = C_{\mathrm{s}\boldsymbol{k}},
\end{aligned} \tag{1.21}$$

$$\left[k_\perp^2 + \frac{1}{\varepsilon_0}\sum_{\rm s}\frac{e_{\rm s}^2 n_{\rm s}}{T_{\rm s}}\left(1-\Gamma_{0{\rm s}\boldsymbol{k}}\right)\right]\tilde{\phi}_{\boldsymbol{k}} = \frac{1}{\varepsilon_0}\sum_{\rm s} e_{\rm s}\int dv^3 J_{0{\rm s}\boldsymbol{k}}\tilde{f}_{{\rm s}\boldsymbol{k}}, \tag{1.22}$$

$$k_\perp^2 \tilde{A}_{\parallel\boldsymbol{k}} = \mu_0 \sum_{\rm s} e_{\rm s}\int dv^3 J_{0{\rm s}\boldsymbol{k}} v_\parallel \tilde{f}_{{\rm s}\boldsymbol{k}}, \tag{1.23}$$

where $J_{0{\rm s}\boldsymbol{k}} = J_0(k_\perp\rho_{\rm s})$ and $\Gamma_{0{\rm s}\boldsymbol{k}} = I_0(k_\perp^2\rho_{\rm ts}^2)e^{-k_\perp^2\rho_{\rm ts}^2}$ with 0th-order Bessel and modified Bessel functions J_0 and I_0. The included operators are again listed below,

$$\begin{aligned}
\nabla_\parallel &= \frac{c_b}{B\sqrt{g}}\frac{\partial}{\partial z}, \\
k_\perp^2 &= g^{xx}k_x^2 + 2g^{xy}k_xk_y + g^{yy}k_y^2, \\
i\boldsymbol{k}\cdot(\boldsymbol{v}_{\rm sG}+\boldsymbol{v}_{\rm sC}) &= \frac{m_{\rm s}v_\parallel^2+\mu B}{e_{\rm s}c_b}\left(iK_xk_x+iK_yk_y\right) + i\frac{m_{\rm s}v_\parallel^2}{e_{\rm s}c_b}\frac{dP/dx}{B^2/\mu_0}k_y, \\
i\boldsymbol{k}\cdot\boldsymbol{v}_{{\rm s}*} &= -i\frac{T_{\rm s}}{e_{\rm s}c_b}\left[\frac{1}{L_{n{\rm s}}} + \left(\frac{m_{\rm s}v_\parallel^2}{2T_{\rm s}}+\frac{\mu B}{T_{\rm s}}-\frac{3}{2}\right)\frac{1}{L_{T{\rm s}}}\right]k_y, \\
\mathcal{N}_{{\rm s}\boldsymbol{k}} &= -\sum_{\boldsymbol{k}'}\sum_{\boldsymbol{k}''}\delta_{\boldsymbol{k}'+\boldsymbol{k}'',\boldsymbol{k}}\frac{\boldsymbol{b}\cdot\boldsymbol{k}'\times\boldsymbol{k}''}{c_b}J_{0{\rm s}\boldsymbol{k}'} \\
&\quad\times\left(\tilde{\phi}_{\boldsymbol{k}'}-v_\parallel\tilde{A}_{\parallel\boldsymbol{k}'}\right)\left(\tilde{f}_{{\rm s}\boldsymbol{k}''}+\frac{e_{\rm s}F_{\rm Ms}}{T_{\rm s}}J_{0{\rm s}\boldsymbol{k}''}\tilde{\phi}_{\boldsymbol{k}''}\right).
\end{aligned} \tag{1.24}$$

The coefficients for magnetic drift K_x, K_y are given by Eqs. (1.13) and (1.14), and the collision operator $C_{{\rm s}\boldsymbol{k}}$ is given by one of Eqs. (1.17) – (1.19).

CHAPTER

TWO

NORMALIZATION

2.1 Reference units

We denote the reference values of physical quantities as follows.

- Reference magnetic field strength $B_{\rm ref}$ ($= B_a$ magnetic field strength at the magnetic axis)
- Reference length $L_{\rm ref}$ ($= R_a$ major radius at the magnetic axis)
- Reference density $n_{\rm ref}$ ($= n_e(\rho_0)$ electron density at the center of flux-tube domain)
- Reference temperature $T_{\rm ref}$ ($= T_i(\rho_0)$ main ion temperature at the center of flux-tube domain)
- Reference mass $m_{\rm ref}$ ($= m_{\rm p}$ the proton mass)
- Reference electric charge $e_{\rm ref}$ ($= e$ elementary charge)

We also define the following notations $v_{\rm ref} = \sqrt{T_{\rm ref}/m_{\rm ref}}$, $\rho_{\rm ref} = m_{\rm ref} v_{\rm ref}/(e_{\rm ref} B_{\rm ref})$, $\delta_{\rm ref} = \rho_{\rm ref}/L_{\rm ref}$. For single-species simulations with adiabatic electron/ion models, see Appendix *Adiabatic electron/ion model for nprocs=1*.

2.2 Normalized equations

We represent a dimensionless quantity by an overline, $\bar{f}$, in this section.

The coordinates, variables, operators are normalized as

$$
\begin{aligned}
&t = \frac{L_{\rm ref}}{v_{\rm ref}}\bar{t},\ \ x = \rho_{\rm ref}\bar{x},\ \ k_x = \frac{1}{\rho_{\rm ref}}\bar{k}_x,\ \ y = \rho_{\rm ref}\bar{y},\ \ k_y = \frac{1}{\rho_{\rm ref}}\bar{k}_y,\ \ z = \bar{z},\\
&v_\parallel = v_{\rm ts}\bar{v}_\parallel = v_{\rm ref}\sqrt{\frac{\bar{T}_{\rm s}}{\bar{m}_{\rm s}}}\bar{v}_\parallel,\ \ \mu = \frac{T_{\rm s}}{B_{\rm ref}}\bar{\mu} = \frac{T_{\rm ref}}{B_{\rm ref}}T_{\rm s}\bar{\mu},\\
&\tilde{f}_{s\boldsymbol{k}} = \delta_{\rm ref}\frac{n_{\rm s}}{v_{\rm ts}^3}\bar{f}_{s\boldsymbol{k}},\ \ \tilde{\phi}_{\boldsymbol{k}} = \delta_{\rm ref}\frac{T_{\rm ref}}{e_{\rm ref}}\bar{\phi}_{\boldsymbol{k}},\ \ \tilde{A}_{\parallel\boldsymbol{k}} = \delta_{\rm ref}\rho_{\rm ref}B_{\rm ref}\bar{A}_{\parallel\boldsymbol{k}},\\
&n_{\rm s} = n_{\rm ref}\bar{n}_{\rm s},\ \ T_{\rm s} = T_{\rm ref}\bar{T}_{\rm s},\ \ m_{\rm s} = m_{\rm ref}\bar{m}_{\rm s},\ \ e_{\rm s} = e_{\rm ref}\bar{e}_{\rm s},\\
&\nabla_\parallel = \frac{1}{L_{\rm ref}}\bar{\nabla}_\parallel,\ \ \boldsymbol{v}_{\rm sG} = \delta_{\rm ref}v_{\rm ref}\bar{\boldsymbol{v}}_{\rm sG},\ \ \boldsymbol{v}_{\rm sC} = \delta_{\rm ref}v_{\rm ref}\bar{\boldsymbol{v}}_{\rm sC},\ \ \boldsymbol{v}_{\rm s*} = \delta_{\rm ref}v_{\rm ref}\bar{\boldsymbol{v}}_{\rm s*},\\
&F_{\rm sM} = \frac{n_{\rm s}}{v_{\rm ts}^3}\bar{F}_{\rm sM},\ \ J_{0s\boldsymbol{k}} = \bar{J}_{0s\boldsymbol{k}},\ \ \Gamma_{0s\boldsymbol{k}} = \bar{\Gamma}_{0s\boldsymbol{k}},\\
&K_x = \frac{1}{L_{\rm ref}}\bar{K}_x,\ \ K_y = \frac{1}{L_{\rm ref}}\bar{K}_y, L_{n\rm s} = L_{\rm ref}\bar{L}_{n\rm s},\ \ L_{T\rm s} = L_{\rm ref}\bar{L}_{T\rm s},\\
&\frac{dP}{dx} = \frac{n_{\rm ref}T_{\rm ref}}{L_{\rm ref}}\frac{d\bar{P}}{d\bar{x}},\ \ c_b = B_{\rm ref}\bar{c}_b,\ \ B = B_{\rm ref}\bar{B},\\
&\frac{\partial\ln B}{\partial x} = \frac{1}{L_{\rm ref}}\frac{\partial\ln\bar{B}}{\partial\bar{x}},\ \ \frac{\partial\ln B}{\partial y} = \frac{1}{L_{\rm ref}}\frac{\partial\ln\bar{B}}{\partial\bar{y}},\ \ \frac{\partial\ln B}{\partial z} = \frac{\partial\ln\bar{B}}{\partial\bar{z}},\\
&g^{xx} = \bar{g}^{xx},\ \ g^{xy} = \bar{g}^{xy}, g^{xz} = \frac{1}{L_{\rm ref}}\bar{g}^{xz},\ \ g^{yy} = \bar{g}^{yy},\\
&g^{yz} = \frac{1}{L_{\rm ref}}\bar{g}^{yz},\ \ g^{zz} = \frac{1}{L_{\rm ref}^2}\bar{g}^{zz},\ \ \sqrt{g} = L_{\rm ref}\sqrt{\bar{g}},\ \ \nu = \frac{v_{\rm ref}}{L_{\rm ref}}\bar{\nu},\\
&N_{s\boldsymbol{k}} = \frac{v_{\rm ref}}{L_{\rm ref}}\delta_{\rm ref}\frac{n_{\rm s}}{v_{\rm ts}^3}\bar{N}_{s\boldsymbol{k}},\ \ C_{s\boldsymbol{k}} = \frac{v_{\rm ref}}{L_{\rm ref}}\delta_{\rm ref}\frac{n_{\rm s}}{v_{\rm ts}^3}\bar{C}_{s\boldsymbol{k}}.
\end{aligned}
$$

Then, the normalized equations are

$$
\begin{aligned}
&\frac{\partial\bar{f}_{s\boldsymbol{k}}}{\partial\bar{t}} + \left(\sqrt{\frac{\bar{T}_{\rm s}}{\bar{m}_{\rm s}}}\bar{v}_\parallel\bar{\nabla}_\parallel + i\bar{\boldsymbol{k}}\cdot\bar{\boldsymbol{v}}_{\rm sG} + i\bar{\boldsymbol{k}}\cdot\bar{\boldsymbol{v}}_{\rm sC}\right)\left(\bar{f}_{s\boldsymbol{k}} + \frac{\bar{e}_{\rm s}\bar{F}_{\rm sM}}{\bar{T}_{\rm s}}\bar{J}_{0s\boldsymbol{k}}\bar{\phi}_{\boldsymbol{k}}\right)\\
&+ \bar{N}_{s\boldsymbol{k}} - \sqrt{\frac{\bar{T}_{\rm s}}{\bar{m}_{\rm s}}}\bar{\mu}\bar{\nabla}_\parallel\bar{B}\frac{\partial}{\partial\bar{v}_\parallel}\left(\bar{f}_{\rm s} + \frac{\bar{e}_{\rm s}\bar{F}_{\rm sM}}{\bar{T}_{\rm s}}\bar{J}_{0\rm s}\bar{\phi}\right)\\
&+ \frac{e_{\rm s}F_{\rm sM}}{T_{\rm s}}\left[\sqrt{\frac{\bar{T}_{\rm s}}{\bar{m}_{\rm s}}}\bar{v}_\parallel\frac{\partial\bar{J}_{0\rm s}\bar{A}_\parallel}{\partial\bar{t}} - i\bar{\boldsymbol{k}}\cdot\bar{\boldsymbol{v}}_{\rm s*}\bar{J}_{0\rm s}(\bar{\phi} - \sqrt{\frac{\bar{T}_{\rm s}}{\bar{m}_{\rm s}}}\bar{v}_\parallel\bar{A}_\parallel)\right] = \bar{C}_{s\boldsymbol{k}},
\end{aligned}
$$

$$
\left[\bar{\lambda}_{\rm D}^2\bar{k}_\perp^2 + \sum_{\rm s}\frac{\bar{e}_{\rm s}^2\bar{n}_{\rm s}}{\bar{T}_{\rm s}}\left(1 - \bar{\Gamma}_{0s\boldsymbol{k}}\right)\right]\bar{\phi}_{\boldsymbol{k}} = \sum_{\rm s}\bar{e}_{\rm s}\bar{n}_{\rm s}\int d\bar{v}^3\,\bar{J}_{0s\boldsymbol{k}}\bar{f}_{s\boldsymbol{k}}, \tag{2.1}
$$

$$
\bar{k}_\perp^2\bar{A}_{\parallel\boldsymbol{k}} = \bar{\beta}\sum_{\rm s}\bar{e}_{\rm s}\bar{n}_{\rm s}\int d\bar{v}^3\bar{J}_{0s\boldsymbol{k}}\sqrt{\frac{\bar{T}_{\rm s}}{\bar{m}_{\rm s}}}\bar{v}_\parallel\bar{f}_{s\boldsymbol{k}}, \tag{2.2}
$$

where we introduced $\bar{\lambda}_{\rm D}^2 = \frac{\lambda_{\rm D,ref}^2}{\rho_{\rm ref}^2} = \frac{\varepsilon_0 T_{\rm ref}/e_{\rm ref}^2 n_{\rm ref}}{\rho_{\rm ref}^2}$ and $\bar{\beta} = \frac{\mu_0 n_{\rm ref} T_{\rm ref}}{B_{\rm ref}^2}$. The included terms and operators are

$$
\begin{aligned}
&\bar{\nabla}_\parallel = \frac{\bar{c}_b}{\bar{B}\sqrt{\bar{g}}}\frac{\partial}{\partial z},\ \ \bar{k}_\perp^2 = \bar{g}^{xx}\bar{k}_x^2 + 2\bar{g}^{xy}\bar{k}_x\bar{k}_y + \bar{g}^{yy}\bar{k}_y^2,\\
&i\bar{\boldsymbol{k}}\cdot(\bar{\boldsymbol{v}}_{\rm sG} + \bar{\boldsymbol{v}}_{\rm sC}) = \frac{\bar{T}_{\rm s}(\bar{v}_\parallel^2 + \bar{\mu}\bar{B})}{\bar{e}_{\rm s}\bar{c}_b}\left(i\bar{K}_x\bar{k}_x + i\bar{K}_y\bar{k}_y\right) + i\frac{\bar{T}_{\rm s}\bar{v}_\parallel^2}{\bar{e}_{\rm s}\bar{c}_b}\bar{\beta}\frac{d\bar{P}/d\bar{x}}{\bar{B}^2}\bar{k}_y,\\
&i\bar{\boldsymbol{k}}\cdot\bar{\boldsymbol{v}}_{\rm s*} = -i\frac{\bar{T}_{\rm s}}{\bar{e}_{\rm s}\bar{c}_b}\left[\frac{1}{\bar{L}_{n\rm s}} + \left(\frac{\bar{v}_\parallel^2}{2} + \bar{\mu}\bar{B} - \frac{3}{2}\right)\frac{1}{\bar{L}_{T\rm s}}\right]\bar{k}_y,\\
&\bar{\mathcal{N}}_{{\rm s}\boldsymbol{k}} = -\sum_{\bar{\boldsymbol{k}}'}\sum_{\bar{\boldsymbol{k}}''}\delta_{\bar{\boldsymbol{k}}'+\bar{\boldsymbol{k}}'',\bar{\boldsymbol{k}}}\frac{\bar{\boldsymbol{b}}\cdot\bar{\boldsymbol{k}}'\times\bar{\boldsymbol{k}}''}{\bar{c}_b}\\
&\qquad\qquad\times \bar{J}_{0{\rm s}\boldsymbol{k}'}\left(\bar{\phi}_{\boldsymbol{k}'} - \sqrt{\frac{\bar{T}_{\rm s}}{\bar{m}_{\rm s}}}\bar{v}_\parallel\bar{A}_{\parallel\boldsymbol{k}'}\right)\left(\bar{f}_{{\rm s}\boldsymbol{k}''} + \frac{\bar{e}_{\rm s}\bar{F}_{\rm Ms}}{\bar{T}_{\rm s}}\bar{J}_{0{\rm s}\boldsymbol{k}''}\bar{\phi}_{\boldsymbol{k}''}\right),\\
&\bar{K}_x = -\frac{\partial\ln\bar{B}}{\partial\bar{y}} + \frac{\bar{g}^{xz}\bar{g}^{yx} - \bar{g}^{xx}\bar{g}^{yz}}{\bar{B}^2/\bar{c}_b^2}\frac{\partial\ln\bar{B}}{\partial\bar{z}},\\
&\bar{K}_y = \frac{\partial\ln\bar{B}}{\partial\bar{x}} + \frac{\bar{g}^{xz}\bar{g}^{yy} - \bar{g}^{xy}\bar{g}^{yz}}{\bar{B}^2/\bar{c}_b^2}\frac{\partial\ln\bar{B}}{\partial\bar{z}},\\
&\frac{d\bar{P}}{d\bar{x}} = -\sum_{\rm s}\bar{n}_{\rm s}\bar{T}_{\rm s}\left(\frac{1}{\bar{L}_{n\rm s}} + \frac{1}{\bar{L}_{T\rm s}}\right),\ \ \bar{F}_{\rm sM} = \frac{1}{(2\pi)^{\frac{3}{2}}}e^{-\frac{\bar{v}_\parallel^2}{2} - \bar{\mu}\bar{B}},\\
&\bar{J}_{0{\rm s}\boldsymbol{k}} = J_0\left(\bar{k}_\perp\bar{\rho}_{\rm s}\right),\ \ \bar{\Gamma}_{0{\rm s}\boldsymbol{k}} = I_0\left(\bar{k}_\perp^2\bar{\rho}_{\rm ts}^2\right)e^{-\bar{k}_\perp^2\bar{\rho}_{\rm ts}^2},\\
&\bar{\rho}_{\rm s} = \sqrt{\frac{2\bar{m}_{\rm s}\bar{T}_{\rm s}\bar{\mu}}{\bar{e}_{\rm s}^2\bar{B}}},\ \ \bar{\rho}_{\rm ts} = \frac{\sqrt{\bar{m}_{\rm s}\bar{T}_{\rm s}}}{\bar{e}_{\rm s}\bar{B}}
\end{aligned}
$$

and the normalized Lenard-Bernstein, Lorentz, and Sugama collision operators are

$$
\bar{C}_{{\rm a}\boldsymbol{k}}^{\rm LB} = \bar{\nu}_{\rm a}\left[\frac{\partial^2\bar{g}_{{\rm a}\boldsymbol{k}}}{\partial\bar{v}_\parallel^2} + \frac{\partial^2\bar{g}_{{\rm a}\boldsymbol{k}}}{\partial\bar{v}_\perp^2} + \bar{v}_\parallel\frac{\partial\bar{g}_{{\rm a}\boldsymbol{k}}}{\partial\bar{v}_\parallel} + \left(\frac{1}{\bar{v}_\perp} + \bar{v}_\perp\right)\frac{\partial\bar{g}_{{\rm a}\boldsymbol{k}}}{\partial\bar{v}_\perp} + 3\bar{g}_{{\rm a}\boldsymbol{k}} - \bar{k}_\perp^2\bar{\rho}_{\rm ts}^2\bar{g}_{{\rm a}\boldsymbol{k}}\right], \tag{2.3}
$$

$$
\begin{aligned}
\bar{C}_{{\rm a}\boldsymbol{k}}^{\rm Lorentz} = \bar{\nu}_{\rm D}^{\rm ab}\Bigg[&\frac{\bar{v}_\perp^2}{2}\frac{\partial^2\bar{g}_{{\rm a}\boldsymbol{k}}}{\partial\bar{v}_\parallel^2} + \frac{\bar{v}_\parallel^2}{2}\frac{\partial^2\bar{g}_{{\rm a}\boldsymbol{k}}}{\partial\bar{v}_\perp^2}\\
&- \bar{v}_\parallel\bar{v}_\perp\frac{\partial^2\bar{g}_{{\rm a}\boldsymbol{k}}}{\partial\bar{v}_\parallel\partial\bar{v}_\perp} - \bar{v}_\parallel\frac{\partial\bar{g}_{{\rm a}\boldsymbol{k}}}{\partial\bar{v}_\parallel} + \frac{\bar{v}_\perp}{2}\left(\frac{\bar{v}_\parallel^2}{\bar{v}_\perp^2} - 1\right)\frac{\partial\bar{g}_{{\rm a}\boldsymbol{k}}}{\partial\bar{v}_\perp}\\
&- \frac{\bar{k}_\perp^2\bar{\rho}_{\rm ta}^2}{4}(2\bar{v}_\parallel^2 + \bar{v}_\perp^2)\bar{g}_{{\rm a}\boldsymbol{k}}\Bigg],
\end{aligned}
$$

$$
\begin{aligned}
\bar{C}^{\mathrm{Sugama}}_{\mathrm{a}\boldsymbol{k}} &= \sum_{\mathrm{b}} \left[C^{\mathrm{V}}_{\mathrm{ab}}(\bar{g}_{\mathrm{a}\boldsymbol{k}}) + C^{\mathrm{D}}_{\mathrm{ab}}(\bar{g}_{\mathrm{a}\boldsymbol{k}}) + C^{\mathrm{F}}_{\mathrm{ab}}(\bar{g}_{\mathrm{b}\boldsymbol{k}}) \right], \\
C^{\mathrm{V}}_{\mathrm{ab}}(\bar{g}_{\mathrm{a}\boldsymbol{k}}) &= \frac{\bar{\nu}^{\mathrm{ab}}_{\parallel}\bar{v}^2_{\parallel} + \bar{\nu}^{\mathrm{ab}}_{\mathrm{D}}\bar{v}^2_{\perp}}{2} \frac{\partial^2 \bar{g}_{\mathrm{a}\boldsymbol{k}}}{\partial \bar{v}^2_{\parallel}} \\
&\quad + \frac{\bar{\nu}^{\mathrm{ab}}_{\mathrm{D}}\bar{v}^2_{\parallel} + \bar{\nu}^{\mathrm{ab}}_{\parallel}\bar{v}^2_{\perp}}{2} \frac{\partial^2 \bar{g}_{\mathrm{a}\boldsymbol{k}}}{\partial \bar{v}^2_{\perp}} + (\bar{\nu}^{\mathrm{ab}}_{\parallel} - \bar{\nu}^{\mathrm{ab}}_{\mathrm{D}})\bar{v}_{\parallel}\bar{v}_{\perp} \frac{\partial^2 \bar{g}_{\mathrm{a}\boldsymbol{k}}}{\partial \bar{v}_{\parallel}\partial \bar{v}_{\perp}} \\
&\quad + \bar{\nu}^{\mathrm{ab}}_{\mathrm{g}}\bar{v}_{\parallel} \frac{\partial \bar{g}_{\mathrm{a}\boldsymbol{k}}}{\partial \bar{v}_{\parallel}} + \left[\bar{\nu}^{\mathrm{ab}}_{\mathrm{g}} + \frac{\bar{\nu}^{\mathrm{ab}}_{\mathrm{D}}}{2} \left(1 + \frac{\bar{v}^2_{\parallel}}{\bar{v}^2_{\perp}} \right) \right] \bar{v}_{\perp} \frac{\partial \bar{g}_{\mathrm{a}\boldsymbol{k}}}{\partial \bar{v}_{\perp}} \\
&\quad + \left[\frac{\bar{\nu}^{\mathrm{ab}}_{\mathrm{h}}\bar{v}^2}{4} - \frac{\bar{k}^2_{\perp}\bar{\rho}^2_{\mathrm{ta}}}{4} \left\{ \bar{\nu}^{\mathrm{ab}}_{\mathrm{D}}(2\bar{v}^2_{\parallel} + \bar{v}^2_{\perp}) + \bar{\nu}^{\mathrm{ab}}_{\parallel}\bar{v}^2_{\perp} \right\} \right] \bar{g}_{\mathrm{a}\boldsymbol{k}}, \\
C^{\mathrm{D}}_{\mathrm{ab}}(\bar{g}_{\mathrm{a}\boldsymbol{k}}) &= \sum_{j=1}^{6} \bar{X}^{\mathrm{ab}}_{j} \bar{M}^{\mathrm{ab}}_{j}, \\
C^{\mathrm{F}}_{\mathrm{ab}}(\bar{g}_{\mathrm{b}\boldsymbol{k}}) &= \sum_{j=1}^{6} \bar{Y}^{\mathrm{ab}}_{j} \bar{M}^{\mathrm{ba}}_{j}.
\end{aligned}
$$

Normalized input parameters for GKV are summarized in Appendix *List of GKV namelist*.

