Plasma Science and Technology

PAPER

Implementation of field-aligned coordinates in a semi-Lagrangian gyrokinetic code for tokamak turbulence simulation

To cite this article: Lei YE *et al* 2018 *Plasma Sci. Technol.* **20** 074008

Plasma Sci. Technol. **20** (2018) 074008 (14pp) https://doi.org/10.1088/2058-6272/aac013

Implementation of field-aligned coordinates in a semi-Lagrangian gyrokinetic code for tokamak turbulence simulation

Lei YE (叶磊)[1,3], **Xiaotao XIAO** (肖小涛)[1,3], **Yingfeng XU** (徐颖峰)[1], **Zongliang DAI** (戴宗良)[2] **and Shaojie WANG** (王少杰)[2]

[1] Institute of Plasma Physics, Chinese Academy of Science, Hefei 230031, People's Republic of China
[2] Department of Modern Physics, University of Science and Technology of China, Hefei 230026, People's Republic of China

E-mail: lye@ipp.ac.cn and xtxiao@ipp.ac.cn

Received 21 December 2017, revised 4 April 2018
Accepted for publication 19 April 2018
Published 26 June 2018

Abstract

Field-aligned coordinates have been implemented in the gyrokinetic semi-Lagrangian code NLT, Ye *et al* (2016 *J. Comput. Phys.* **316** 180), to improve the computational efficiency for the numerical simulations of tokamak turbulence and transport. 4D B-spline interpolation in field-aligned coordinates is applied to solve the gyrokinetic Vlasov equation. A fast iterative algorithm is proposed for efficiently solving the quasi-neutrality equation. A pseudo transform method is used for the numerical integration of the gyro-average operator for perturbations with a high toroidal mode number. The new method is shown to result in an improved code performance for reaching a given accuracy. Some numerical tests are presented to illustrate the new methods.

Keywords: gyrokinetic simulation, field-aligned coordinates, quasi-neutrality equation, iterative algorithm, gyro-average operator

(Some figures may appear in colour only in the online journal)

1. Introduction

Large-scale gyrokinetic simulation is a powerful tool to study low-frequency ($\omega \ll \omega_c$) phenomena in tokamak plasmas [1, 2], such as drift waves [3], zonal flows [4] and energetic particle-driven modes [5, 6]. In gyrokinetic models, the charged particles are described by 5D distribution functions in gyro-center phase space $Z = (\boldsymbol{X}, v_\parallel, \mu)$ instead of 6D distribution functions in particle phase space $z = (\boldsymbol{x}, \boldsymbol{v})$ due to the decoupling of the fast gyro-motion of the particle from the slow drift motion of the gyro-center. Here, $\boldsymbol{x}$ and $\boldsymbol{v}$ are the particle position and velocity, respectively. $\boldsymbol{X}$ is the gyro-center position, $v_\parallel$ and μ are the particle parallel velocity and magnetic moment, respectively. This reduction in the dimension of phase space can greatly reduce the computational effort for direct numerical simulation. Meanwhile, the time step size can be greatly enlarged ($\omega_c \Delta t \sim 1$) owing to the suppression of the high-frequency waves, such as plasma oscillations and cyclotron waves. Here, ω_c is the gyro-frequency of the charged particle. In spite of these advantages, gyrokinetic simulation remains very expensive in terms of computational resources, especially for the global modeling of nonlinear multi-scale problems in large tokamaks such as ITER. Therefore, modern gyrokinetic codes, on the one hand, require state-of-the-art high-performance computing techniques to improve the computational efficiency on large-scale computing systems. On the other hand, various numerical methods have been developed and utilized to speed up the calculations.

Because of the presence of a strong equilibrium magnetic field, drift wave turbulence in tokamak plasma is characterized by a long wavelength along the field line, i.e. $k_\parallel qR \sim 1$ and a short wavelength perpendicular to it, i.e. $k_\perp \rho \sim 1$ [7]. Here, $k_\parallel$ and $k_\perp$ are wave numbers parallel and perpendicular to the magnetic field, respectively. ρ and R are the particle gyro-radius and major radius of the device, respectively. q is the safety factor. This kind of anisotropy can be exploited to improve the computational efficiency by minimizing the

[3] Authors to whom any correspondence should be addressed.