CHAPTER

THREE

DISCRETIZATION

3.1 Spatial discretization

GKV is a Vlasov (continuum) simulation code. The perpendicular directions are already given by a discrete representation in Fourier space (k_x, k_y). The other three directions $(z, v_\parallel, \mu)$ are discretized by an equidistant grid. Defining box sizes $-L_x \le \bar{x} < L_x, -L_y \le \bar{y} < L_y, -L_z \le \bar{z} < L_z, -L_v \le \bar{v}_\parallel \le L_v, 0 \le \bar{\mu} \le \frac{L_v^2}{2}$ and grid numbers $(2\mathtt{nx}+1, \mathtt{global\_ny}+1, 2\mathtt{global\_nz}, 2\mathtt{global\_nv}, \mathtt{global\_nm}+1)$ in $(k_x, k_y, z, v_\parallel, \mu)$, the grid of GKV is given by

$$
\begin{aligned}
k_x &= \mathtt{mx}\Delta k_x \ \ (-\mathtt{nx} \le \mathtt{mx} \le \mathtt{nx}),\\
k_y &= \mathtt{my}\Delta k_y \ \ (0 \le \mathtt{my} \le \mathtt{global\_ny})\\
z &= \mathtt{iz}\Delta z \ \ (-\mathtt{global\_nz} \le \mathtt{iz} \le \mathtt{global\_nz} - 1),\\
v_\parallel &= -L_v + (\mathtt{iv}-1)\Delta v_\parallel \ \ (1 \le \mathtt{iv} \le 2\mathtt{global\_nv}),\\
\mu &= \frac{(\mathtt{im}\Delta w)^2}{2} \ \ (0 \le \mathtt{im} \le \mathtt{global\_nm}),
\end{aligned}
$$

where $\Delta k_x = \frac{\pi}{L_x}, \Delta k_y = \frac{\pi}{L_y}, \Delta z = \frac{L_z}{\mathtt{global\_nz}}, \Delta v_\parallel = \frac{2L_v}{2\mathtt{global\_nv}-1}, \Delta w = \frac{L_v}{\mathtt{global\_nm}}$.

Derivatives in $(z, v_\parallel, \mu)$ are discretized by finite difference method, and then, GKV solves the δf gyrokinetic equations, Eqs. (2.1) – (2.2), in $(k_x, k_y, z, v_\parallel, \mu)$ space, except for the nonlinear term. Since direct calculation of nonlinear convolution in wavenumber space is computationally expensive, the nonlinear term is evaluated in the real space, employing $(2\mathtt{nxw}, 2\mathtt{nyw}, 2\mathtt{global\_nz}, 2\mathtt{global\_nv}, \mathtt{global\_nm}+1)$ grid points in $(x, y, z, v_\parallel, \mu)$, and is transformed back to the wavenumber space by means of 2D Fast Fourier Transform (FFT) algorithm and the 3/2 de-aliasing rule in (k_x, k_y).

To implement the pseudo-periodic boundary condition along a field line, Eq. (1.16), the shift of the radial wave number $\delta k_x(k_y) = -2N_\theta \pi \hat{s} k_y$ as a function of k_y should be equal to the integral multiple of the minimum radial wave number Δk_x. This gives a constraint between radial and field-line-label box sizes. In GKV, the minimum field-line-label wave number Δk_y and the ratio $m = \left|\frac{\delta k_x(\Delta k_y)}{N_\theta \Delta k_x}\right|$ are given in the namelist (`kymin` and `m_j`, respectively), and then, $\Delta k_x = |2\pi\hat{s}\Delta k_y/m|$, $L_x = \pi/\Delta k_x$, $L_y = \pi/\Delta k_y$.

3.1.1 Choice of velocity-space coordinate

After gkvp_f0.63, altanate velocity space coordinate, the parallel and perpendicular velocities $(v_\parallel, v_\perp)$ rather than the magnetic coordinate $(v_\parallel, \mu)$. This choice is beneficial when the magnetic field strength significantly varies, like the dipole geometry. Switching the velocity-space coordinates is done in run/gkvp_namelist:

```
vp_coord = 0, # For (vl,mu) coordinates
```

or

```
vp_coord = 1, # For (vl,vp) coordinates
```

In the equation, the parallel advection term is modified:

$$\frac{\partial}{\partial z}_{(z,v_\parallel,\mu)} \to \frac{\partial}{\partial z}_{(z,v_\parallel,v_\perp)} + \frac{v_\perp}{2B}\frac{\partial B}{\partial z}\frac{\partial}{\partial v_\perp}_{(z,v_\parallel,v_\perp)} \tag{3.1}$$

3.2 Temporal discretization

3.2.1 Explicit implementation

GKV usually uses 4th-order explicit Runge-Kutta-Gill method. [3-1]

3.2.2 Explicit collisionless physics and implicit collision implementation

An alternative option is implicit collision solver which is useful for Lorentz or Sugama collision operators having velocity-dependent collision frequencies [3-2]. Splitting collisionless physics and collision operator by means of 2nd-order (Strang) operator split, the collisionless physics is solved by using 4th-order explicit Runge-Kutta-Gill method, while the collision operator is solved by using 2nd-order semi-implicit Crank-Nicolson method. Bi-CGSTAB method is used as an iterative matrix solver for implicit collision.

3.3 Inter-node decomposition by using MPI

The computations are parallelized by using the OpenMP/MPI hybrid parallelization which suites for hierarchical hardware of the nodes having a number of cores with a shared memory and connected by an interconnect network.

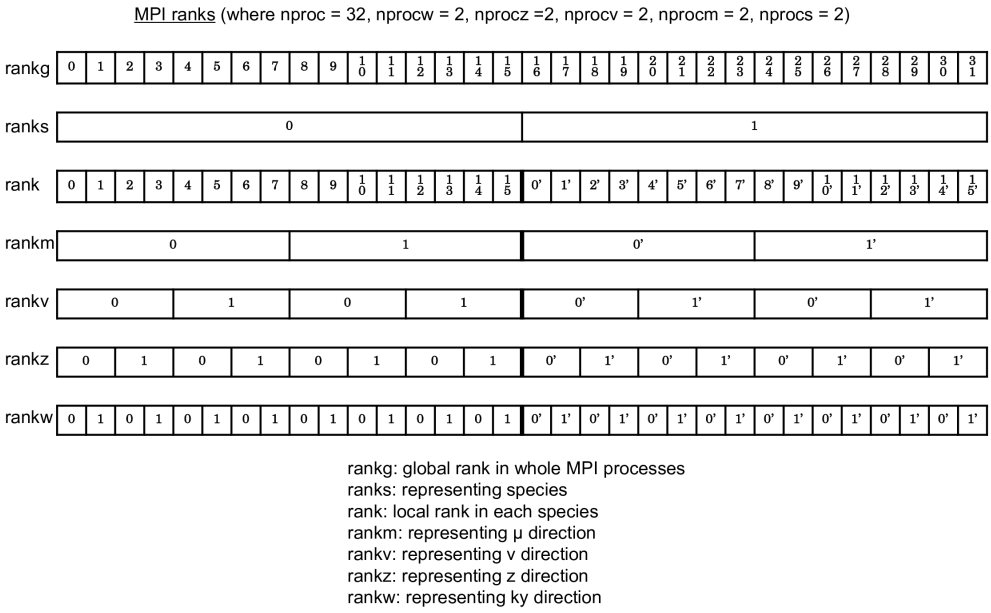

Fig. 3.1: An example of MPI ranks in GKV.

Taking advantage of the multi-dimensional problem, multi-dimensional domain decomposition is applied for y, z, $v_\parallel$, μ and s, where 2D FFTs in x and y are parallelized by means of the transpose split method. Then, the required MPI communications are data transpose for the parallel 2D FFTs in x and y, point-to-point communications in z, $v_\parallel$ and μ for finite difference methods, and reduction communications over $v_\parallel$, μ and s for velocity-space and species

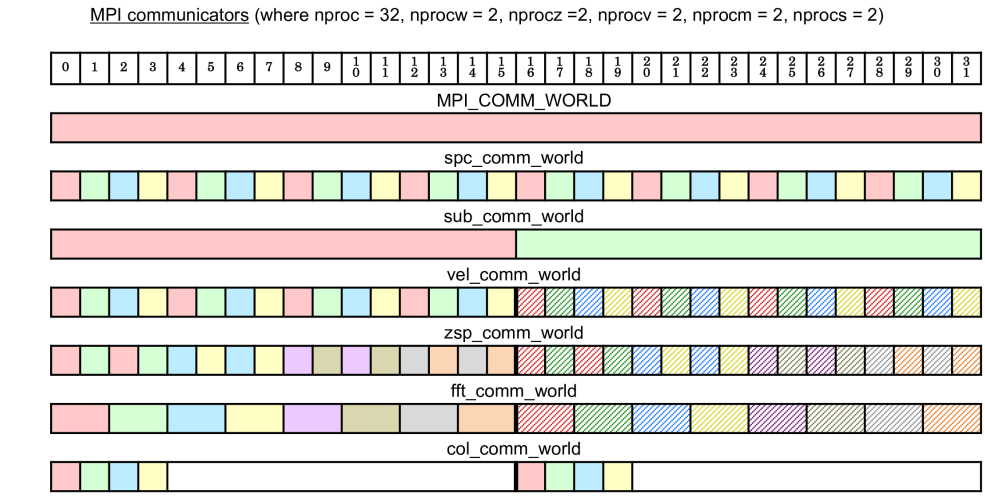

Fig. 3.2: An example of MPI communicators in GKV.

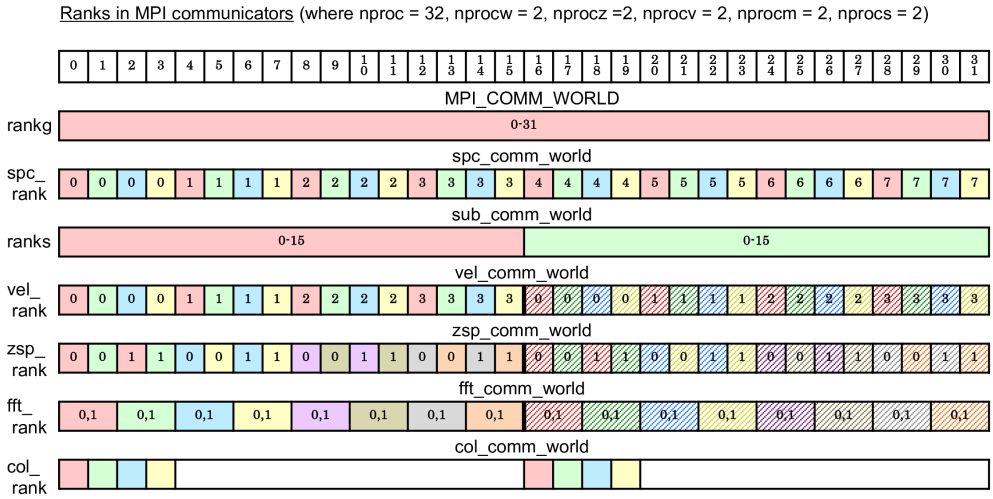

Fig. 3.3: An example of MPI ranks in communicators in GKV.

integration. Figures 3.1 – 3.3 show schematic pictures of the multi-layer structure of the multi-dimensional domain decomposition, illustrating the case that y, z, $v_{\parallel}$, μ and s are respectively split by two MPI processes (and thus $2^5 = 32$ processes in total). Plasma species s are decomposed as `ranks` $= 0, 1$, and each species are hierarchically decomposed by the magnetic moment μ (`rankm`), the parallel velocity $v_{\parallel}$ (`rankv`), the parallel direction z (`rankz`), and the perpendicular direction x, y (`rankw`). Thus, data transpose in x and y is performed for different subranks of `rankw` by using `fft_comm_world` communicator, point-to-point communications in z (v or m) are performed between `rankz` (`rankv` or `rankm`), while reduction communications over v, μ and s are performed for fixed `rankz` and `rankw` by using `spc_comm_world` communicator.

3.4 Intra-node decomposition by using OpenMP

Intra-node decomposition is basically implemented by loop-level parallelization with OpenMP. Time-consuming MPI communications are masked by computation-communication overlap technique using MASTER thread. For more details, see Ref. [3-3].

CHAPTER

FOUR

SIMULATION

4.1 Structure of GKV

NOTE: This explanation is based on the GKV version gkvp_f0.65. When one expands the GKV package, there are

- `gkvp_f0.65/`
  - `README_for_namelist.txt`
  - `Version_memo.txt`
  - `src/`
    - `gkvp_header.f90` (Module for setting grid resolutions and MPI processes)
    - `gkvp_out.f90` (Module for data output)
    - ...
  - `lib/`
    - ... (contains libraries for random number and Bessel functions)
  - `extra_tools/`
  - `run/`
    - `gkvp_namelist` (Namelist for setting physical plasma parameters)
    - `sub.q` (Batch script, depending on machines)
    - `shoot` (Script for submitting jobs, depending on machines)
    - `Makefile` (depending on machines)
    - `backup/`
    - ...

4.2 Setting parameters

Listing 4.1: src/gkvp_header.f90

```
...
!-------------------------------------
! Dimension size (grid numbers)
!-------------------------------------
! Global simulation domain
! in x, y,z,v,m (0:2*nxw-1, 0:2*nyw-1,-global_nz:global_nz-1,1:2*global_nv,0:global_nm)
```

(continues on next page)

(continued from previous page)

```
! in kx,ky,z,v,m ( -nx:nx,0:global_ny,-global_nz:global_nz-1,1:2*global_nv,0:global_nm)
integer, parameter :: nxw = 128, nyw = 64
integer, parameter :: nx = 84, global_ny = 41 ! 2/3 de-aliasing rule
integer, parameter :: global_nz = 24, global_nv = 48, global_nm = 15
integer, parameter :: nzb = 3, &  ! the number of ghost grids in z
                      nvb = 3    ! the number of ghost grids in v and m
!-------------------------------------
! Data distribution for MPI
!-------------------------------------
integer, parameter :: nprocw = 2, nprocz = 2, nprocv = 4, nprocm = 2, nprocs = 2
...
```

Grid resolutions and MPI processes are set in src/gkvp_header.f90.

- nxw | Grid number in x
- nyw | Grid number in y
- nx | Mode number in k_x (-nx:nx)
- global_ny | Mode number in k_y (0:global_ny)
- global_nz | Grid number in z (-global_nz:global_nz-1.)
- global_nv | Grid number in $v_\parallel$ (1:2global_nv.)
- global_nm | Grid number in μ (0:global_nm.)
- nprocw | MPI processes for k_y decomposition
- nprocz | MPI processes for z decomposition
- nprocv | MPI processes for $v_\parallel$ decomposition
- nprocm | MPI processes for μ decomposition
- nprocs | MPI processes for s decomposition

Note that (i) nxw $>$ 3nx/2 and nyw $>$ 3global_ny/2 in nonlinear simulations; (ii) global_nz/nprocz, global_nv/nprocv, (global_nm + 1)/nprocm should be integer; (iii) nprocs is same as the number of kinetic plasma species; (iv) (global_nm + 1)/nprocm ≥ 4. nzb and nvb are the number of ghost grid in z and $v_\parallel/\mu$, whose required numbers depend on the employed finite difference methods. nzb =``nvb`` =3 is enough by default. Plasma parameters are set in run/gkvp_namelist. See Appendix *List of GKV namelist* in detail. GKV has MHD equilibrium interfaces, IGS for Tokamaks and BZX for Stellarators. See Appendix *Use of MHD equilibrium interfaces* for the use of IGS and BZX.

4.3 Building

Prepare a proper run/Makefile. Some samples are found in run/backup/. Then,

```
$ cd run
$ make
```

will create the load module run/gkvp.exe.

4.4 Running

Prepare proper `run/sub.q` and `run/shoot`. Some samples are found in `run/backup/`.

Listing 4.2: run/sub.q for Sub-system A of the Plasma Simulator at NIFS/QST

```
#!/bin/bash
#PBS -P YOUR_PROJECT_NAME
#PBS -q A_S
#PBS -l walltime=00:15:00
#PBS -l select=1:ncpus=128:mem=376gb:mpiprocs=16

export OMP_NUM_THREADS=8   # Set number of OpenMP threads per MPI process

...
```

In the batch script `sub.q`, users specify the numbers of available computation nodes, MPI processes, and OpenMP threads. Required MPI process number of GKV is $\texttt{nproc} = \texttt{nprocw} * \texttt{nprocz} * \texttt{nprocv} * \texttt{nprocm} * \texttt{nprocs}$. Usually, $\texttt{nproc} * \texttt{OMP\_NUM\_THREADS} = \texttt{nodes} * (\texttt{cores per node})$ is a reasonable choice.

Listing 4.3: run/shoot

```
#!/bin/csh
#### Environment setting
set DIR=/data/maeyama/gkv_training/test01
set LDM=gkvp.exe
set NL=gkvp_namelist
set SC=qsub
set JS=sub.q
## For VMEC, set VMCDIR including metric_boozer.bin.dat
set VMCDIR=./input_vmec
## For IGS, set IGSDIR including METRIC_{axi,boz,ham}.OUT
set IGSDIR=./input_eqdsk
...
```

In the Script for submitting jobs, `shoot`, users set the output directory `DIR` where all output of GKV will be dumped. After finishing the all settings, type as follow to submit step jobs,

```
$ cd run/
$ ./shoot START_NUM END_NUM (JOB_ID)
```

For example, `./shoot 1 1` gives a single job (*.001). Similarly, `./shoot 2 2` gives a single job (*.002), which continues from the first run (*.001). Step job submission allows some sets of successive continuing jobs, e.g., `./shoot 3 5` gives step jobs (*.003 - *.005). Above three examples assume that the previous job has already finished. If a previous (*.005) job having `JOB_ID = 11223` is still in queue, `./shoot 6 7 11223` adds step jobs (*.006 - *.007) which sequentially follow after the end of previous job (*.005).

Before running expensive nonlinear simulations, it is strongly recommended to test computational performance and its scalability: (i) Run a short-time run at the target problem size, (ii) Try some combination of $(\texttt{nprocw}, \texttt{nprocz}, \texttt{nprocv}, \texttt{nprocm}, \texttt{nprocs}, \texttt{OMP\_NUM\_THREADS})$ while keeping the number of $\texttt{node} * (\texttt{cores per node})$, (iii) Check scalablity of the computational performance against the number of computation nodes. Optimal setting may strongly depend on the target problem size.

Listing 4.4: run/gkvp_namelist

```
&cmemo memo="GKV-plus f0.65 developed for pre-exa-scale computing", &end
&calct calc_type="lin_freq",
       z_bound="outflow",
       z_filt="off",
       z_calc="cf4",
       art_diff=0.1d0,
       init_random=.true.,
       num_triad_diag=0,
       vp_coord=1, &end
&triad mxt = 0, myt = 0/
&equib equib_type = "analytic", &end
&run_n inum=%%%,
       ch_res = .false., &end
&files f_log="%%DIR%%/log/gkvp.",
       f_hst="%%DIR%%/hst/gkvp.",
       f_phi="%%DIR%%/phi/gkvp.",
       f_fxv="%%DIR%%/fxv/gkvp.",
       f_cnt="%%DIR%%/cnt/gkvp.", &end
&runlm e_limit = 60.d0, &end
&times tend = 200.d0,
       dtout_fxv = 10.d0,
       dtout_ptn = 0.1d0,
       dtout_eng = 0.1d0,
       dtout_dtc = 0.1d0, &end
&deltt dt_max = 0.01d0,
       adapt_dt = .true.,
       courant_num = 0.5d0,
       time_advnc = "auto_init", &end
&physp R0_Ln = 2.22d0,
       R0_Lt = 6.92d0,
       nu = 1.d0,
       Anum = 1.d0,
       Znum = 1.d0,
       fcs = 1.d0,
       sgn = 1.d0,
       tau = 1.d0,
       dns1 = 1.d-2,
       tau_ad = 1.d0,
       lambda_i = 0.d0,
       beta = 0.d0,
       ibprime = 0,
       vmax = 4.5d0,
       nx0 = 10000, &end
&rotat mach = 0.d0,
       uprime = 0.d0,
       gamma_e = 0.d0, &end
&nperi n_tht = 3,
       kymin = 0.05d0,
       m_j   = 1,
       del_c = 0.d0, &end
&confp eps_r    = 0.18d0,
```

(continues on next page)

(continued from previous page)

```
       eps_rnew = 1.d0,
       q_0      = 1.4d0,
       s_hat    = 0.8d0,
       lprd     = 0.d0,
       mprd     = 0.d0,
       eps_hor  = 0.d0,
       eps_mor  = 0.d0,
       eps_por  = 0.d0,
       rdeps00  = 0.d0,
       rdeps1_0 = 1.d0,
       rdeps1_10= 0.d0,
       rdeps2_10= 0.d0,
       rdeps3_10= 0.d0,
       malpha   = 0.d0,    &end

&ring  ring_a = 0.5d0,
       kxmin  = 0.05d0, &end

&vmecp s_input = 0.5d0,
         nss = 501,
         ntheta = 384,
         nzeta  = 0,      &end
&bozxf f_bozx="%%DIR%%/vmec/",  &end

&igsp s_input = 0.5d0,
         mc_type = 0,
         q_type = 1,
         nss = 101,
         ntheta = 49,      &end
&igsf f_igs="%%DIR%%/eqdsk/",  &end

&nu_ref Nref = 4.5d19,
        Lref = 1.7d0,
        Tref = 2.d0,
        col_type = "LB",
        iFLR = 1,
        icheck = 0, &end
```

CHAPTER

FIVE

DIAGNOSTICS

5.1 Output files of GKV

When finishing a run of GKV, all simulation output will be dumped in the output directory `DIR` set in the `run/shoot` script (for example in Listing 4.3, `DIR=/data/maeyama/gkv_training/test01/`), as classified into the following directories,

- `DIR/`
  - `log/` (Log files on simulation runs)
  - `cnt/` (Binary data for restart)
  - `fxv/` (Binary data of distribution functions $tilde{f}_{mathrm{s}bm{k}}(k_x,k_y,v_parallel,mu)$ at several positions of z)
  - `phi/` (Binary data of potentials, fluid moments, and variables in the entropy balance equation)
  - `hst/` (Ascii data of the GKV standard output)
  - ... (Others are back up of the source code and environmental settings)

List of GKV output is summarized in Appendix *List of GKV namelist*.

Users may diagnose these output data by themselves. The present version of GKV provides two post-processing tools, `gkvfig` and `diag_python` on GKV GitHub page.

5.2 gkvfig - Generating PDF of GKV standard ASCII output

gkvfig is a Python package that generates a summary figure PDF of GKV standard ASCII output. (Previous `fig_stdout` tool using shell/awk/gnuplot/latex is converted to Python.)

5.2.1 Installation

To install from PyPI:

```
$ pip install gkvfig
```

Or install the latest development version from GitHub:

```
$ pip install git+https://github.com/GKV-developers/gkvfig.git
```

5.2.2 Usage

(i) Basic usage: As a command line tool

```
$ python -m gkvfig -d DIR
```

The argument `DIR` is the path of GKV output directory. The namelist file `DIR/gkvp.namelist.001`, log file `DIR/log/gkvp.000000.0.log.001`, and hst directory `DIR/hst/` should exist. You get a summary PDF file `CWD/figpdf_yyyymmdd_hhmmss/fig_stdout.pdf`.

(ii) As a Python function

```
from gkvfig import gkvfig

gkvfig(gkv_stdout_dir="YOUR GKV OUTPUT DIR")
```

You get a summary PDF file `CWD/figpdf_yyyymmdd_hhmmss/fig_stdout.pdf`, always in the current working directory `CWD`.

5.2.3 Dependencies

gkvfig requires the following Python packages: - `numpy`, `matplotlib`, `pandas`, `reportlab`, `pypdf`

5.3 diag_python - Post-processing tool for BINARY output

diag_python is a set of Python scripts, which read Zarr format files of GKV binary output. (Python version of the previous Fortran post-processing tool `diag`.)

5.3.1 How to use diag_python

(i) Copy whole `diag_python/` into the output directory of GKV. For example,

- **`YOUR_GKV_EXECUTED_DIR/`**
  - `diag_python/`
  - `cnt/` (Zarr format files *.zarr will be read by diag_python)
  - `fxv/` (Zarr format files *.zarr will be read by diag_python)
  - `phi/` (Zarr format files *.zarr will be read by diag_python)
  - `hst/` (gkvp.mtr.001 will be read by diag_python)
  - `src/` (gkvp_header.f90 will be read by diag_python)
  - `log/`
  - `gkvp_namelist.001` (gkvp_namelist.001 will be read by diag_python)

(ii) Initial settings in `main.py`:

```
import sys
sys.path.append("./src/")
from diag_rb import rb_open, rb_get_tri_filelist
from diag_geom import geom_set
### Read Zarr store gkvp.phi.*.zarr/ by xarray ###
xr_phi = rb_open('../phi/gkvp.phi.*.zarr/')
```

(continues on next page)

(continued from previous page)

```
xr_Al  = rb_open('../phi/gkvp.Al.*.zarr/')
xr_mom = rb_open('../phi/gkvp.mom.*.zarr/')
xr_fxv = rb_open('../fxv/gkvp.fxv.*.zarr/')
xr_cnt = rb_open('../cnt/gkvp.cnt.*.zarr/')
xr_trn = rb_open('../phi/gkvp.trn.*.zarr/')
tri_filelist = rb_get_tri_filelist('../phi/gkvp.tri.*.zarr/')
xr_tri_list=[]
for file in tri_filelist:
    xr_tri=rb_open(file + '.*.zarr/')
    xr_tri_list.append(xr_tri)
### Set geometric constants ###
geom_set(headpath='../src/gkvp_header.f90', nmlpath="../gkvp_namelist.001", mtrpath='../
↪hst/gkvp.mtr.001')
```

(iii) Call functions, e.g.:

```
from out_mominxy import phiinxy
# Plot phi[y,x] at t[it], zz[iz]
it = 3
iz = 8
phiinxy(it, iz, xr_phi, flag="display")
```

See help(phiinxy) for details.

For more details, Appendix *Diagnostics modules in the post-processing program diag* shows some examples of diagnostics modules.

APPENDIX

A

APPENDIX

A.1 List of GKV namelist

Table A.1: List of run/gkvp_namelist

Group	Name	Parameter
&cmemo	memo	Memo
&calct	calc_type	• "linear" – for linear runs • "lin_freq" – for linear runs with frequency check $k_x = 0$ • "nonlinear" – for nonlinear runs
&calct	z_bound	• "zerofixed" – Fixed boundary in z • "mixed" – Outflow boundary in z only for $\tilde{f}_{s\boldsymbol{k}}$
&calct	z_filt	• "on" – Enable 4th-order filtering in z on $d\tilde{f}_{s\boldsymbol{k}}/dt$ • "off" — Disable filtering
&calct	z_calc	• "cf4" – 4th-order central finite difference for $d\tilde{f}_{s\boldsymbol{k}}/dz$ (nzb $= 2$) • "up5" – 5th-order upwind finite difference for $d\tilde{f}_{s\boldsymbol{k}}/dz$ (nzb $= 3$)
&calct	art_diff	Coefficient of artificial diffusion for z_calc="cf4"
&calct	init_random	Switch whether phases of initial Fourier modes are randomized
&calct	num_triad_diag	Number of triad transfer diagnostics, which should be consistent with the number of "&triad mxt=*, myt=*/".
&calct	vp_coord	• "0" — the magnetic moment mu is the perpendicular velocity-space coordinate • "1" — the perpendicular velocity vp is the perpendicular velocity-space coordinate
&triad	mxt=*, myt=*/	Diagnosed mode number of triad transfer analysis. Add lines of "&triad mxt=*,myt=*/" as desire.

continues on next page

Table A.1 – continued from previous page

Group	Name	Parameter
&equib	equib_type	• "analytic" – Analytic helical field with the metrics in cylinder • "s-alpha" – s-alpha model with alpha = 0 (cylindrical metrics) • "s-alpha-shift" – s-alpha model with Shafranov shift • "circ-MHD" – Concentric circular field with consistent metrics • "vmec" – Tokamak/stellarator field from the VMEC code • "eqdsk" – Tokamak field (MEUDAS/TOPICS or G-EQDSK) via IGS code • "slab" – Shearless slab geometry • "ring" – Ring dipole geometry
&run_n	inum	Current run number
&run_n	ch_res	• .true. – Change perpendicular resolutions (editing `gkvp_f0.48_set.f90` is required.) • .false. – Disable changing resolution
&files	f_log	Data directory for log data
&files	f_hst	Data directory for time-series data
&files	f_phi	Data directory for field quantity data
&files	f_fxv	Data directory for distribution function data
&files	f_cnt	Data directory for continue data
&runlm	e_limit	Elapsed time limit [sec]
&times	tend	End of simulation time [L_ref/v_ref]
&times	dtout_fxv	Time spacing for data output [L_ref/v_ref]
&times	dtout_ptn	Time spacing for data output [L_ref/v_ref]
&times	dtout_eng	Time spacing for data output [L_ref/v_ref]
&times	dtout_dtc	Time spacing for time-step-size adaption [L_ref/v_ref]
&deltt	dt_max	Maximum time step size [L_ref/v_ref]
&deltt	adapt_dt	• .true. – Enable time-step-size adaption • .false. – Time step size is fixed to be dt = dt_max
&deltt	courant_num	Courant number for time-step-size adaption

&deltt	time_advnc	• "rkg4" – Explicit time integration by 4th-order Runge-Kutta-Gill method
		• "imp_colli" – 2nd-order operator split + 2nd-order implicit collision solver + 4th-order RKG method for collisionless physics
		• "auto_init" –
		  – If collision restricts linear time step size, time_advnc="imp_colli".
		  – Otherwise, time_advnc="rkg4"

&physp	R0_Ln	Normalized density gradient, L_ref/L_ne, L_ref/L_ni, …
&physp	R0_Lt	Normalized temperature gradient, L_ref/L_te, L_ref/L_ti, …
&physp	nu	Bias factor for LB collision model, e.g., 1.d0, 0.5d0, 2.d0, … **Note** NOTE that after gkvp_f0.40, collision frequencies are consistently calculated by (Nref, Tref, Lref) in &nu_ref, and nu is just used as a bias factor only for LB case. Also, nu is not used in multi-species collisions (full).
&physp	Anum	Mass number, m_e/m_ref, m_i/m_ref, …
&physp	Znum	Atomic number, |e_e/e_ref|, |e_i/e_ref|, …

continues on next page

Table A.1 – continued from previous page

Group	Name	Parameter
&physp	fcs	Charge fraction |e_e*n_e/(e_ref*n_ref)|, |e_i*n_i/(e_ref*n_ref)|, … **Note** NOTE that fcs = 1.0 for electron in the recommended setting (n_ref = n_e).
&physp	sgn	Sign of charge, e_e/ |e_e|, e_i/ |e_i|, …
&physp	tau	Normalized temperature, T_e/T_ref, T_i/T_ref, … **Note** NOTE that T_i/T_ref = 1.0 for the first ion species in the recommended setting (T_ref = T_i of first ion).
&physp	dns1	Initial perturbation amplitude, (L_ref/rho_ref)* $\tilde{n}_e$/n_ref, (L_ref/rho_ref)* $\tilde{n}_i$/n_ref, …
&physp	tau_ad	T_i/T_e for single species ITG-ae (sgn=+1), T_e/T_i for single species ETG-ai (sgn=-1)
&physp	lambda_i	Ratio of (Debye_length / rho_ref)**2 = epsilon_0 * B_ref**2 / (m_ref * n_ref)
&physp	beta	Local beta value evaluated by mu_0*n_ref*T_ref/B_ref**2
&physp	ibprime	• "1" – Enable a grad-p (finite beta-prime) contribution on the magnetic drift kvd for equib_type = "eqdsk" and "vmec" • "0" – Ignore it
&physp	vmax	Velocity domain size in the unit of each thermal speed [v_ts]
&physp	nx0	Radial mode number assigned for the initial perturbation **Note** NOTE that if nx0 exceeds nx, nx0 is reset to nx. A sufficiently large value, thus, gives perturbations for entire kx-modes.
&rotat	mach	Not yet implemented
&rotat	uprime	Not yet implemented
&rotat	gamma_e	Equilibrium $\boldsymbol{E} \times \boldsymbol{B}$ flow shearing rate γ_E
&nperi	n_tht	The length of fluxtube, z-domain $= \pm$ N_tht* π
&nperi	kymin	Minimum field-line-label (or poloidal) wave number [1/rho_ref]
&nperi	m_j	Mode connection number for pseudo-periodic boundary in fluxtube, kxmin = |2*pi*s_hat*kymin/m_j|
&nperi	del_c	Mode connection phase in fluxtube model (Since it is arbitrary, del_c = 0.d0 in standard.)
&confp	eps_r	Inverse aspect ratio at the center of fluxtube, a*rho_0/L_ref
&confp	eps_rnew	Model factor for equib_type = "analytic"
&confp	q_0	Safety factor at the center of fluxtube, q(rho_0)
&confp	s_hat	Magnetic shear at the center of fluxtube, s(rho_0)
&confp	lprd ⋮ malpha	factor for equib_type = "analytic"

continues on next page

Table A.1 – continued from previous page

Group	Name	Parameter
&ring	ring_a	ring_a = a / R0, which specify a flux tube of the ring dipole. [There is a ring current at R=a. The field line passing through (R,Z)=(R0,0) is picked up as a flux-tube domain. The reference length is set to be R0 (not the ring current at R=a). The reference magnetic field strength is B0 at (R,Z)=(R0,0).]
&ring	kxmin	Minimum wavenumber in kx, valid only when equib_type == "ring"
&vmecp	s_input	Minor radial position of the local flux-tube analysis in Stellarator (VMEC) equilibrium?
&vmecp	nss	Number of radial grids on METRIC data (=nrho in BZX)
&vmecp	ntheta	ntheta = (number of poloidal grids on METRIC = ntht in BZX) = 2*global_nz
&vmecp	nzeta	= 0
&bozxf	f_bozx	File location of METRIC data produced by BZX code
&igsp	s_input	Reference radial flux surface, rho_0, in Tokamak (MEUDAS/TOPICS or G-EQDSK) equilibrium
&igsp	mc_type	• "0" – Axisymmetric coordinates • "1" – Boozer coordinates • "2" – Hamada coordinates
&igsp	q_type	• "1" – Use consistent q-value on g-eqdsk equilibrium (Recommended) • "0" – Use inconsistent, but given q_0 value in &confp.
&igsp	nss	Number of radial grids on METRIC data
&igsp	ntheta	ntheta = (number of poloidal grids on METRIC = ntht in BZX) = 2*global_nz
&igsf	f_igs	File location of METRIC data produced by IGS code
&nu_ref	Nref	Local electron density at the center of fluxtube, $n_{\rm e}(\rho_0)$ [m^-3]
&nu_ref	Lref	Major radius at the magnetic axis, R_a [m]
&nu_ref	Tref	Main ion temperature at the center of fluxtube $T_i(\rho_0)$ [keV]
&nu_ref	col_type	• "LB" – Lenard-Bernstein model collision operator • "lorentz" – Lorentz model collision operator • "full" – Sugama model collision operator for multiple plasma species
&nu_ref	iFLR	• "1" – Enable the FLR (gyrophase-averaging and classical diffusion) terms (for LB and full) • "0" – Disable it (DK-limit)
&nu_ref	icheck	• "0" – for production runs • "1" – Debug test with Maxwellian Annihilation (should be used with iFLR = 0)

Note

Note that `inum=%%%` and `f_**="%%DIR%%/..."` will be automatically set by the `shoot` script. In the `&physp` group, species-dependent names `R0_Ln - dns1` are the array of length `nprocs`. The `&vmecp` and `&bozxf` groups are active only when `equib_type = "vmec"`. Similarly, the `&igsp` and `&igsf` groups are active only when `equib_type = "eqdsk"`.

A.2 Use of MHD equilibrium interfaces

In preparation.

A.2.1 Use of IGS (EQDSK for Tokamaks)

In preparation.

A.2.2 Use of BZX (VMEC for Stellarators)

In preparation.

A.3 List of GKV output

GKV output files are:

- The output directory `DIR/`
  - `cnt/*cnt*`
  - `fxv/*fxv*`
  - `phi/\*phi\*, \*Al\*, \*mom\*, \*trn\*, (\*tri\* for nonlinear runs)`
  - `hst/\*bln\*, \*geq\*, \*gem\*, \*qes\*, \*qem\*, \*wes\*, \*wem\*, \*eng\*, \*men\*, \*dtc\*, \*mtr\*, (\*frq\*, \*dsp\* for linear runs)`
  - `log/*log*`

Their explanations are summarized below.

Explanations on GKV output files

Table A.2: cnt/gkvp.cnt.(inum in 3 digits).zarr/

Field	Description
File type	`Zarr format binary`
Output timing	`End of the run`
MPI ranks	`All`
Total files	`nprocw*nprocz*nprocv*nprocm*nprocs*(Total run numbers)`
GKV unit	`ocnt`
Stored data	Coordinates: • **t** Simulation time • **is** particle species (integer) • **mu** magnetic moment (or **vp** perpendicular velocity) • **vl** parallel velocity • **zz** field-aligned coordinate • **ky** Field-line-label (poloidal) wavwenumber • **kx** radial wavenumber Data variables: **cnt[t,is,mu,vl,zz,ky,kx]** Perturbed distribution function $\tilde{f}_{\mathrm{sk}}$ (double complex).

Table A.3: fxv/gkvp.fxv.(inum in 3 digits).zarr/

Field	Description
File type	`Zarr format binary`
Output timing	`dtout_fxv`
MPI ranks	`All`
Total files	`nprocw*nprocz*nprocv*nprocm*nprocs*(Total run numbers)`
GKV unit	`ofxv`
Stored data	Coordinates: • **t** Simulation time, • **is** particle species (integer) • **mu** magnetic moment (or **vp** perpendicular velocity) • **vl** parallel velocity, **zz** field-aligned coordinate • **ky** Field-line-label (poloidal) wavwenumber • **kx** radial wavenumber Data variables: **fxv[t,is,mu,vl,zz,ky,kx]** Perturbed distribution function $\tilde{f}_{\mathbf{sk}}$ (double complex) at `iz = -nz` in each `rankz`.

Table A.4: phi/gkvp.phi.(inum in 3 digits).zarr/

Field	Description
File type	`Zarr format binary`
Output timing	`dtout_ptn`
MPI ranks	`ranks == 0 .and. vel_rank == 0`
Total files	`nprocw*nprocz*(Total run numbers)`
GKV unit	`ophi`
Stored data	Coordinates: • **t** Simulation time • **zz** field-aligned coordinate • **ky** Field-line-label (poloidal) wavwenumber • **kx** radial wavenumber Data variables: **phi[t,zz,ky,kx]** Perturbed electrostatic potential $\tilde{\phi}_{\mathbf{k}}$.

Table A.5: phi/gkvp.Al.(inum in 3 digits).zarr/

Field	Description
File type	`Zarr format binary`
Output timing	`dtout_ptn`
MPI ranks	`ranks == 0 .and. vel_rank == 0`
Total files	`nprocw*nprocz*(Total run numbers)`
GKV unit	`oAl`
Stored data	Coordinates: • **t** Simulation time • **zz** field-aligned coordinate • **ky** Field-line-label (poloidal) wavwenumber • **kx** radial wavenumber Data variables: **Al[t,zz,ky,kx]** Perturbed vector potential $\tilde{A}_{\parallel\mathbf{k}}$.

Table A.6: phi/gkvp.mom.(inum in 3 digits).zarr/

Field	Description
File type	`Zarr format binary`
Output timing	`dtout_ptn`
MPI ranks	`vel_rank == 0`
Total files	`nprocw*nprocz*nprocs*(Total run numbers)`
GKV unit	`omom`
Stored data	Coordinates: • **t** Simulation time • **is** particle species (integer) • **imom** index of fluid moments (integer) • **zz** field-aligned coordinate • **ky** Field-line-label (poloidal) wavwenumber • **kx** radial wavenumber Data variables: **mom[t,is,imom,zz,ky,kx]** Perturbed fluid moment (double complex). In the present version `nmom = 6`: • $\tilde{n}_{\mathrm{s}\mathbf{k}} = \int dv^3\, J_{0\mathrm{s}\mathbf{k}}\, \tilde{f}_{\mathrm{s}\mathbf{k}}$ • $\tilde{u}_{\parallel\mathrm{s}\mathbf{k}} = \int dv^3\, v_\parallel\, J_{0\mathrm{s}\mathbf{k}}\, \tilde{f}_{\mathrm{s}\mathbf{k}}$ • $\tilde{p}_{\parallel\mathrm{s}\mathbf{k}} = \int dv^3\, \frac{m_\mathrm{s} v_\parallel^2}{2}\, J_{0\mathrm{s}\mathbf{k}}\, \tilde{f}_{\mathrm{s}\mathbf{k}}$ • $\tilde{p}_{\perp\mathrm{s}\mathbf{k}} = \int dv^3\, \mu B\, J_{0\mathrm{s}\mathbf{k}}\, \tilde{f}_{\mathrm{s}\mathbf{k}}$ • $\tilde{q}_{\parallel\parallel\mathrm{s}\mathbf{k}} = \int dv^3\, v_\parallel \frac{m_\mathrm{s} v_\parallel^2}{2}\, J_{0\mathrm{s}\mathbf{k}}\, \tilde{f}_{\mathrm{s}\mathbf{k}}$ • $\tilde{q}_{\parallel\perp\mathrm{s}\mathbf{k}} = \int dv^3\, v_\parallel \mu B\, J_{0\mathrm{s}\mathbf{k}}\, \tilde{f}_{\mathrm{s}\mathbf{k}}$ • Normalized by $\delta_\mathrm{ref} n_\mathrm{ref}$, $\delta_\mathrm{ref} n_\mathrm{ref} v_\mathrm{ref}$, $\delta_\mathrm{ref} n_\mathrm{ref} T_\mathrm{ref}$, $\delta_\mathrm{ref} n_\mathrm{ref} T_\mathrm{ref}$, $\delta_\mathrm{ref} n_\mathrm{ref} T_\mathrm{ref} v_\mathrm{ref}$, $\delta_\mathrm{ref} n_\mathrm{ref} T_\mathrm{ref} v_\mathrm{ref}$, respectively.

Table A.7: phi/gkvp.trn.(inum in 3 digits).zarr/

Field	Description
File type	`Zarr format binary`
Output timing	`dtout_eng`
MPI ranks	`zsp_rank == 0 .and. vel_rank == 0`
Total files	`nprocw*nprocs*(Total run numbers)`
GKV unit	`otrn`
Stored data	Coordinates: • **t** Simulation time • **is** particle species (integer) • **itrn** index of entropy balance diagnostics (integer) • **ky** Field-line-label (poloidal) wavwenumber • **kx** radial wavenumber Data variables: **trn[t,is,itrn,ky,kx]** • $S_{\mathrm{s}\mathbf{k}}(-nx:nx,0:ny)$: Perturbed gyrocenter entropy $[\delta^2_{\mathrm{ref}} n_{\mathrm{ref}} T_{\mathrm{ref}}]$ (real*8) • $W_{\mathrm{E}\mathbf{k}}(-nx:nx,0:ny)$: Electrostatic field energy including polarization $[\delta^2_{\mathrm{ref}} n_{\mathrm{ref}} T_{\mathrm{ref}}]$ (real*8) • $W_{\mathrm{M}\mathbf{k}}(-nx:nx,0:ny)$: Magnetic field energy $[\delta^2_{\mathrm{ref}} n_{\mathrm{ref}} T_{\mathrm{ref}}]$ (real*8) • $R_{\mathrm{sE}\mathbf{k}}(-nx:nx,0:ny)$: Wave–particle interaction $(W_{\mathrm{E}\mathbf{k}} \to S_{\mathrm{s}\mathbf{k}})$ $[\delta^2_{\mathrm{ref}} n_{\mathrm{ref}} T_{\mathrm{ref}} v_{\mathrm{ref}}/L_{\mathrm{ref}}]$ (real*8) • $R_{\mathrm{sM}\mathbf{k}}(-nx:nx,0:ny)$: Wave–particle interaction $(W_{\mathrm{M}\mathbf{k}} \to S_{\mathrm{s}\mathbf{k}})$ $[\delta^2_{\mathrm{ref}} n_{\mathrm{ref}} T_{\mathrm{ref}} v_{\mathrm{ref}}/L_{\mathrm{ref}}]$ (real*8) • $I_{\mathrm{sE}\mathbf{k}}(-nx:nx,0:ny)$: Nonlinear entropy transfer by $\mathbf{E}\times\mathbf{B}$ flow $[\delta^2_{\mathrm{ref}} n_{\mathrm{ref}} T_{\mathrm{ref}} v_{\mathrm{ref}}/L_{\mathrm{ref}}]$ (real*8) • $I_{\mathrm{sM}\mathbf{k}}(-nx:nx,0:ny)$: Nonlinear entropy transfer by magnetic flutter $[\delta^2_{\mathrm{ref}} n_{\mathrm{ref}} T_{\mathrm{ref}} v_{\mathrm{ref}}/L_{\mathrm{ref}}]$ (real*8) • $D_{\mathrm{s}\mathbf{k}}(-nx:nx,0:ny)$: Collisional dissipation $[\delta^2_{\mathrm{ref}} n_{\mathrm{ref}} T_{\mathrm{ref}} v_{\mathrm{ref}}/L_{\mathrm{ref}}]$ (real*8) • $\Gamma_{\mathrm{sE}\mathbf{k}}(-nx:nx,0:ny)$: Particle flux by $\mathbf{E}\times\mathbf{B}$ flow $[\delta^2_{\mathrm{ref}} n_{\mathrm{ref}} v_{\mathrm{ref}}]$ (real*8) • $\Gamma_{\mathrm{sM}\mathbf{k}}(-nx:nx,0:ny)$: Particle flux by magnetic flutter $[\delta^2_{\mathrm{ref}} n_{\mathrm{ref}} v_{\mathrm{ref}}]$ (real*8) • $Q_{\mathrm{sE}\mathbf{k}}(-nx:nx,0:ny)$: Energy flux by $\mathbf{E}\times\mathbf{B}$ flow $[\delta^2_{\mathrm{ref}} n_{\mathrm{ref}} T_{\mathrm{ref}} v_{\mathrm{ref}}]$ (real*8) • $Q_{\mathrm{sM}\mathbf{k}}(-nx:nx,0:ny)$: Energy flux by magnetic flutter $[\delta^2_{\mathrm{ref}} n_{\mathrm{ref}} T_{\mathrm{ref}} v_{\mathrm{ref}}]$ (real*8) *See also Supplemental Entropy balance equation for each wavenumber and plasma species.*

Table A.8: phi/gkvp.tri.mx(mxt in 4 digits)my(myt in 4 digits).(inum in 3 digits).zarr/

Field	Description
File type	`Zarr format binary`
Output timing	`dtout_ptn` (when `calc_type == "nonlinear"` and `num_triad_diag > 0`)
MPI ranks	`rank == 0`
Total files	`nprocs*num_triad_diag*(Total run numbers)`
GKV unit	`otri`
Stored data	Coordinates: • **t** Simulation time • **is** particle species (integer) • **itri** index of triad transfer diagnostics (integer) • **ky** Field-line-label (poloidal) wavwenumber • **kx** radial wavenumber Data variables: **tri[t,is,itri,ky,kx]** • $J^{\mathbf{p},\mathbf{q}}_{\mathrm{sE}\mathbf{k}}(-nx:nx,-global_y:global_y)$: Triad transfer function from modes $\mathbf{p},\mathbf{q}$ to mode $\mathbf{k}$ via $\mathbf{E}\times\mathbf{B}$ nonlinearity $[\delta^2_{\mathrm{ref}} n_{\mathrm{ref}} T_{\mathrm{ref}} v_{\mathrm{ref}}/L_{\mathrm{ref}}]$ (real*8) • $J^{\mathbf{q},\mathbf{k}}_{\mathrm{sE}\mathbf{p}}(-nx:nx,-global_y:global_y)$: Cyclic change $(\mathbf{k},\mathbf{p},\mathbf{q})\to(\mathbf{p},\mathbf{q},\mathbf{k})$ (real*8) • $J^{\mathbf{k},\mathbf{p}}_{\mathrm{sE}\mathbf{q}}(-nx:nx,-global_y:global_y)$: Cyclic change $(\mathbf{p},\mathbf{q},\mathbf{k})\to(\mathbf{q},\mathbf{k},\mathbf{p})$ (real*8) • $J^{\mathbf{p},\mathbf{q}}_{\mathrm{sM}\mathbf{k}}(-nx:nx,-global_y:global_y)$: Triad transfer function from modes $\mathbf{p},\mathbf{q}$ to mode $\mathbf{k}$ via magnetic-flutter nonlinearity $[\delta^2_{\mathrm{ref}} n_{\mathrm{ref}} T_{\mathrm{ref}} v_{\mathrm{ref}}/L_{\mathrm{ref}}]$ (real*8) • $J^{\mathbf{q},\mathbf{k}}_{\mathrm{sM}\mathbf{p}}(-nx:nx,-global_y:global_y)$: Cyclic change $(\mathbf{k},\mathbf{p},\mathbf{q})\to(\mathbf{p},\mathbf{q},\mathbf{k})$ (real*8) • $J^{\mathbf{k},\mathbf{p}}_{\mathrm{sM}\mathbf{q}}(-nx:nx,-global_y:global_y)$: Cyclic change $(\mathbf{p},\mathbf{q},\mathbf{k})\to(\mathbf{q},\mathbf{k},\mathbf{p})$ (real*8) Diagnosed for a fixed mode $\mathbf{k}=$ (`mxt,myt`) and plotted as a 2D function of $\mathbf{p}=(p_x,p_y)$, where the triad condition determines $\mathbf{q}=-\mathbf{k}-\mathbf{p}$. *See also Supplemental Triad transfer function.*

Table A.9: hst/gkvp_f0.48.bln.(ranks in 1 digits).(inum in 3 digits)

Field	Description
File type	`Ascii`
Output timing	`dtout_eng`
MPI ranks	`rank == 0`
Total files	`nprocs*(Total run numbers)`
GKV unit	`obln`
Stored data	time, $S_{\rm s}$, $W_{\rm E}$, $W_{\rm M}$, $R_{\rm sE}$, $R_{\rm sM}$, $I_{\rm sE}$, $I_{\rm sM}$, $D_{\rm s}$, $\frac{T_{\rm s}\Gamma_{\rm sE}}{L_{ps}}$, $\frac{T_{\rm s}\Gamma_{\rm sM}}{L_{ps}}$, $\frac{\Theta_{\rm sE}}{L_{T\rm s}}$, $\frac{\Theta_{\rm sM}}{L_{T\rm s}}$ where, • **time**: Simulation time t [$L_{\rm ref}/v_{\rm ref}$] (real*8) • $S_{\rm s}$ (0:1): Perturbed gyrocenter entropy [$\delta^2_{\rm ref} n_{\rm ref} T_{\rm ref}$] (real*8). • $W_{\rm E}$ (0:1): Electrostatic field energy including polarization [$\delta^2_{\rm ref} n_{\rm ref} T_{\rm ref}$] (real*8). • $W_{\rm M}$ (0:1): Magnetic field energy [$\delta^2_{\rm ref} n_{\rm ref} T_{\rm ref}$] (real*8). • $R_{\rm sE}$ (0:1): Wave-particle interaction ($W_{{\rm E}\boldsymbol{k}} \to S_{{\rm s}\boldsymbol{k}}$) [$\delta^2_{\rm ref} n_{\rm ref} T_{\rm ref} v_{\rm ref}/L_{\rm ref}$] (real*8). • $R_{\rm sM}$ (0:1): Wave-particle interaction ($W_{{\rm M}\boldsymbol{k}} \to S_{{\rm s}\boldsymbol{k}}$) [$\delta^2_{\rm ref} n_{\rm ref} T_{\rm ref} v_{\rm ref}/L_{\rm ref}$] (real*8). • $I_{\rm sE}$ (0:1): Nonlinear entropy transfer by $\boldsymbol{E}\times\boldsymbol{B}$ flow [$\delta^2_{\rm ref} n_{\rm ref} T_{\rm ref} v_{\rm ref}/L_{\rm ref}$] (real*8). • $I_{\rm sM}$ (0:1): Nonlinear entropy transfer by magnetic flutter [$\delta^2_{\rm ref} n_{\rm ref} T_{\rm ref} v_{\rm ref}/L_{\rm ref}$] (real*8). • $D_{\rm s}$ (0:1): Collisional dissipation [$\delta^2_{\rm ref} n_{\rm ref} T_{\rm ref} v_{\rm ref}/L_{\rm ref}$] (real*8). • $\frac{T_{\rm s}\Gamma_{\rm sE}}{L_{ps}}$: Particle flux term by $\boldsymbol{E}\times\boldsymbol{B}$ flow [$\delta^2_{\rm ref} n_{\rm ref} T_{\rm ref} v_{\rm ref}/L_{\rm ref}$] (real*8). • $\frac{T_{\rm s}\Gamma_{\rm sM}}{L_{ps}}$: Particle flux term by magnetic flutter [$\delta^2_{\rm ref} n_{\rm ref} T_{\rm ref} v_{\rm ref}/L_{\rm ref}$] (real*8). • $\frac{\Theta_{\rm sE}}{L_{T\rm s}}$: Heat flux term by $\boldsymbol{E}\times\boldsymbol{B}$ flow [$\delta^2_{\rm ref} n_{\rm ref} T_{\rm ref} v_{\rm ref}/L_{\rm ref}$] (real*8). • $\frac{\Theta_{\rm sM}}{L_{T\rm s}}$: Heat flux term by magnetic flutter [$\delta^2_{\rm ref} n_{\rm ref} T_{\rm ref} v_{\rm ref}/L_{\rm ref}$] (real*8). The 0th and 1st components of $S_{\rm s}$ – $D_{\rm s}$ correspond to non-zonal ($k_y \neq 0$) and zonal ($k_y = 0$) fluctuations, respectively.

Table A.10: hst/gkvp_f0.48.ges.(ranks in 1 digits).(inum in 3 digits)

Field	Description
File type	`Ascii`
Output timing	`dtout_eng`
MPI ranks	`rank == 0`
Total files	`nprocs*(Total run numbers)`
GKV unit	`oges`
Stored data	time, $\Gamma_{\rm sE}$, $\Gamma_{{\rm sE}k_y}$ where, • **time**: Simulation time t [$L_{\rm ref}/v_{\rm ref}$] (real) • $\Gamma_{\rm sE}$: Total particle flux by $\boldsymbol{E}\times\boldsymbol{B}$ flow [$\delta^2_{\rm ref} n_{\rm ref} v_{\rm ref}$] (real). • $\Gamma_{{\rm sE}k_y}$ (0:global_ny): k_y spectrum of the particle flux by $\boldsymbol{E}\times\boldsymbol{B}$ flow [$\delta^2_{\rm ref} n_{\rm ref} v_{\rm ref}$] (real).

Table A.11: hst/gkvp.gem.(ranks in 1 digits).(inum in 3 digits)

Field	Description
File type	`Ascii`
Output timing	`dtout_eng`
MPI ranks	`rank == 0`
Total files	`nprocs*(Total run numbers)`
GKV unit	`ogem`
Stored data	time, Γ_{sM}, $\Gamma_{\mathrm{sM}k_y}$ where, • **time**: Simulation time t [$L_{\mathrm{ref}}/v_{\mathrm{ref}}$] (real) • Γ_{sM}: Total particle flux by magnetic flutter [$\delta^2_{\mathrm{ref}} n_{\mathrm{ref}} v_{\mathrm{ref}}$] (real). • $\Gamma_{\mathrm{sM}k_y}$ (0:global_ny): k_y spectrum of the particle flux by magnetic flutter [$\delta^2_{\mathrm{ref}} n_{\mathrm{ref}} v_{\mathrm{ref}}$] (real).

Table A.12: hst/gkvp.qes.(ranks in 1 digits).(inum in 3 digits)

Field	Description
File type	`Ascii`
Output timing	`dtout_eng`
MPI ranks	`rank == 0`
Total files	`nprocs*(Total run numbers)`
GKV unit	`oqes`
Stored data	time, Q_{sE}, $Q_{\mathrm{sE}k_y}$ where, • **time**: Simulation time t [$L_{\mathrm{ref}}/v_{\mathrm{ref}}$] (real) • Q_{sE}: Total energy flux by $\boldsymbol{E} \times \boldsymbol{B}$ flow [$\delta^2_{\mathrm{ref}} n_{\mathrm{ref}} T_{\mathrm{ref}} v_{\mathrm{ref}}$] (real). • $Q_{\mathrm{sE}k_y}$ (0:global_ny): k_y spectrum of the energy flux by $\boldsymbol{E} \times \boldsymbol{B}$ flow [$\delta^2_{\mathrm{ref}} n_{\mathrm{ref}} T_{\mathrm{ref}} v_{\mathrm{ref}}$] (real).

Table A.13: hst/gkvp.qem.(ranks in 1 digits).(inum in 3 digits)

Field	Description
File type	`Ascii`
Output timing	`dtout_eng`
MPI ranks	`rank == 0`
Total files	`nprocs*(Total run numbers)`
GKV unit	`oqem`
Stored data	time, Q_{sM}, $Q_{\mathrm{sM}k_y}$ where, • **time**: Simulation time t [$L_{\mathrm{ref}}/v_{\mathrm{ref}}$] (real) • Q_{sM}: Total energy flux by magnetic flutter [$\delta^2_{\mathrm{ref}} n_{\mathrm{ref}} T_{\mathrm{ref}} v_{\mathrm{ref}}$] (real). • $Q_{\mathrm{sM}k_y}$ (0:global_ny): k_y spectrum of the energy flux by magnetic flutter [$\delta^2_{\mathrm{ref}} n_{\mathrm{ref}} T_{\mathrm{ref}} v_{\mathrm{ref}}$] (real).

Table A.14: hst/gkvp.wes.(inum in 3 digits)

Field	Description
File type	`Ascii`
Output timing	`dtout_eng`
MPI ranks	`rankg == 0`
Total files	`(Total run numbers)`
GKV unit	`owes`
Stored data	time, $W_{\rm E}$, $W_{{\rm E}k_y}$ where, • **time**: Simulation time t [$L_{\rm ref}/v_{\rm ref}$] (real) • $W_{\rm E}$: Total electrostatic field energy including polarization [$\delta_{\rm ref}^2 n_{\rm ref} T_{\rm ref}$] (real). • $W_{{\rm E}k_y}$ (0:global_ny): k_y spectrum of the electrostatic field energy [$\delta_{\rm ref}^2 n_{\rm ref} T_{\rm ref}$] (real).

Table A.15: hst/gkvp.wem.(inum in 3 digits)

Field	Description
File type	`Ascii`
Output timing	`dtout_eng`
MPI ranks	`rankg == 0`
Total files	`(Total run numbers)`
GKV unit	`owem`
Stored data	time, $W_{\rm M}$, $W_{{\rm M}k_y}$ where, • **time**: Simulation time t [$L_{\rm ref}/v_{\rm ref}$] (real) • $W_{\rm M}$: Total magnetic field energy [$\delta_{\rm ref}^2 n_{\rm ref} T_{\rm ref}$] (real). • $W_{{\rm M}k_y}$ (0:global_ny): k_y spectrum of the magnetic field energy [$\delta_{\rm ref}^2 n_{\rm ref} T_{\rm ref}$] (real).

Table A.16: hst/gkvp.eng.(inum in 3 digits)

Field	Description
File type	`Ascii`
Output timing	`dtout_eng`
MPI ranks	`rankg == 0`
Total files	`(Total run numbers)`
GKV unit	`oeng`
Stored data	time, $\sum_{k_x,k_y}\langle\|\tilde{\phi}_{\boldsymbol{k}}\|^2\rangle$, $\sum_{k_x}\langle\|\tilde{\phi}_{\boldsymbol{k}}\|^2\rangle$ where, • **time**: Simulation time t [$L_{\rm ref}/v_{\rm ref}$] (real) • $\sum_{k_x,k_y}\langle\|\tilde{\phi}_{\boldsymbol{k}}\|^2\rangle$: Squared amplitude of the perturbed electrostatic potential [$(\delta_{\rm ref} T_{\rm ref}/e_{\rm ref})^2$] (real). • $\sum_{k_x}\langle\|\tilde{\phi}_{\boldsymbol{k}}\|^2\rangle$ (0:global_ny): k_y spectrum of the squared amplitude of the perturbed electrostatic potential [$(\delta_{\rm ref} T_{\rm ref}/e_{\rm ref})^2$] (real).

Table A.17: hst/gkvp.men.(inum in 3 digits)

Field	Description
File type	`Ascii`
Output timing	`dtout_eng`
MPI ranks	`rankg == 0`
Total files	`(Total run numbers)`
GKV unit	`omen`
Stored data	time, $\sum_{k_x,k_y}\langle\|\tilde{A}_{\parallel \boldsymbol{k}}\|^2\rangle$, $\sum_{k_x}\langle\|\tilde{A}_{\parallel \boldsymbol{k}}\|^2\rangle$ where, • **time**: Simulation time t [$L_{\rm ref}/v_{\rm ref}$] (real) • $\sum_{k_x,k_y}\langle\|\tilde{A}_{\parallel \boldsymbol{k}}\|^2\rangle$: Squared amplitude of the perturbed electrostatic potential [$(\delta_{\rm ref}\rho_{\rm ref}B_{\rm ref})^2$] (real). • $\sum_{k_x}\langle\|\tilde{A}_{\parallel \boldsymbol{k}}\|^2\rangle$ (0:global_ny): k_y spectrum of the squared amplitude of the perturbed electrostatic potential [$(\delta_{\rm ref}\rho_{\rm ref}B_{\rm ref})^2$] (real).

Table A.18: hst/gkvp.dtc.(inum in 3 digits)

Field	Description
File type	*Ascii*
Output timing	*dtout_dtc*
MPI ranks	*rankg == 0*
Total files	*(Total run numbers)*
GKV unit	*odtc*
Stored data	time, dt, dt_limit, dt_nl where, • **time**: Simulation time t [$L_{\rm ref}/v_{\rm ref}$] (real) • **dt**: Time step size [$L_{\rm ref}/v_{\rm ref}$] (real) • **dt_limit**: Estimation of time step size limit [$L_{\rm ref}/v_{\rm ref}$] (real) • **dt_nl**: Estimation of time step size limit from nonlinear advection [$L_{\rm ref}/v_{\rm ref}$] (real)

Table A.19: hst/gkvp.mtr.(inum in 3 digits)

Field	Description
File type	`Ascii`
Output timing	`Beginning of the run`
MPI ranks	`rankg == 0`
Total files	`(Total run numbers)`
GKV unit	`omtr`
Stored data	time, θ (or φ), B, $\frac{\partial B}{\partial x}$, $\frac{\partial B}{\partial y}$, $\frac{\partial B}{\partial z}$, $g^{xx}, g^{xy}, g^{xz}, g^{yy}, g^{yz}, g^{zz}, \sqrt{g}$ where, • **time**: Simulation time t [$L_{\rm ref}/v_{\rm ref}$] (real) • θ: Poloidal angle (or Toroidal angle φ when equib_type = "vmec") (real) • B: Magnetic field strength [$B_{\rm ref}$] (real) • $\frac{\partial B}{\partial x}$, $\frac{\partial B}{\partial y}$, $\frac{\partial B}{\partial z}$: Derivative of B [$B_{\rm ref}/L_{\rm ref}$] (real) • $g^{xx}, g^{xy}, g^{xz}[L_{\rm ref}^{-1}], g^{yy}, g^{yz}[L_{\rm ref}^{-1}], g^{zz}[L_{\rm ref}^{-2}]$: Metric tensor (real) • $\sqrt{g}$: Jacobian [$L_{\rm ref}$] (real)

Table A.20: hst/gkvp.frq.(inum in 3 digits)

Field	Description
File type	`Ascii`
Output timing	`dtout_eng` (when `calc_type == "linear"` or `"lin_freq"`)
MPI ranks	`rankg == 0`
Total files	`(Total run numbers)`
GKV unit	`ofrq`
Stored data	time, omega where, • **time**: Simulation time t [$L_{\rm ref}/v_{\rm ref}$] (real) • **omega(1:global_ny)**: k_y spectrum of complex linear frequency ω = (real frequency, growthrate) [$v_{\rm ref}/L_{\rm ref}$] (real, real) Complex frequency for $k_x = 0$ at each time is evaluated as $\omega = \omega_{\rm r} + i\gamma = \frac{\ln \tilde{\phi}_{\boldsymbol{k}}(t+\Delta t) - \ln \tilde{\phi}_{\boldsymbol{k}}(t)}{-i\Delta t}$ by assuming $\tilde{\phi}_{\boldsymbol{k}}(t) \propto e^{-i\omega t}$.

Table A.21: hst/gkvp.dsp.(inum in 3 digits)

Field	Description
File type	`Ascii`
Output timing	`End of the run` (when `calc_type == "linear"` or `"lin_freq"`)
MPI ranks	`rankg == 0`
Total files	`(Total run numbers)`
GKV unit	`odsp`
Stored data	ky, omega, diff, 1-ineq where, • **ky**: Field-line-label (poloidal) wavenumber k_y $[\rho_{\rm ref}^{-1}]$ (real) • **omega**: Complex linear frequency ω = (real frequency, growthrate) $[v_{\rm ref}/L_{\rm ref}]$ (real, real) • **diff**: Relative residual error $\frac{\omega(t)-\omega(t-\Delta t)}{\omega(t)}$ (real, real) • **1-ineq**: Convergence check based on Schwartz inequality (real) At the end of run, estimated complex frequency for $k_x = 0$ are dumped. If some modes are not yet converged, they are commented out.

Table A.22: log/gkvp.(rankg in 6 digits).(ranks in 1 digit).log.(inum in 3 digits)

Field	Description
File type	`Ascii`
Output timing	`As needed`
MPI ranks	`All`
Total files	`nprocw*nprocz*nprocv*nprocm*nprocs*(Total run numbers)`
GKV unit	`olog`
Stored data	`Simulation log`

A.4 Data-reading module diag_rb in the post-processing program diag

To read GKV binary output in the post-processing program `diag`, use the data-reading module `diag_rb`.

Listing A.1: An example to use diag_rb

```
use diag_rb, only : rb_phi_loop
complex(kind=DP) :: phi(-nx:nx, 0:global_ny, -global_nz:global_nz-1)
integer :: loop = 100
call rb_phi_loop(loop, phi) ! Read potential phi at output record loop=100 (time=dtout_
↪ptn*loop)
```

The output record number `loop` is counted up from the first run (`inum` =1) by evaluating file size of GKV binary output. As shown in Fig. A.1, output record number for the binary output `$DIR/phi/*phi*` is from `loop_phi_sta` (001) = 0 to `loop_phi_end(enum)` = `nloop_phi`. Therefore, even if you analyze only run numbers from `snum` > 1 to `enum`, all GKV binary output data from `inum` =1 should be left in the diagnosed directory.

Taking a look at the source code of `diag_rb`, one finds various types of subroutines which read electrostatic potential $\tilde{\phi}_{\boldsymbol{k}}$ in (k_x, k_y, z) or in (k_x, k_y) at a given z or in (z) for a given mode k_x, k_y, etc., and similarly read magnetic vector potential $\tilde{A}_{\parallel \boldsymbol{k}}$, fluid moments, and so on. Some typical subroutines are listed below. One may find more efficient

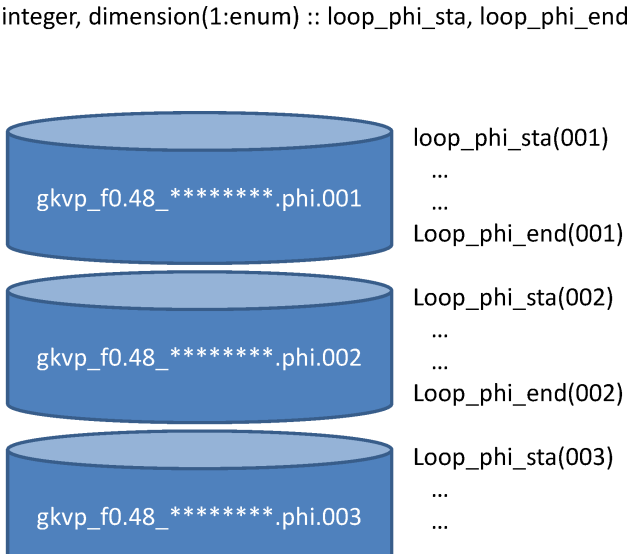

Fig. A.1: Output record number in the post-processing program `diag`

subroutine in the source code of `diag_rb`.

List of subroutines in the data-reading module `diag_rb`

Table A.23: rb_phi_gettime(loop, time)

Field	Description
Arguments	• integer, intent(in) :: loop • real(kind=DP), intent(out) :: time
GKV binary output	phi/gkvp_f0.48_(rankg in 6 digits).0.phi.(inum in 3 digits)
Description	Read simulation time $time$ corresponding to the output record $loop$. ($time \simeq dtout\_ptn \times loop$)

Table A.24: rb_Al_gettime(loop, time)

Field	Description
Arguments	• integer, intent(in) :: loop • real(kind=DP), intent(out) :: time
GKV binary output	phi/gkvp_f0.48_(rankg in 6 digits).0.Al.(inum in 3 digits)
Description	Read simulation time $time$ corresponding to the output record $loop$. ($time \simeq dtout\_ptn \times loop$)

Table A.25: rb_mom_gettime(loop, time)

Field	Description
Arguments	• integer, intent(in) :: loop • real(kind=DP), intent(out) :: time
GKV binary output	phi/gkvp_f0.48_(rankg in 6 digits).(ranks in 1 digit).mom.(inum in 3 digits)
Description	Read simulation time $time$ corresponding to the output record $loop$. ($time \simeq dtout\_ptn \times loop$)

Table A.26: rb_trn_gettime(loop, time)

Field	Description
Arguments	• integer, intent(in) :: loop • real(kind=DP), intent(out) :: time
GKV binary output	phi/gkvp_f0.48_(rankg in 6 digits).(ranks in 1 digit).trn.(inum in 3 digits)
Description	Read simulation time $time$ corresponding to the output record $loop$. ($time \simeq dtout\_eng \times loop$)

Table A.27: rb_phi_loop(loop, phi)

Field	Description
Arguments	• integer, intent(in) :: loop • complex(kind=DP), intent(out) :: phi(-nx:nx,0:global_ny,-global_nz:global_nz-1)
GKV binary output	phi/gkvp_f0.48_(rankg in 6 digits).0.phi.(inum in 3 digits)
Description	Read electrostatic potential phi corresponding to the output record $loop$. ($time \simeq dtout\_ptn \times loop$)

Table A.28: rb_Al_loop(loop, Al)

Field	Description
Arguments	• integer, intent(in) :: loop • complex(kind=DP), intent(out) :: Al(-nx:nx,0:global_ny,-global_nz:global_nz-1)
GKV binary output	phi/gkvp_f0.48_(rankg in 6 digits).0.Al.(inum in 3 digits)
Description	Read vector potential Al corresponding to the output record $loop$. ($time \simeq dtout_ptn \times loop$)

Table A.29: rb_mom_imomisloop(imom, is, loop, mom)

Field	Description
Arguments	• integer, intent(in) :: imom, is, loop • complex(kind=DP), intent(out) :: mom(-nx:nx,0:global_ny,-global_nz:global_nz-1)
GKV binary output	phi/gkvp_f0.48_(rankg in 6 digits).(ranks in 1 digit).mom.(inum in 3 digits)
Description	Read a fluid moment mom corresponding to the output record $loop$ ($time \simeq dtout\_ptn * loop$), where is specifies the plasma species, and $imom = 0 - 5$ correspond to $\tilde{n}_{s\boldsymbol{k}}$, $\tilde{u}_{\parallel s\boldsymbol{k}}$, $\tilde{p}_{\parallel s\boldsymbol{k}}$, $\tilde{p}_{\perp s\boldsymbol{k}}$, $\tilde{q}_{\parallel\parallel s\boldsymbol{k}}$, $\tilde{q}_{\parallel\perp s\boldsymbol{k}}$.

Table A.30: rb_trn_itrnisloop(itrn, is, loop, trn)

Field	Description
Arguments	• integer, intent(in) :: itrn, is, loop • real(kind=DP), intent(out) :: trn(-nx:nx,0:global_ny)
GKV binary output	phi/gkvp_f0.48_(rankg in 6 digits).(ranks in 1 digit).trn.(inum in 3 digits)
Description	Read a variable corresponding to the entropy balance trn at the output record $loop$ ($time \simeq dtout\_eng * loop$), where is specifies the plasma species, and $itrn = 0 - 11$ correspond to perturbed gyrocenter entropy, electrostatic field energy including polarization, magnetic field energy, wave-particle interaction via electrostatic fluctuations, wave-particle interaction via magnetic fluctuations, nonlinear entropy transfer via $\boldsymbol{E} \times \boldsymbol{B}$ flows, nonlinear entropy transfer via magnetic flutters, collisional dissipation, particle flux by $\boldsymbol{E} \times \boldsymbol{B}$ flows, particle flux by magnetic flutters, energy flux by $\boldsymbol{E} \times \boldsymbol{B}$ flows, energy flux by magnetic flutters.

A.5 Diagnostics modules in the post-processing program diag

Some diagnostics modules are explained below.

List of subroutines in diagnostics modules

Table A.31: phiinxy(giz, loop)

Field	Description
Contained in	`out_mominxy` module
Arguments	• integer, intent(in) :: giz, loop
Output	post/data/phiinxy_z(giz in 4 digits)_t(loop in 8 digits).dat
Description	Write 2D electrostatic potential $\tilde{\phi}(x, y)$ for $z = z(giz)$ at output record $loop$ ($time \simeq dtout\_ptn * loop$).

Table A.32: Alinxy(giz, loop)

Field	Description
Contained in	`out_mominxy` module
Arguments	• integer, intent(in) :: giz, loop
Output	post/data/Alinxy_z(giz in 4 digits)_t(loop in 8 digits).dat
Description	Write 2D vector potential $\tilde{A}_{\parallel}(x,y)$ for $z = z(giz)$ at output record $loop$ ($time \simeq dtout\_ptn * loop$).

Table A.33: mominxy(giz, is, loop)

Field	Description
Contained in	`out_mominxy` module
Arguments	• integer, intent(in) :: giz, is, loop
Output	post/data/mominxy_z(giz in 4 digits)s(is in 1 digit)_t(loop in 8 digits).dat
Description	Write 2D fluid moments $\tilde{n}_{\rm s}(x,y)$, $\tilde{u}_{\parallel{\rm s}}(x,y)$, $\tilde{p}_{\parallel{\rm s}}(x,y)$, $\tilde{p}_{\perp{\rm s}}(x,y)$, $\tilde{q}_{\parallel\parallel{\rm s}}(x,y)$, $\tilde{q}_{\parallel\perp{\rm s}}(x,y)$ of the plasma species is for $z = z(giz)$ at output record $loop$ ($time \simeq dtout\_ptn * loop$).

Table A.34: phiinz(mx, gmy, loop)

Field	Description
Contained in	`out_mominz` module
Arguments	• integer, intent(in) :: mx, gmy, loop
Output	post/data/phiinz_mx(mx in 4 digits)my(gmy in 4 digits)_t(loop in 8 digits).dat
Description	Write electrostatic potential along a field line $\tilde{\phi}_{\boldsymbol{k}}(z)$ for the given mode $(kx(mx), ky(gmy))$ at output record $loop$ ($time \simeq dtout\_ptn * loop$).

Table A.35: Alinz(mx, gmy, loop)

Field	Description
Contained in	`out_mominz` module
Arguments	• integer, intent(in) :: mx, gmy, loop
Output	post/data/Alinz_mx(mx in 4 digits)my(gmy in 4 digits)_t(loop in 8 digits).dat
Description	Write vector potential along a field line $\tilde{A}_{\parallel\boldsymbol{k}}(z)$ for the given mode $(kx(mx), ky(gmy))$ at output record $loop$ ($time \simeq dtout\_ptn * loop$).

Table A.36: mominz(mx, gmy, is, loop)

Field	Description
Contained in	`out_mominz` module
Arguments	• integer, intent(in) :: mx, gmy, is, loop
Output	post/data/mominz_mx(mx in 4 digits)my(gmy in 4 digits)s(is in 1 digit)_t(loop in 8 digits).dat
Description	Write fluid moments along a field line $\tilde{n}_{s\boldsymbol{k}}(z)$, $\tilde{u}_{\parallel s\boldsymbol{k}}(z)$, $\tilde{p}_{\parallel s\boldsymbol{k}}(z)$, $\tilde{p}_{\perp s\boldsymbol{k}}(z)$, $\tilde{q}_{\parallel\parallel s\boldsymbol{k}}(z)$, $\tilde{q}_{\parallel\perp s\boldsymbol{k}}(z)$ of the plasma species is for the given mode $(kx(mx), ky(gmy))$ at output record $loop$ ($time \simeq dtout\_ptn * loop$).

Table A.37: phiinz_connect(mx, gmy, loop)

Field	Description
Contained in	`out_mominz` module
Arguments	• integer, intent(in) :: mx, gmy, loop
Output	post/data/phiinz_connect_mx(mx in 4 digits)my(gmy in 4 digits)_t(loop in 8 digits).dat
Description	Write electrostatic potential along a field line $\tilde{\phi}_{\boldsymbol{k}}(z)$ for the given mode $(kx(mx), ky(gmy))$ at output record $loop$ ($time \simeq dtout\_ptn * loop$). With considering the pseudo-periodic boundary condition in the fluxtube model, the mode structure is extended in the field-aligned coordinate by connecting $k_x \pm \delta k_x$ modes.

Table A.38: phiinkxky(loop)

Field	Description
Contained in	`out_mominkxky` module
Arguments	• integer, intent(in) :: loop
Output	post/data/phiinkxky_t(loop in 8 digits).dat
Description	Write (k_x, k_y) spectrum of electrostatic potential $\langle\|\tilde{\phi}_{\boldsymbol{k}}|^2\rangle/2$ at output record $loop$ ($time \simeq dtout\_ptn * loop$).

Table A.39: Alinkxky(loop)

Field	Description
Contained in	`out_mominkxky` module
Arguments	• integer, intent(in) :: loop
Output	post/data/Alinkxky_t(loop in 8 digits).dat
Description	Write (k_x, k_y) spectrum of vector potential $\langle\|\tilde{A}_{\parallel\boldsymbol{k}}|^2\rangle/2$ at output record $loop$ ($time \simeq dtout\_ptn * loop$).

Table A.40: mominkxky(is, loop)

Field	Description
Contained in	`out_mominkxky` module
Arguments	• integer, intent(in) :: is, loop
Output	post/data/mominkxky_s(is in 1 digit)_t(loop in 8 digits).dat
Description	Write (k_x, k_y) spectra of fluid moments $\langle\|\tilde{n}_{s\boldsymbol{k}}\|^2\rangle/2$, $\langle\|\tilde{u}_{\parallel s\boldsymbol{k}}\|^2\rangle/2$, $\langle\|\tilde{p}_{\parallel s\boldsymbol{k}}\|^2\rangle/2$, $\langle\|\tilde{p}_{\perp s\boldsymbol{k}}\|^2\rangle/2$, $\langle\|\tilde{q}_{\parallel\parallel s\boldsymbol{k}}\|^2\rangle/2$, $\langle\|\tilde{q}_{\parallel\perp s\boldsymbol{k}}\|^2\rangle/2$ of the plasma species is at output record $loop$ ($time \simeq dtout\_ptn * loop$).

Table A.41: trninkxky(is, loop)

Field	Description
Contained in	`out_trninkxky` module
Arguments	• integer, intent(in) :: is, loop
Output	post/data/trninkxky_s(is in 1 digit)_t(loop in 8 digits).dat
Description	Write (k_x, k_y) spectra of variables in entropy balance relation of the plasma species is at output record $loop$ ($time \simeq dtout\_eng * loop$).

Table A.42: triinkxky(mxt, myt, is, loop)

Field	Description
Contained in	`out_triinkxky` module
Arguments	• integer, intent(in) :: mxt, myt, is, loop
Output	post/data/trninkxky_s(is in 1 digit)_t(loop in 8 digits).dat
Description	Write (p_x, p_y) spectra of triad transfer functions $J^{\boldsymbol{p},\boldsymbol{q}}_{\mathrm{sE}\boldsymbol{k}}$, $J^{\boldsymbol{q},\boldsymbol{k}}_{\mathrm{sE}\boldsymbol{p}}$, $J^{\boldsymbol{k},\boldsymbol{p}}_{\mathrm{sE}\boldsymbol{q}}$, $J^{\boldsymbol{p},\boldsymbol{q}}_{\mathrm{sM}\boldsymbol{k}}$, $J^{\boldsymbol{q},\boldsymbol{k}}_{\mathrm{sM}\boldsymbol{p}}$, $J^{\boldsymbol{k},\boldsymbol{p}}_{\mathrm{sM}\boldsymbol{q}}$ of the plasma species is for the mode $(k_x(mxt), k_y(myt))$ at output record $loop$ ($time \simeq dtout\_ptn * loop$).

A.6 Adiabatic electron/ion model for nprocs=1

When one runs a single-species simulation with setting `nprocs` =1, GKV employs adiabatic model for electrons or ions. In both case, electrostatic limit is assumed ($\tilde{A}_\parallel = 0$), and `lambda_i` and `beta` in `gkvp_namelist` are neglected. Setting of kinetic electrons with adiabatic ion model is `nprocs` =1 in `src/gkvp_header.f90`, and `Anum` =1.d0, `Znum` =1.d0, `fcs` =1.d0, `sgn` =-1.d0 in `run/gkvp_namelist`. Then the Poisson eq. with adiabatic ion model is

$$\left[\frac{e^2 n_0}{T_\mathrm{i}} + \frac{e^2 n_0}{T_\mathrm{e}}\left(1 - \Gamma_{0\mathrm{e}\boldsymbol{k}}\right)\right]\tilde{\phi}_{\boldsymbol{k}} = -e\int dv^3 J_{0\mathrm{e}\boldsymbol{k}}\tilde{f}_{\mathrm{e}\boldsymbol{k}} \tag{A.1}$$

Density, temperature and mass are normalized electrons' value. Then the normalized Poisson eq. is

$$\left[\frac{T_\mathrm{e}}{T_\mathrm{i}} + 1 - \bar{\Gamma}_{0\mathrm{e}\boldsymbol{k}}\right]\bar{\phi}_{\boldsymbol{k}} = -\int d\bar{v}^3 \bar{J}_{0\mathrm{e}\boldsymbol{k}}\bar{f}_{\mathrm{e}\boldsymbol{k}}. \tag{A.2}$$

The temperature ratio $T_\mathrm{e}/T_\mathrm{i}$ is given by `tau_ad` in `run/gkvp_namelist`. Setting of kinetic ions with adiabatic electron model is `nprocs` =1 in `src/gkvp_header.f90`, and `Anum` =1.d0, `Znum` =1.d0, `fcs` =1.d0, `sgn` =1.d0 in `run/`

gkvp_namelist. Then the Poisson eq. with adiabatic electron model is

$$\frac{e^2 n_0}{T_\mathrm{i}} \left(1 - \Gamma_{0\mathrm{i}\boldsymbol{k}}\right) \tilde{\phi}_{\boldsymbol{k}} = -\frac{e^2 n_0}{T_\mathrm{e}} \left(\tilde{\phi}_{\boldsymbol{k}} - \langle \tilde{\phi}_{\boldsymbol{k}} \rangle \delta_{k_y,0}\right) + e \int dv^3 J_{0\mathrm{i}\boldsymbol{k}} \tilde{f}_{\mathrm{i}\boldsymbol{k}}, \tag{A.3}$$

where $\langle \cdots \rangle$ denotes the flux-surface average, and $\delta_{i,j}$ is the Kronecker's delta. Density, temperature and mass are normalized ions' value. Then the normalized Poisson eq. is

$$\left(1 - \bar{\Gamma}_{0\mathrm{i}\boldsymbol{k}}\right) \bar{\phi}_{\boldsymbol{k}} + \frac{T_\mathrm{i}}{T_\mathrm{e}} \left(\bar{\phi}_{\boldsymbol{k}} - \langle \bar{\phi}_{\boldsymbol{k}} \rangle \delta_{k_y,0}\right) = \int dv^3 \bar{J}_{0\mathrm{i}\boldsymbol{k}} \bar{f}_{\mathrm{i}\boldsymbol{k}}, \tag{A.4}$$

The temperature ratio $T_\mathrm{i}/T_\mathrm{e}$ is given by tau_ad in run/gkvp_namelist.

APPENDIX

B

SUPPLEMENTAL

B.1 Entropy balance equation for each wavenumber and plasma species

$$
\begin{aligned}
\frac{dS_{\mathrm{s}\boldsymbol{k}}}{dt} &= \frac{T_{\mathrm{s}}\Gamma_{\mathrm{s}\boldsymbol{k}}}{L_{p\mathrm{s}}} + \frac{\Theta_{\mathrm{s}\boldsymbol{k}}}{L_{T\mathrm{s}}} + I_{\mathrm{s}\boldsymbol{k}} + R_{\mathrm{s}\boldsymbol{k}} + D_{\mathrm{s}\boldsymbol{k}} + E_{\mathrm{s}\boldsymbol{k}}, \\
\frac{dW_{\mathrm{E}\boldsymbol{k}}}{dt} &= -\sum_{\mathrm{s}} R_{\mathrm{sE}\boldsymbol{k}}, \\
\frac{dW_{\mathrm{M}\boldsymbol{k}}}{dt} &= -\sum_{\mathrm{s}} R_{\mathrm{sM}\boldsymbol{k}},
\end{aligned}
\tag{B.1}
$$

where

$$
\begin{aligned}
S_{\mathrm{s}\boldsymbol{k}} &= \left\langle \int dv^3 \frac{T_{\mathrm{s}}|\tilde{f}_{\mathrm{s}\boldsymbol{k}}|^2}{2F_{\mathrm{sM}}} \right\rangle, \\
W_{\mathrm{E}\boldsymbol{k}} &= \left\langle \left[\varepsilon_0 k_\perp^2 + \sum_{\mathrm{s}} \frac{e_{\mathrm{s}}^2 n_{\mathrm{s}}}{T_{\mathrm{s}}} \left(1 - \Gamma_{0\mathrm{s}\boldsymbol{k}}\right) \right] \frac{|\tilde{\phi}_{\boldsymbol{k}}|^2}{2} \right\rangle, \\
W_{\mathrm{M}\boldsymbol{k}} &= \left\langle \frac{k_\perp^2}{\mu_0} \frac{|\tilde{A}_{\parallel\boldsymbol{k}}|^2}{2} \right\rangle, \\
\Gamma_{\mathrm{s}\boldsymbol{k}} &= \Gamma_{\mathrm{sE}\boldsymbol{k}} + \Gamma_{\mathrm{sM}\boldsymbol{k}} = \mathrm{Re}\left[\left\langle -\frac{ik_y\tilde{\phi}_{\boldsymbol{k}}}{c_b}\tilde{n}_{\mathrm{s}\boldsymbol{k}}^* + \frac{ik_y\tilde{A}_{\parallel\boldsymbol{k}}}{c_b}\tilde{u}_{\parallel\mathrm{s}\boldsymbol{k}}^* \right\rangle \right], \\
Q_{\mathrm{s}\boldsymbol{k}} &= Q_{\mathrm{sE}\boldsymbol{k}} + Q_{\mathrm{sM}\boldsymbol{k}} = \mathrm{Re}\left[\left\langle -\frac{ik_y\tilde{\phi}_{\boldsymbol{k}}}{c_b}\tilde{p}_{\mathrm{s}\boldsymbol{k}}^* + \frac{ik_y\tilde{A}_{\parallel\boldsymbol{k}}}{c_b}\tilde{q}_{\parallel\mathrm{s}\boldsymbol{k}}^* \right\rangle \right], \\
\Theta_{\mathrm{s}\boldsymbol{k}} &= Q_{\mathrm{s}\boldsymbol{k}} - \frac{5}{2}T_{\mathrm{s}}\Gamma_{\mathrm{s}\boldsymbol{k}}, \\
I_{\mathrm{s}\boldsymbol{k}} &= \sum_{\boldsymbol{p}}\sum_{\boldsymbol{q}} J_{\mathrm{s}\boldsymbol{k}}^{\boldsymbol{p},\boldsymbol{q}}, \\
R_{\mathrm{s}\boldsymbol{k}} &= R_{\mathrm{sE}\boldsymbol{k}} + R_{\mathrm{sM}\boldsymbol{k}} = \mathrm{Re}\left[\left\langle -\tilde{\phi}_{\boldsymbol{k}}^* \frac{\partial e_{\mathrm{s}}\tilde{n}_{\mathrm{s}\boldsymbol{k}}}{\partial t} - e_{\mathrm{s}}\tilde{u}_{\mathrm{s}\boldsymbol{k}}^* \frac{\partial \tilde{A}_{\parallel\boldsymbol{k}}}{\partial t} \right\rangle \right], \\
D_{\mathrm{s}\boldsymbol{k}} &= \mathrm{Re}\left[\left\langle \int dv^2 \frac{T_{\mathrm{s}}\tilde{g}_{\mathrm{s}\boldsymbol{k}}^*}{F_{\mathrm{sM}}} C_{\mathrm{s}\boldsymbol{k}} \right\rangle \right], \\
E_{\mathrm{s}\boldsymbol{k}} &= \mathrm{Re}\left[-\left\langle \int dv^3 v_\parallel \nabla_\parallel \frac{T_{\mathrm{s}}|\tilde{g}_{\mathrm{s}\boldsymbol{k}}|^2}{2F_{\mathrm{sM}}} \right\rangle \right],
\end{aligned}
\tag{B.2}
$$

with

$$
\begin{aligned}
\tilde{g}_{\mathrm{s}\boldsymbol{k}} &= \tilde{f}_{\mathrm{s}\boldsymbol{k}} + \frac{e_{\mathrm{s}} J_{0\mathrm{s}\boldsymbol{k}} \tilde{\phi}_{\boldsymbol{k}}}{T_{\mathrm{s}}} F_{\mathrm{sM}}, \\
\tilde{n}_{\mathrm{s}\boldsymbol{k}} &= \int dv^3 J_{0\mathrm{s}\boldsymbol{k}} \tilde{f}_{\mathrm{s}\boldsymbol{k}}, \\
\tilde{u}_{\parallel\mathrm{s}\boldsymbol{k}} &= \int dv^3 v_{\parallel} J_{0\mathrm{s}\boldsymbol{k}} \tilde{f}_{\mathrm{s}\boldsymbol{k}}, \\
\tilde{p}_{\parallel\mathrm{s}\boldsymbol{k}} &= \int dv^3 \frac{m_{\mathrm{s}} v_{\parallel}^2}{2} J_{0\mathrm{s}\boldsymbol{k}} \tilde{f}_{\mathrm{s}\boldsymbol{k}}, \\
\tilde{p}_{\perp\mathrm{s}\boldsymbol{k}} &= \int dv^3 \mu B J_{0\mathrm{s}\boldsymbol{k}} \tilde{f}_{\mathrm{s}\boldsymbol{k}}, \\
\tilde{q}_{\parallel\parallel\mathrm{s}\boldsymbol{k}} &= \int dv^3 v_{\parallel} \frac{m_{\mathrm{s}} v_{\parallel}^2}{2} J_{0\mathrm{s}\boldsymbol{k}} \tilde{f}_{\mathrm{s}\boldsymbol{k}}, \\
\tilde{q}_{\parallel\perp\mathrm{s}\boldsymbol{k}} &= \int dv^3 v_{\parallel} \mu B J_{0\mathrm{s}\boldsymbol{k}} \tilde{f}_{\mathrm{s}\boldsymbol{k}}, \\
\tilde{p}_{\mathrm{s}\boldsymbol{k}} &= \tilde{p}_{\parallel\mathrm{s}\boldsymbol{k}} + \tilde{p}_{\perp\mathrm{s}\boldsymbol{k}}, \\
\tilde{q}_{\parallel\mathrm{s}\boldsymbol{k}} &= \tilde{q}_{\parallel\parallel\mathrm{s}\boldsymbol{k}} + \tilde{q}_{\parallel\perp\mathrm{s}\boldsymbol{k}}.
\end{aligned}
\tag{B.3}
$$

See also Refs. [B-1] and [B-2].

B.2 Triad transfer function

$$
J_{\mathrm{s}\boldsymbol{k}}^{\boldsymbol{p},\boldsymbol{q}} = \delta_{\boldsymbol{k}+\boldsymbol{p}+\boldsymbol{q},\boldsymbol{0}} \frac{\boldsymbol{b}\cdot\boldsymbol{p}\times\boldsymbol{q}}{2c_b} \mathrm{Re}\left[\left\langle \int dv^3 (\chi_{\mathrm{s}\boldsymbol{p}} \tilde{g}_{\mathrm{s}\boldsymbol{q}} - \chi_{\mathrm{s}\boldsymbol{q}} \tilde{g}_{\mathrm{s}\boldsymbol{p}}) \frac{T_{\mathrm{s}} \tilde{g}_{\mathrm{s}\boldsymbol{k}}}{F_{\mathrm{sM}}} \right\rangle \right], \tag{B.4}
$$

where $\tilde{g}_{\mathrm{s}\boldsymbol{k}} = \tilde{f}_{\mathrm{s}\boldsymbol{k}} + e_{\mathrm{s}} J_{0\mathrm{s}\boldsymbol{k}} \tilde{\phi}_{\boldsymbol{k}} F_{\mathrm{sM}}/T_{\mathrm{s}}$ and $\chi_{\mathrm{s}\boldsymbol{k}} = J_{0\mathrm{s}\boldsymbol{k}}(\tilde{\phi}_{\boldsymbol{k}} - v_{\parallel}\tilde{A}_{\parallel\boldsymbol{k}})$. The triad transfer function satisfy the following properties [B-3]:

$$
\begin{aligned}
J_{\mathrm{s}\boldsymbol{k}}^{\boldsymbol{p},\boldsymbol{q}} &= J_{\mathrm{s}\boldsymbol{k}}^{\boldsymbol{q},\boldsymbol{p}}, \\
J_{\mathrm{s}\boldsymbol{k}}^{\boldsymbol{p},\boldsymbol{q}} + J_{\mathrm{s}\boldsymbol{p}}^{\boldsymbol{q},\boldsymbol{k}} + J_{\mathrm{s}\boldsymbol{q}}^{\boldsymbol{k},\boldsymbol{p}} &= 0.
\end{aligned}
\tag{B.5}
$$

Note that $J_{\mathrm{s}\boldsymbol{k}}^{\boldsymbol{p},\boldsymbol{q}}$ is symmetrized so as to eliminate asymmetric components, which cancel out in the net entropy transfer $I_{\mathrm{s}\boldsymbol{k}}$ and thus do not contribute to physics. Since the terms of $\tilde{\phi}_{\boldsymbol{k}}$ and of $\tilde{A}_{\parallel\boldsymbol{k}}$ respectively correspond to $\boldsymbol{E}\times\boldsymbol{B}$ and magnetic flutter nonlinearities, these contributions can be evaluated separately,

$$
\begin{aligned}
I_{\mathrm{s}\boldsymbol{k}} &= I_{\mathrm{sE}\boldsymbol{k}} + I_{\mathrm{sM}\boldsymbol{k}} = \sum_{\boldsymbol{p}} \sum_{\boldsymbol{q}} J_{\mathrm{sE}\boldsymbol{k}}^{\boldsymbol{p},\boldsymbol{q}} + \sum_{\boldsymbol{p}} \sum_{\boldsymbol{q}} J_{\mathrm{sM}\boldsymbol{k}}^{\boldsymbol{p},\boldsymbol{q}}, \\
J_{\mathrm{s}\boldsymbol{k}}^{\boldsymbol{p},\boldsymbol{q}} &= J_{\mathrm{sE}\boldsymbol{k}}^{\boldsymbol{p},\boldsymbol{q}} + J_{\mathrm{sM}\boldsymbol{k}}^{\boldsymbol{p},\boldsymbol{q}}.
\end{aligned}
\tag{B.6}
$$

B.3 Integrals in GKV

Flux-surface average:

$$
\begin{aligned}
\left\langle \tilde{\phi}(x,y,z) \right\rangle &= \sum_{k_x} \left\langle \tilde{\phi}_{k_x,k_y=0}(z) \right\rangle e^{ik_x x}, \\
\left\langle \tilde{\phi}_{k_x,k_y=0}(z) \right\rangle &= \frac{\int_{\pi}^{\pi} dz \sqrt{g} \tilde{\phi}_{k_x,k_y=0}(z)}{\int_{\pi}^{\pi} dz \sqrt{g}}.
\end{aligned}
\tag{B.7}
$$

Volume average:

$$
\int dx^3 \left| \tilde{\phi}(x,y,z) \right|^2 = \sum_{k_x} \sum_{k_y} \left\langle \left| \tilde{\phi}_{\boldsymbol{k}}(z) \right|^2 \right\rangle. \tag{B.8}
$$

Velocity-space integral:

$$\int dv^3 \tilde{f}_{\mathrm{s}\boldsymbol{k}}(z, v_\parallel, mu) = \int_{-L_v}^{L_v} dv_\parallel \int_0^{L_v} dv_\perp 2\pi v_\perp \tilde{f}_{\mathrm{s}\boldsymbol{k}}(z, v_\parallel, \mu). \tag{B.9}$$

Table of Contents

Setting up a Python virtual environment

A Python virtual environment is set up by following the steps below.

Initial setup 1. Construction of a Python virtual environment

```
$ mkdir ~/myvenv/                    # All venv virtual environments are stored under ~/
↪myvenv/.
$ python3.12 -m venv ~/myvenv/env01  # A virtual environment named env01 is created␣
↪based on the preinstalled Python 3.12.
$ source ~/myvenv/env01/bin/activate # The env01 environment is activated.
```

Initial setup 2. Installation of required Python packages

```
$ python -m pip install --upgrade pip        # First, upgrade the package manager pip␣
↪itself
$ pip install jupyterlab xarray dask zarr f90nml numba pyevtk pyvista pydmd # for diag_
↪python
$ pip install bzx                            # for BZX
$ pip install gkvfig                         # for gkvfig
```

Downloading the GKV code

The GKV code can be downloaded with the following command:

```
$ wget https://github.com/GKV-developers/gkvp/archive/refs/tags/f0.65.tar.gz -O gkvp-f0.
↪65.tar.gz
```

The code can also be obtained in other ways.

Alternative method 1. The file gkvp-f0.65.tar.gz is obtained by accessing https://github.com/GKV-developers/gkvp/, selecting gkvp_f0.65 from the Releases button, downloading the archive, and then transferring it via SSH or a similar

method.

Alternative method 2. For users familiar with Git, the repository may be obtained by running git clone after appropriate configuration.

After downloading, the source code is unpacked as follows:

```
$ tar xzvf gkvp-f0.65.tar.gz
```

If the archive is unpacked correctly, a directory `gkvp-f0.65/` is created.

When working on Plasma Simulator subsystem A, the Intel module must be loaded in order to compile the program:

```
$ module load intel/2025.1
```

Code structure and overview of GKV

The code structure of GKV (version gkvp_f0.65) is as follows:

```
gkvp-f0.65/
  ├── README_for_namelist.txt  Brief notes on the namelist
  ├── Version_memo.txt         Recent update history
  ├── src/                     Source files
  │   ├── gkvp_header.f90      Module for resolution and MPI settings
  │   └── gkvp_out.f90         Module for standard data output
  ├── lib/                     Modules for random number and Bessel-function libraries
  ├── run/                     Compilation and execution of simulations
  │   ├── gkvp_namelist        Physical parameter settings
  │   ├── sub.q                Batch job script (machine dependent)
  │   ├── shoot                Job submission script (machine dependent)
  │   ├── Makefile             Compilation settings (machine dependent)
  │   └── backup/              Backup of sub.q, shoot, and Makefile for other machines
  ├── extra_tools/             Legacy post-processing tools and others
  └── benchmarks/              Benchmark datasets
```

With GKV, local analysis of microinstabilities, turbulent fluctuations, and particle and heat transport can be carried out under a given equilibrium.

1. Linear analysis
   - Growth rates, real frequencies, and cross-phases among fluctuating quantities are examined.
   - Owing to the independence of linear modes, only specific wavenumbers are analyzed, so the computation is relatively fast.
   - Because nonlinear saturation mechanisms are not included, absolute fluctuation amplitudes are not obtained.
2. Nonlinear analysis
   - Turbulent fluctuation analysis is performed, including fluctuation spectra, particle and heat transport, and various related quantities.
   - Since turbulent mixing involves nonlinear coupling among many modes, the computation is time-consuming. The required resolution depends on the problem, so checks such as entropy balance and convergence of spectra are needed to ensure numerical soundness.

In this tutorial, linear analysis of microinstabilities is carried out under a circular tokamak model magnetic configuration, following the steps below.

1. Physical parameters are specified in `run/gkvp_namelist`.
2. The numbers of grid points and MPI processes are set in `src/gkvp_header.f90`.
3. The batch job script `sub.q` is configured.
4. The directory settings are adjusted in the job submission script `shoot`.
5. The program is compiled and the simulation is executed.
6. The output data are analyzed (using post-processing tools).

Nonlinear analysis follows essentially the same procedure, with the calculation type set to `"nonlinear"` and with higher resolution adopted as needed.

Editing various files

The procedure for converting experimental measurements into GKV parameters and entering them into the namelist is outlined below.

From experimental measurements, the following steps are taken:

- Choice of radial position for analysis and evaluation of local plasma parameters
- Construction of MHD equilibrium

Next, in order to carry out analysis with GKV, the following steps are performed:

- Conversion (normalization) from experimental values to GKV parameters
- Input of corresponding physical parameters into `run/gkvp_namelist`
- Processing of MHD equilibrium into metric data required by GKV

In the present tutorial, MHD equilibrium magnetic configuration data are not used. Instead, a circular tokamak magnetic field model, known as the s–alpha model, is adopted. The required parameters are therefore limited to quantities such as inverse aspect ratio, safety factor, magnetic shear, density gradient, and temperature gradient. (The conversion from experimental data and the reading of MHD equilibrium magnetic configuration data are not treated here.)

The input to the namelist is then adjusted. Entries in `run/gkvp_namelist` related to the execution of the simulation are edited as appropriate (for example, setting `calc_type = "nonlinear"` for nonlinear simulations, or extending the runtime by specifying `e_limit = 3600.d0`).

Listing 1: run/gkvp_namelist

```
&cmemo memo="GKV-plus f0.65 developed for pre-exa-scale computing", &end
&calct calc_type="lin_freq", # Calculation type: lin_freq / nonlinear
z_bound="outflow",
z_filt="off",
z_calc="cf4",
art_diff=0.1d0,
init_random=.true.,
num_triad_diag=0,
vp_coord=0, &end # Velocity-space coordinate system: 0 = vl-mu, 1 = vl-vp
&triad mxt = 0, myt = 0/
&equib equib_type = "analytic", &end # Equilibrium magnetic field model:
...                                  #     analytic / s-alpha / s-alpha-shift / circ-
↪MHD /
...                                  #     vmec / eqdsk / slab / ring
&runlm e_limit = 60.d0, &end # Wall-clock time limit for the simulation [s]
```

(continues on next page)

(continued from previous page)

```
&times tend = 200.d0, # Upper limit of simulation time [R_{ref}/v_{ref} ]
dtout_fxv = 10.d0, # Time interval for data output 1
dtout_ptn = 0.1d0, # Time interval for data output 2
dtout_eng = 0.1d0, # Time interval for data output 3
dtout_dtc = 0.1d0, &end # Interval for adjusting the adaptive time step
...
&physp R0_Ln = 2.22d0, # Normalized density gradient R_0/L_{n_s}
R0_Lt = 6.92d0, # Normalized temperature gradient R_0/L_{T_s}
nu = 1.d0,
Anum = 1.d0, # Mass number m_s/m_{ref}
Znum = 1.d0, # Absolute charge number |e_s|/e
fcs = 1.d0, # Charge density s_s n_s/(e n_{ref})
sgn = 1.d0, # Sign of charge e_s/|e_s|
tau = 1.d0, # Temperature T_s/T_{ref}
dns1 = 1.d-2,
tau_ad = 1.d0,
lambda_i = 0.d0,
beta = 0.d0, # Plasma beta mu_0 n_{ref} T_{ref}/B_{ref}^2
2
...
&nperi n_tht = 3, # Box size along the field line (poloidal angle ±n_tht*pi)
kymin = 0.05d0, # Box size in field-line label direction ly = pi/kymin
m_j = 1, # Radial box size lx = pi/kxmin
del_c = 0.d0, &end kxmin = |2*pi*s_hat*kymin/m_j|
&confp eps_r = 0.18d0, # Inverse aspect ratio r_{at-fluxtube-center}/L_{ref}
eps_rnew = 1.d0,
q_0 = 1.4d0, # Safety factor q
s_hat = 0.8d0, # Magnetic shear s
```

The numbers of grid points and MPI processes are specified in `src/gkvp_header.f90`. The total number of grid points (`global_ny`) and the numbers of MPI processes (`nprocw`, `nprocz`, `nprocv`, `nprocm`, `nporcs`) are adjusted as appropriate.

Listing 2: src/gkvp_header.f90

```
!-------------------------------------
! Dimension size (grid numbers)
!-------------------------------------
! Global simulation domain
! in x, y,z,v,m (0:2*nxw-1, 0:2*nyw-1,-global_nz:global_nz-1,1:2*global_nv,0:global_nm)
! in kx,ky,z,v,m ( -nx:nx,0:global_ny,-global_nz:global_nz-1,1:2*global_nv,0:global_nm)
integer, parameter :: nxw = 2, nyw = 20
integer, parameter :: nx = 0, global_ny = 12 ! 2/3 de-aliasing rule
integer, parameter :: global_nz = 48, global_nv = 24, global_nm = 15
integer, parameter :: nzb = 2, & ! the number of ghost grids in z
nvb = 2 ! the number of ghost grids in v and m
!-------------------------------------
! Data distribution for MPI
!-------------------------------------
integer, parameter :: nprocw = 1, nprocz = 4, nprocv = 2, nprocm = 2, nprocs = 1
```

Here,

- `nx`

Number of k_x modes (index range `-nx:nx`)

- `global_ny`

  Number of k_y modes (index range `0:global_ny`).

  The parameters `nxw` and `nyw` are chosen so that

  `nxw` > `nx` * $\frac{3}{2}$ and `nyw` > `global_ny` * $\frac{3}{2}$.

- `global_nz`

  Coordinate along the magnetic field line, where $-n_{\text{tht}}\pi < zz < n_{\text{tht}}\pi$

  is discretized with index range `-global_nz:global_nz-1`.

- `global_nv`

  Parallel velocity along the field line, where $-v_{\max} < vl < v_{\max}$

  is discretized with index range `1:2*global_nv`.

- `global_nm`

  Magnetic moment, where $0 < \mu < v_{\max}^2/2$

  is discretized with index range `0:global_nm` (for `vp_coord = 0`),

  or perpendicular velocity, where $0 < vp < v_{\max}$

  is discretized with index range `0:global_nm` (for `vp_coord = 1`).

The parameters `nprocw`, `nprocz`, `nprocv`, `nprocm`, and `nprocs` represent the numbers of MPI domain decompositions in the k_y, zz, vl, μ directions and in the particle-species direction s, respectively.

The following conditions are imposed:

- `(global_ny + 1) / nprocw` is preferably an integer.
- `global_nz / nprocz`, `global_nv / nprocv`, `global_nm + 1 / nprocm` must be integers.
- `nprocs` matches the number of particle species treated.

When working on Plasma Simulator subsystem A, the batch job script `sub.q` and the job submission script `shoot` are configured.

In `run/sub.q`, the MPI/OpenMP parallel configuration is specified.

- The number of MPI processes is set so that it is consistent with `src/gkvp_header.f90`, e.g. `select*mpiprocs = nprocw * nprocz * nprocv * nprocm * nprocs`.

If the name of the project to which the user belongs is not known, the group directory name corresponding to the project can be checked with the `check_quota` command.

Listing 3: run/sub.q

```
...
#PBS -P NIFS23KIST041            # Project name
#PBS -q A_S                      # Queue class A_dev / A_S / A_M / A_L
#PBS -l walltime=00:15:00        # Wall-clock time limit (hh:mm:ss)
#PBS -l select=1:ncpus=128:mem=384gb:mpiprocs=16
                # Resource request. See Section 8.1.2 of the Plasma Simulator User Guide.
export OMP_NUM_THREADS=8         # Number of OpenMP threads per MPI process
...
```

Then, the directory settings are configured in the job submission script `shoot`. `DIR` may be replaced with any desired path under each user's directory. Example: `DIR=/data/(user ID)/gkv_training/linear_test/`

Listing 4: run/shoot

```
...
if [ $# -lt 2 ]; then
  echo "HOW TO USE: ./shoot [START_NUMBER] [END_NUMBER] ([JOB-ID])"
  exit
fi

#### Environment setting
DIR=/data/(User ID)/gkvp/f0.65/ITGae-lin    # Data output directory after execution
LDM=gkvp.exe
NL=gkvp_namelist
SC=qsub
JS=sub.q
### For VMEC, set VMCDIR including metric_boozer.bin.dat
#VMCDIR=./input_vmec/vmec_sample_nss501ntheta1024nzeta0
### For IGS, set IGSDIR including METRIC_{axi,boz,ham}.OUT
#IGSDIR=../../input_eqdsk_for_eqdskbench/
...
```

Compilation and execution

After editing the various files, the code is compiled.

```
$ cd gkvp-f0.65/run/
$ make clean          # Optional
$ make                # Ensure that 'module load intel' has been executed
```

The simulation is executed using the `shoot` script in the following format, which submits step jobs.

```
$ ./shoot START_NUM END_NUM (JOB_ID)
```

Example usage:

- Single job submission (`*.001`): `./shoot 1 1`
- Single job submission (`*.002`): `./shoot 2 2`
- Step job submission (`*.003` - `*.005`): `./shoot 3 5`
- Submission of continuation step jobs: `./shoot 6 7 11223` (In this example, a job with `JOB_ID = 11223` that runs up to `*.005` is already in the queue, and jobs for `*.006` - `*.007` are arranged to follow it.)

In this tutorial, a single job is submitted once.

```
$ ./shoot 1 1
```

When using the Plasma Simulator, the status of submitted jobs can be checked with the `qstat` command.

```
$ qstat
```

If the simulation finishes normally, the following data are written to the output directory specified in `run/shoot` (for example: `DIR=/data/(user ID)/gkv_training/linear_test/`):

- `log/` – Calculation logs
- `cnt/` – Binary data for restart (continuation runs)
- `fxv/` – Binary data of distribution functions (at several positions along the field line)
- `phi/` – Binary data of potential, fluid moments, and entropy balance
- `hst/` – Standard output in ASCII format
- Others – Copies for backing up the execution environment

The output files are summarized in Appendix *List of GKV output*. For further details, see the source file `src/gkvp_out.f90`. By moving to the output directory, the generated data can be inspected.

```
$ cd /data/(User ID)/gkv_training/linear_test/
```

Analysis of output data

To analyze the output data, either manual processing is carried out or post-processing tools are used.

a. Manual processing
   - GKV output data are summarized in a list, so subsequent post-processing can be arranged as needed.
   - Standard output in ASCII format is relatively easy to handle.
   - Reading binary data with MPI domain decomposition is rather laborious; however, from `gkvp_f0.64` onward, the binary data have also been organized in Zarr format, which allows easy reading in Python.

In the GKV GitHub repository, the following two post-processing tools are provided and may be used:

b. `gkvfig`, a Python package for batch conversion of ASCII standard output in `hst/` into PDF

c. `diag_python`, a collection of post-processing Python scripts for analyzing binary data such as those in `phi/`

The items a–c are explained in order below.

a. Manual loading and plotting of GKV output data

Example 1: Time evolution of the squared amplitude of electrostatic potential fluctuations (from the ASCII output files `./hst/gkvp.eng.***`)

```
#!/usr/bin/env python3
import numpy as np
import matplotlib.pyplot as plt
data = np.loadtxt("./hst/gkvp.eng.001")
t = data[:,0]
eng_total = data[:,1]
plt.plot(t,eng_total,"+-")
plt.show()
```

Exponential growth of the fluctuations can be observed.

Example 2: Linear dispersion relation (`./hst/gkvp.dsp.***` ASCII output files; only for `calc_type = "lin_freq"`)

```
#!/usr/bin/env python3
import numpy as np
import matplotlib.pyplot as plt
data = np.loadtxt("./hst/gkvp.dsp.001")
```

(continues on next page)

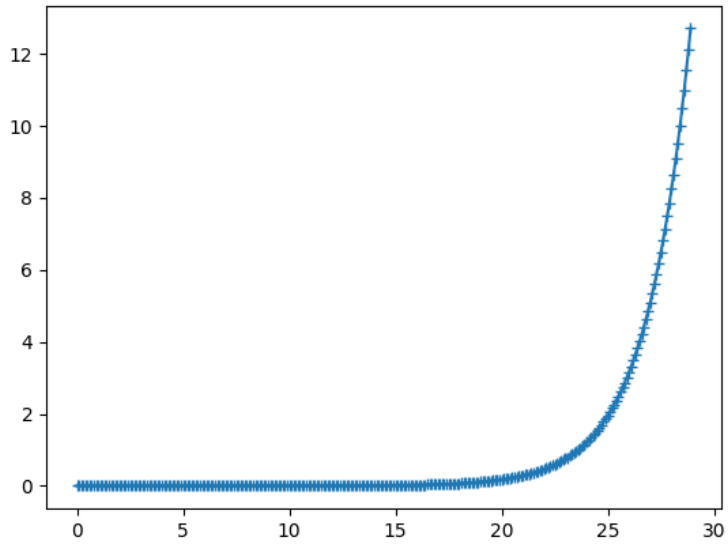

Fig. 1: Time evolution of the squared amplitude of electrostatic potential fluctuations

(continued from previous page)

```
ky = data[:,1]
gamma = data[:,3]
plt.plot(ky,gamma,"+-")
plt.show()
```

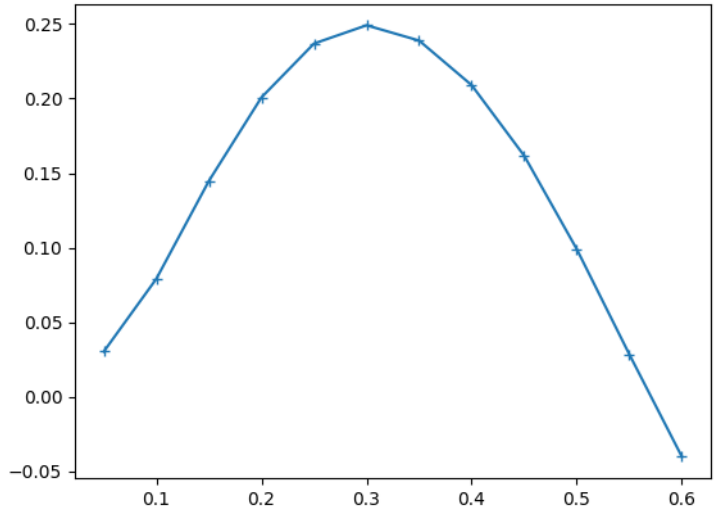

Fig. 2: Linear dispersion relation

Example 3: Dependence of electrostatic potential fluctuations on the coordinate along the magnetic field line (`./phi/gkvp.phi.***.zarr/` Zarr-format binary output files)

```
#!/usr/bin/env python3
import xarray as xr
import matplotlib.pyplot as plt
```

(continues on next page)

(continued from previous page)

```
ds = xr.open_mfdataset("./phi/gkvp.phi.*.zarr", consolidated=False)
zz = ds.zz[:]
phi = ds.sel(ky=0.25, method="nearest").phi[-1,:,0]
plt.plot(zz,phi.real,"+-")
plt.plot(zz,phi.imag,".-")
plt.show()
```

A ballooning structure localized around the bad-curvature region at $z = 0$ can be seen.

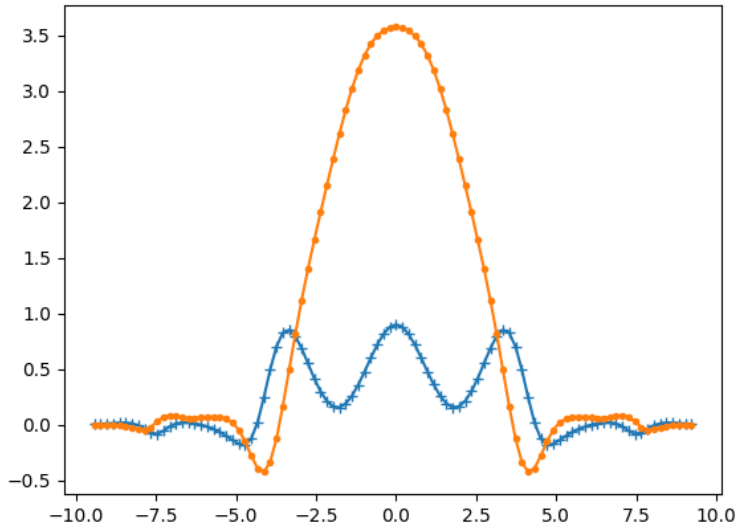

Fig. 3: Dependence of electrostatic potential fluctuations on the coordinate along the magnetic field line

b) `gkvfig`: a Python package for batch conversion of ASCII standard output in `hst/` into PDF

Since `gkvfig` has already been installed (see [*Setting up a Python virtual environment, Initial setup 2*]), it can be used as a command-line tool as follows.

(i) When the GKV output directory is specified by an absolute path

```
$ python -m gkvfig -d /data/(User ID)/gkv_training/linear_test/
```

(ii) When the GKV output directory is specified by a relative path

```
$ cd /data/(User ID)/gkv_training/linear_test/ # Move to the output directory
$ python -m gkvfig -d ./
```

The file `figpdf_YYYYMMDD_HHMMSS/fig_sdtout.pdf` is obtained, in which the ASCII standard output in `hst/` has been batch-converted into PDF.

```
(env01) [(User ID)@fes3 ~]$ cd /data/(User ID)/dkv_training/linear_test/
(env01) [(User ID)@fes3 linear_test]$ python -m gkvfig -d ./
/data/(User ID)/gkv_training/linaer_test/figpdf_20250723_163935/fig_stdout.pdf generated.
(env01) [(User ID)@fes3 linear_test]$ ls
cnt/                    gkvp.exe*          lib/      phi/
figpdf_20250723_163935/  gkvp_namelist.001  log/      src/
fxv/                    hst/               Makefile  sub.q.001*
```

(continues on next page)

(continued from previous page)

```
(env01) [(User ID)@fes3 linear_test]$ ls figpdf_20250723_163935/
data/  fig/  fig_stdout.pdf
(env01) [(User ID)@fes3 linear_test]$
```

As an example of linear simulation output, excerpts from `fig_stdout.pdf` are shown. The PDF can be viewed with **evince**.

```
$ cd figpdf_YYYYMMDD_HHMMSS/
$ evince fig_stdout.pdf
```

Distribution of magnetic field strength, dependence of linear growth rate and real frequency on poloidal wavenumber, time evolution of squared fluctuation amplitude are shown in Figs. 4, 5, 6, respectively. $z = 0$ corresponds to the outer side of the torus (bad-curvature region), and $z = \pi$ corresponds to the inner side of the torus (good-curvature region).

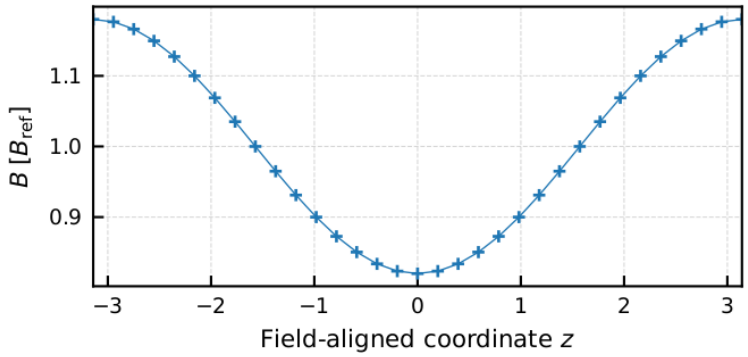

Fig. 4: Distribution of magnetic field strength along the field-line coordinate z

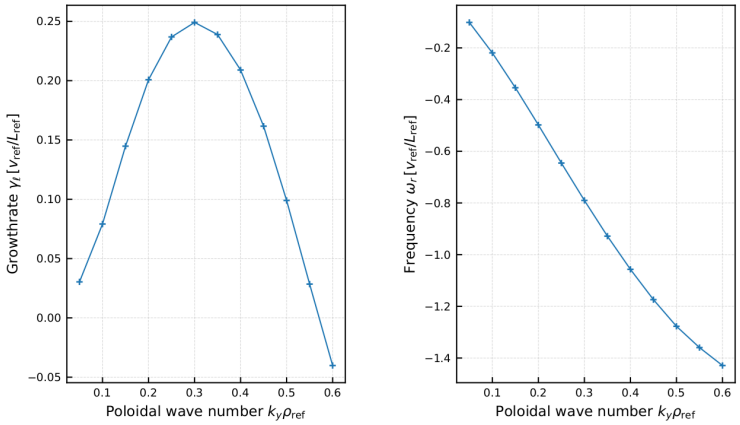

Fig. 5: Dependence of linear growth rate and real frequency on poloidal wavenumber

c) `diag_python`: a collection of post-processing Python scripts for analyzing binary data

`diag_python` is downloaded and unpacked in the GKV output directory `/data/(user ID)/gkv_training/linear_test/`.

```
$ cd /data/(User ID)/gkv_training/linear_test/
$ wget https://github.com/GKV-developers/diag_python/archive/refs/tags/f0.65_01.tar.gz -O
```

(continues on next page)

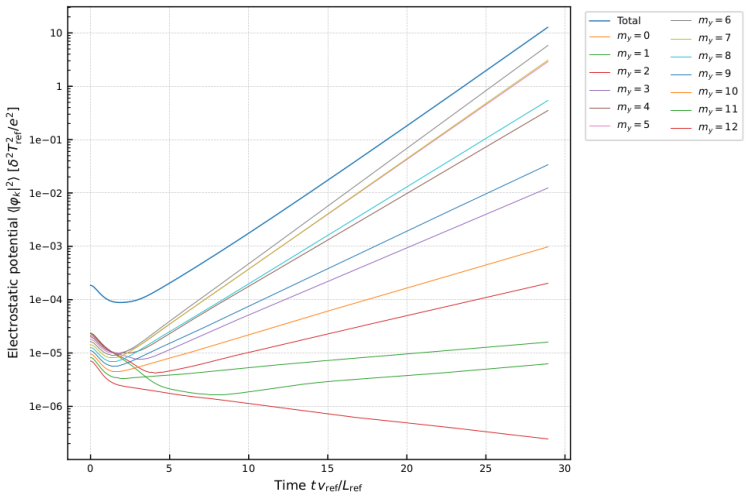

Fig. 6: Time evolution of squared fluctuation amplitude for each poloidal mode number

(continued from previous page)

```
$ diag_python-f0.65_01.tar.gz
$ tar xzvf diag_python-f0.65_01.tar.gz
```

Although the tools may be invoked from any familiar Python environment on the user side, this section illustrates the use of JupyterLab provided by the web front-end service **Open OnDemand** on the Plasma Simulator. [See "Plasma Simulator User Guide", Section 10.1.10.]

Note

Note: By using SSH port forwarding, it is also possible to start a Jupyter server on the Plasma Simulator and access it from the user's local PC. This is a general approach that can be used even on other supercomputer systems where a web front-end service such as Open OnDemand is not available.

Fig. 7: Selection of Open OnDemand on the Plasma Simulator

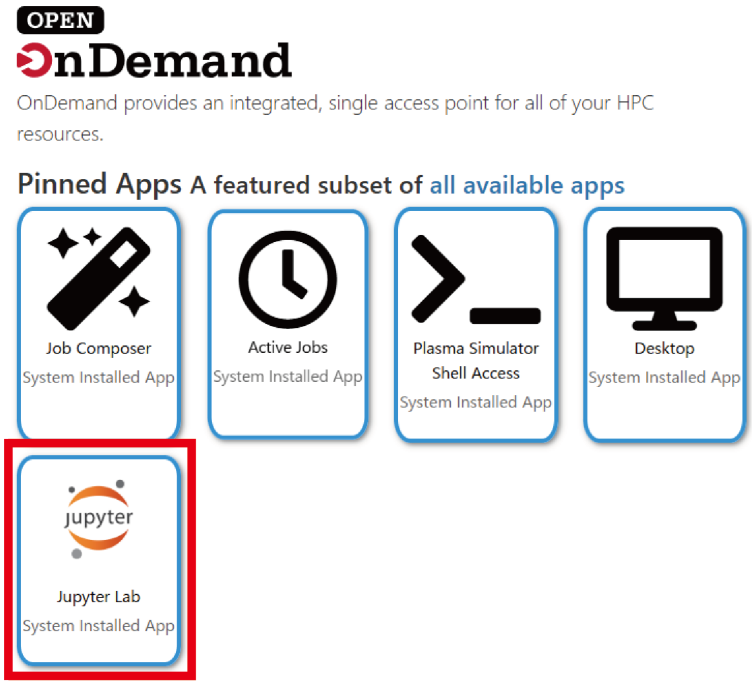

Fig. 8: Selection of JupyterLab on Open OnDemand

An initial configuration is performed so that the user-created Python virtual environment `env01` can be added to the Jupyter kernels launched by Open OnDemand. From Jupyter, a Terminal is started and `~/myvenv/env01/pyvenv.cfg` is edited (this setting allows access to the `ipykernel` environment used by Jupyter launched via Open OnDemand).

The value of `include-system-site-packages` is changed from `false` to `true`.

Listing 5: ~/myenv/env01/pyvenv.cfg

```
home = /usr/bin
include-system-site-packages = true
version = 3.12.9
executable = /usr/bin/python3.12
command = /usr/bin/python3.12 -m venv /home/(User ID)/myvenv/env01
```

Next, the Jupyter kernel for `env01` is registered by executing the following command in the Terminal.

```
$ source ~/myvenv/env01/bin/activate # Optional if env01 is already activated
$ python -m ipykernel install --user --name=env01
```

Note

Note: The environment that is added here is the Python environment active at the time this command is executed (which should be the one previously activated with `$ source ~/myvenv/env01/bin/activate`). The suffix `--name=env01` specifies the kernel name registered in Jupyter and can be changed arbitrarily, but it is generally safest to keep it consistent with the name of the virtual environment. If the registration is successful, the `env01` kernel appears in the JupyterLab Launcher.

In addition, a symbolic link is created so that the `/data/(user ID)/` directory can be accessed.

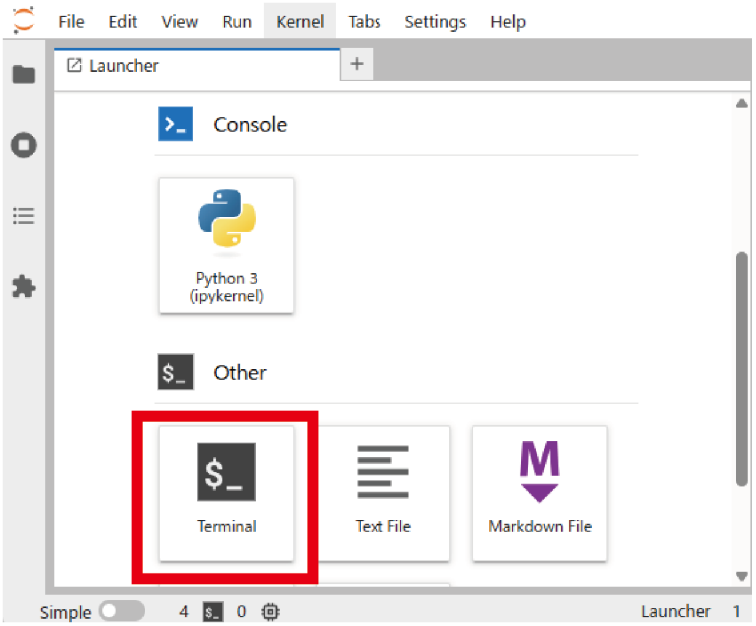

Fig. 9: Launching a Terminal in Jupyter.

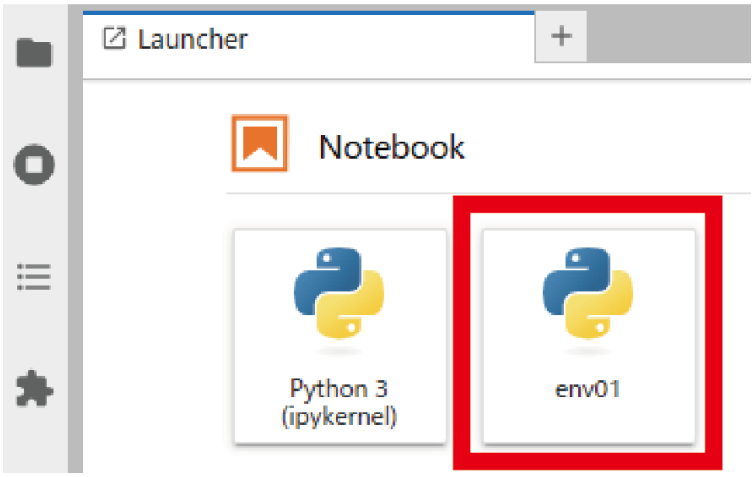

Fig. 10: Display of the env01 kernel in the JupyterLab Launcher

```
$ ln -s /data/(User ID)/ ~/data
```

Supplementary note 1. Handling Jupyter kernels

- Registration of a Jupyter kernel from a Python environment (for example, registering it with the kernel name `env01`)

```
$ python -m ipykernel install --user --name=env01
```

- Checking the list of available Jupyter kernels

```
$ jupyter kernelspec list
```

- Removal of a Jupyter kernel (for example, deleting the `env01` kernel)

```
$ jupyter kernelspec remove env01
```

Note

Note: Jupyter itself is an integrated development environment that supports not only Python but also many other languages such as R and Julia. When registering a kernel, one can think of it as submitting an additional "registration request" from the user's Python environment to the Jupyter installation on the system side, using `ipykernel`. On the other hand, management of kernels that have already been registered is performed with the `jupyter kernelspec` command.

Supplementary note 2. Handling symbolic links

- Example of creating a symbolic link:

```
$ ln -s /data/(User ID)/ ~/data
```

- Example of deleting a symbolic link:

```
$ unlink ~/data
```

Note

Note: Symbolic links are removed with `unlink`. Although `rm` can also be used, there is a risk of accidentally deleting the directory itself, so the use of `unlink` is strongly recommended.

Example that must be strictly avoided (should never do): If `$ rm ~/data/` is mistakenly executed, the `rm` command is applied not to the symbolic link `~/data` itself, but to its target `~/data/ = /data/(user ID)/`.

The basic operations for using diag_python are outlined below.

- The notebook `main.ipynb` is opened.
- A cell is selected, and its contents are executed with `[Shift] + [Enter]`.
- All intermediate variables and similar information remain in memory. To start again from the beginning, the menu items **[Kernel] – [Restart Kernel and Clear All Outputs]** are chosen from the top menu.

By clicking on the `env01` area shown below Fig. 11, the Jupyter kernel selection screen shown in Fig. 12 can be displayed.

Note

Note: The first cell loads the paths of the Zarr files and of the header, namelist, and metric data, and performs the initial setup of various geometry data. Execution of this cell is essential.

Note

Note: Cells from the second one onward are executed as needed according to the desired analysis.

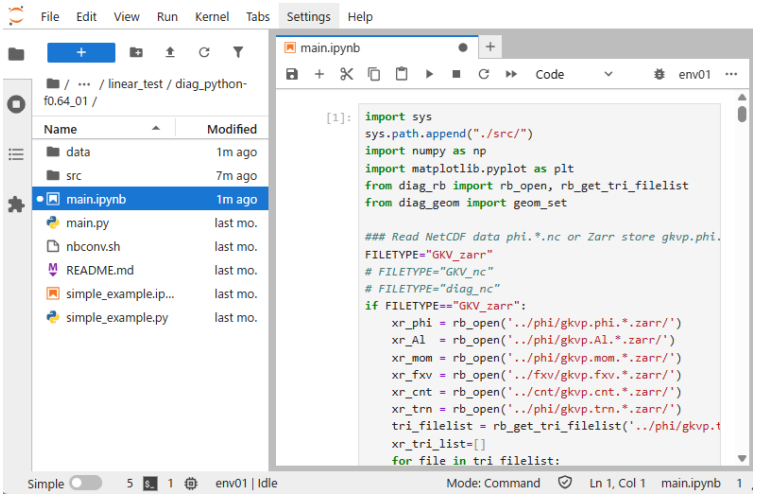

Fig. 11: Display of main.ipynb

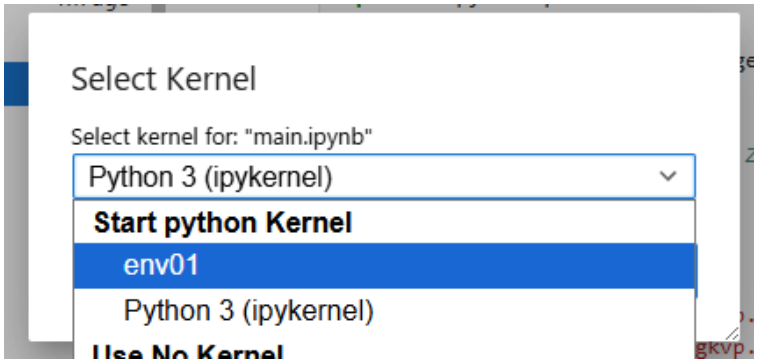

Fig. 12: Switching of Jupyter kernels

An example of a Python script for using `diag_python` (`simple_example.ipynb`) is shown. In `simple_example.ipynb`, the following operations are performed:

- The path is extended so that the Python scripts in `diag_python/src/` can be imported.
- The Zarr-format data `gkvp.phi.*.zarr/` are loaded (using xarray).

- Information such as the namelist is loaded and magnetic-configuration data are constructed; these are used for calculations such as flux-surface averages.
- Functions are imported from modules corresponding to the desired analysis and then executed.

In this example, the function `phiinxy` defined in `./src/out_mominxy.py` is imported. It computes the two-dimensional perpendicular distribution $\phi(y,x)$ of the electrostatic potential at a given time t[it] and at a given position zz[iz] along the magnetic field line.

The options `flag="display"` and `flag="savefig"` control whether a figure is displayed or saved. If `flag=None`, the data `x`, `y`, and $\phi(y,x)$ are returned, so users can plot them in any preferred way.

For further details, see the help text by calling `help(phiinxy)`.

Listing 6: simple_example.ipynb

```
### Append diag_python packages to path ###
import sys
sys.path.append("./src/")

### Read Zarr format data *.zarr/ by xarray ###
from diag_rb import rb_open
xr_phi = rb_open('../phi/gkvp.phi.*.zarr')

### Set geometric constants from GKV namelist etc. ###
from diag_geom import geom_set
geom_set(headpath='../src/gkvp_header.f90',
         nmlpath='../gkvp_namelist.001',
         mtrpath='../hst/gkvp.mtr.001')

# Plot phi[y,x] at t[it], zz[iz]
from out_mominxy import phiinxy
it = 250
iz = 48
phiinxy(it, iz, xr_phi, flag="display")
```

Nonlinear turbulence simulation

This section describes the execution and data analysis of nonlinear turbulence simulations as an application.

The physical parameters used in the linear calculation are modified for use in nonlinear turbulence simulations.

Listing 7: run/gkvp_namelist

```
&calct calc_type="nonlinear",   ! Change calculation type to nonlinear
...
&runlm e_limit = 3600.d0, &end  ! Increase wall-clock time limit [s] for the run
...
&nperi n_tht = 1,           ! Change box size along field line to ±pi in poloidal angle
       kymin = 0.05d0,      ! Box size in field-line label direction ly = pi / kymin
       m_j   = 4,           ! Radial box size lx = pi / kxmin
       del_c = 0.d0, &end   ! kxmin = |2 * pi * s_hat * kymin / m_j|
...                         ! Adjust so that lx and ly become reasonably close to each␣
↪other
```

The resolution and the number of MPI processes are also adjusted so that multiple modes are treated.

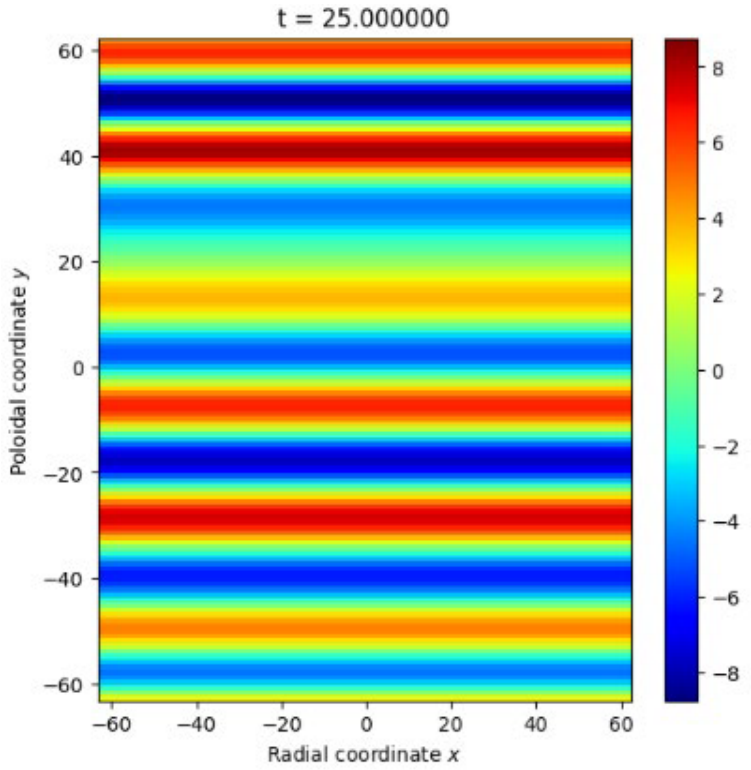

Fig. 13: Two-dimensional perpendicular distribution of the electrostatic potential

Listing 8: src/gkvp_header.f90

```
integer, parameter :: nxw = 84, nyw = 42
integer, parameter :: nx = 55, global_ny =27 ! 2/3 de-aliasing rule
integer, parameter :: global_nz = 24, global_nv = 32, global_nm = 15
...
integer, parameter :: nprocw = 2, nprocz = 3, nprocv = 4, nprocm = 2, nprocs = 1
```

The script `sub.q` is modified in accordance with the simulation runtime and the number of MPI processes.

Listing 9: run/sub.q

```
...
#PBS -P NIFS23KIST041      # Project name
#PBS -q A_S                # Queue class A_dev / A_S / A_M / A_L
#PBS -l walltime=01:30:00  # Wall-clock time limit (hh:mm:ss)
#PBS -l select=4:ncpus=128:mem=384gb:mpiprocs=12
                           # Resource request. See Section 8.1.2 of the Plasma Simulator␣
↪User Guide.
export OMP_NUM_THREADS=10  # Number of OpenMP threads per MPI process
...
```

The output destination directory is also changed.

Listing 10: run/shoot

```
DIR=/data/(user ID)/gkv_training/nonlinear_test/  # Data output directory after execution
```

Compilation is carried out with `make`, and the job is submitted with `./shoot 1 1`.

Note

Note: The resolution settings in this example are based on the ITGae-nl case in `benchmarks/`. It is useful to check whether the results obtained from one's own simulation agree with the reference data contained in `benchmarks/`.

In nonlinear simulations, however, the time series may not coincide step by step with the benchmark because of the accumulation of numerical errors and similar effects.

As an example of nonlinear simulation output, excerpts from `fig_stdout.pdf` are shown. In the time evolution of the fluctuation amplitude for each poloidal mode number (Fig. 14), it can be seen that the zonal flow with ky=0 (my=0) is dominant. In the time evolution of the ion heat transport flux Q_E for each poloidal mode number (Fig. 15), the mode with ky=0.2 (my=4) is mainly responsible for the transport. The evaluation of the entropy balance (Fig. 16) shows that drive (Θ/L_T) and dissipation (D) are balanced and the system reaches a steady state $(dS/dt \sim 0)$, and the error is kept at a low level.

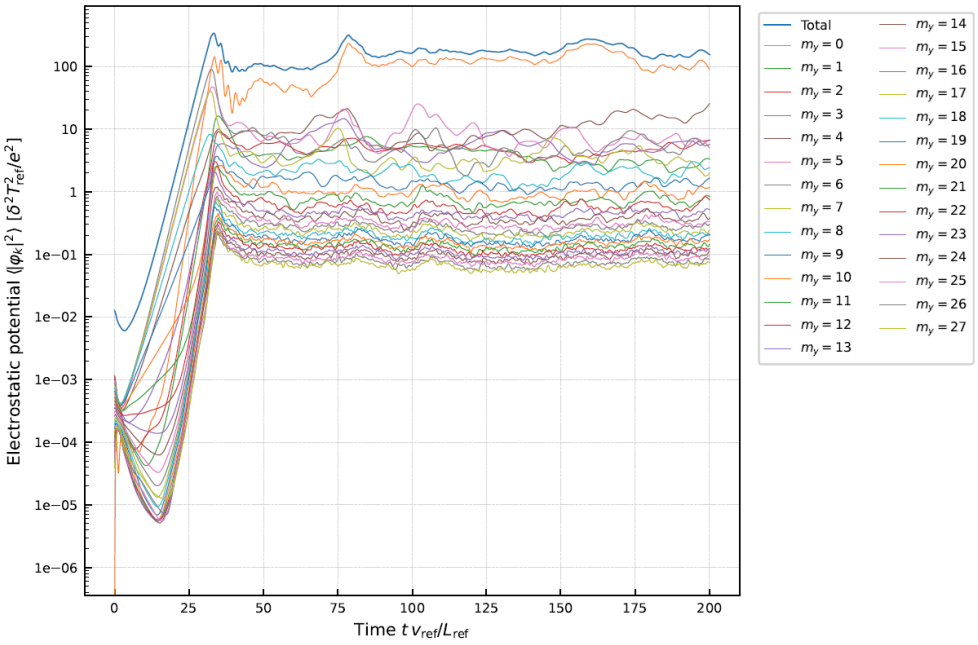

Fig. 14: Time evolution of the squared fluctuation amplitude for each poloidal mode number

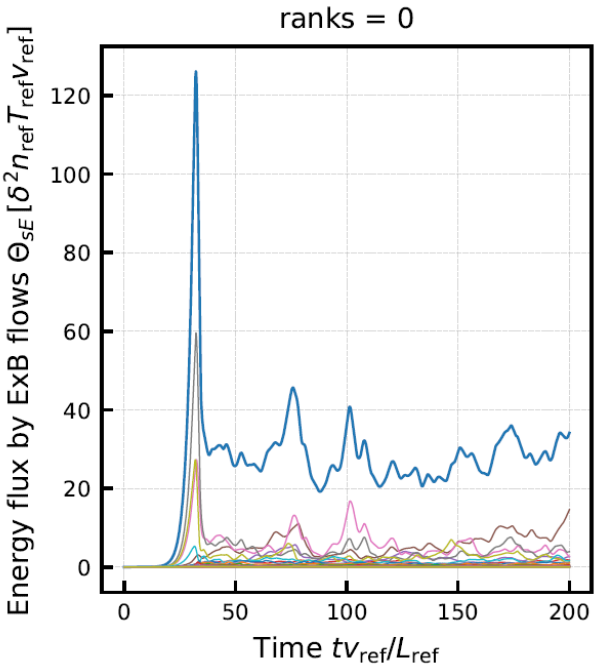

Fig. 15: Time evolution of ion heat transport flux for each poloidal mode number

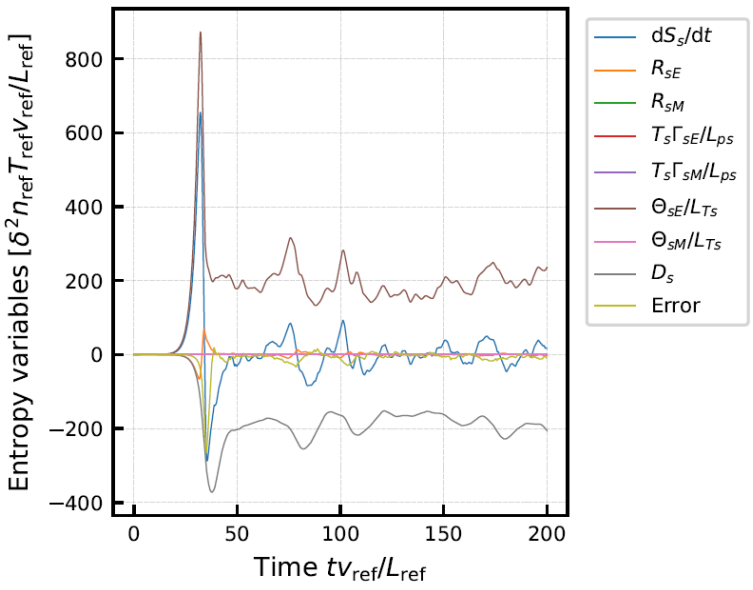

Fig. 16: Evaluation of entropy balance

UPDATE HISTORY

Date	Contents
December 11, 2025	Updated for gkvp_f0.65.
November 26, 2025	Translated from LaTeX to MkDocs and Read the Docs.
March 15, 2018	Add an explanation on adiabatic electron/ion model.
March 8, 2018	First draft based on gkvp_f0.48.

BIBLIOGRAPHY

[1-1] T.-H. Watanabe and H. Sugama. Velocity–space structures of distribution function in toroidal ion temperature gradient turbulence. *Nucl. Fusion*, 46(1):24, Dec 2005. doi:10.1088/0029-5515/46/1/003.

[1-2] X. Garbet, Y. Idomura, L. Villard, and T.-H. Watanabe. Gyrokinetic simulations of turbulent transport. *Nucl. Fusion*, 50(4):043002, Mar 2010. doi:10.1088/0029-5515/50/4/043002.

[1-3] M. A. Beer, S. C. Cowley, and G. W. Hammett. Field-aligned coordinates for nonlinear simulations of tokamak turbulence. *Phys. Plasmas*, 2(7):2687–2700, Jul 1995. doi:10.1063/1.871232.

[1-4] S. Maeyama, T.-H. Watanabe, M. Nakata, M. Nunami, Y. Asahi, and A. Ishizawa. Rotating flux-tube model for local gyrokinetic simulations with background flow and magnetic shears. *J. Comput. Phys.*, 522:113595, 2025. doi:https://doi.org/10.1016/j.jcp.2024.113595.

[1-5] H. Sugama, T.-H. Watanabe, and M. Nunami. Linearized model collision operators for multiple ion species plasmas and gyrokinetic entropy balance equations. *Phys. Plasmas*, 16(11):112503, Nov 2009. doi:10.1063/1.3257907.

[1-6] M. Nakata, M. Nunami, T.-H. Watanabe, and H. Sugama. Improved collision operator for plasma kinetic simulations with multi-species ions and electrons. *Comput. Phys. Commun.*, 197:61–72, 2015. doi:https://doi.org/10.1016/j.cpc.2015.08.007.

[3-1] S. Gill. A process for the step-by-step integration of differential equations in an automatic digital computing machine. *Proc. Cambridge Philosophical Soc.*, 47(1):96–108, 1951. doi:10.1017/S0305004100026414.

[3-2] S. Maeyama, T.-H. Watanabe, Y. Idomura, M. Nakata, and M. Nunami. Implementation of a gyrokinetic collision operator with an implicit time integration scheme and its computational performance. *Comput. Phys. Commun.*, 235:9–15, 2019. doi:https://doi.org/10.1016/j.cpc.2018.07.015.

[3-3] S. Maeyama, T.-H. Watanabe, Y. Idomura, M. Nakata, M. Nunami, and A. Ishizawa. Improved strong scaling of a spectral/finite difference gyrokinetic code for multi-scale plasma turbulence. *Parallel Comput.*, 49:1–12, 2015. doi:https://doi.org/10.1016/j.parco.2015.06.001.

[B-1] H. Sugama, T.-H. Watanabe, and M. Nunami. Linearized model collision operators for multiple ion species plasmas and gyrokinetic entropy balance equations. *Phys. Plasmas*, 16(11):112503, Nov 2009. doi:10.1063/1.3257907.

[B-2] S. Maeyama, A. Ishizawa, T.-H. Watanabe, M. Nakata, N. Miyato, M. Yagi, and Y. Idomura. Comparison between kinetic-ballooning-mode-driven turbulence and ion-temperature-gradient-driven turbulence. *Phys. Plasmas*, 21(5):052301, May 2014. doi:10.1063/1.4873379.

[B-3] M. Nakata, T.-H. Watanabe, and H. Sugama. Nonlinear entropy transfer via zonal flows in gyrokinetic plasma turbulence. *Phys. Plasmas*, 19(2):022303, Feb 2012. doi:10.1063/1.3675855.