number of computational grid points along the magnetic field line for a given numerical accuracy. Taking this point into consideration, the field-aligned coordinate system is a suitable choice for tokamak turbulence simulation and has been employed by many gyrokinetic codes [8–10]. The main idea of constructing field-aligned coordinates is to express the parallel gradient through a partial derivative with respect to (w.r.t.) one of the dependent coordinates, i.e. $\nabla_\parallel \propto \partial/\partial l$. Here, $\nabla_\parallel = \mathrm{i}k_\parallel = \boldsymbol{b}\cdot\nabla$ with $\boldsymbol{b} = \boldsymbol{B}/B$ the unit magnetic field vector. l is one of the field-aligned coordinates that specifies the direction parallel to $\boldsymbol{B}$. It is clear that for a given accuracy, far fewer grid points for l can be used to resolve the long-wavelength perturbation along the field line. Traditionally, field-aligned coordinates are usually constructed based on the magnetic-flux coordinates [8], in which the magnetic field lines on a flux surface are straight lines. Recently, a so-called flux-coordinate independent (FCI) method [11] has been developed to construct field-aligned coordinates independent of the flux coordinates. The field-aligned coordinates designed by using the FCI method have some advantages compared to those by the traditional method and are especially suitable for studying the open field line region, such as the tokamak edge region containing seperatrix and X-point.

In this work, we describe the numerical implementation of the field-aligned coordinates for the gyrokinetic continuum code, the numerical Lie transform (NLT) for simulations of tokamak ion temperature gradient (ITG) turbulence. NLT is a δf global semi-Lagrangian code, which is based on the I-transform theoretical model where the unperturbed particle motion is decoupled from the perturbed part [12]. This simulation model leads to a fast fixed point interpolation algorithm, which can significantly accelerate the computational speed [13]. Moreover, due to the application of a high-dimensional (3D and 4D) B-spline interpolation technique, the NLT code does not suffer from splitting errors [14, 15], which are caused by time-splitting schemes for high-dimensional partial differential equations (PDEs). Nevertheless, NLT previously employs the magnetic-flux coordinates and is thus is not so efficient for studying nonlinear turbulence in tokamak plasmas. To improve the computational efficiency by employing field-aligned coordinates, almost all the components of the code have been upgraded including the Vlasov solver, field solver and gyro-average integrator, etc. These code modifications originate from not only the conversion of the governing equations due to coordinate transform, but new numerical algorithms related to the properties of the field-aligned coordinates. All these numerical aspects will be detailed in the remaining part of this paper, which is organized as follows. In section 2, the field-aligned coordinates and basic equations used in the NLT code are briefly introduced. In section 3, the numerical implementation of field-aligned coordinates in the NLT code are described including the gyrokinetic Vlasov solver, quasi-neutrality (QN) solver and gyro-average integrator. In section 4, some simulation results are presented. Finally, we summarize the main results in section 5.

2. The field-aligned coordinates and basic equations in the NLT code

2.1. Field-aligned coordinates and boundary conditions

The field-aligned coordinates can be constructed in a straightforward way from the magnetic-flux coordinates, which are closely related to the magnetic configuration in a tokamak. The equilibrium magnetic field $\boldsymbol{B}$ in a tokamak, which is composed of a toroidal component $\boldsymbol{B}_\mathbf{t}$ provided by current in external coils and a poloidal component $\boldsymbol{B}_\mathbf{p}$ generated by the plasma current, can be written in either a contravariant or a covariant form as [16]

$$\begin{aligned}\boldsymbol{B} &= \boldsymbol{B}_\mathbf{t} + \boldsymbol{B}_\mathbf{p}\\ &= q(\psi)\nabla\psi\times\nabla\theta + \nabla\zeta\times\nabla\psi\\ &= g(\psi)\nabla\zeta + I(\psi)\nabla\theta + g(\psi)\delta(\psi,\theta)\nabla\psi,\end{aligned}$$

where (ψ, θ, ζ) are the magnetic-flux coordinates, with ψ the poloidal magnetic flux, θ the poloidal angle and ζ the toroidal angle. Note that θ is the straight field line poloidal angle, so that $q(\psi) = \boldsymbol{B}\cdot\nabla\zeta/\boldsymbol{B}\cdot\nabla\theta$ is the safety factor. $2\pi\mu_0 g(\psi)$ is the poloidal current outside ψ; $2\pi\mu_0 I(\psi)$ is the total current inside ψ. $\delta(\psi, \theta)$ is related to the non-orthogonality of $\nabla\psi$ and $\nabla\theta$. The Jacobian of (ψ, θ, ζ) coordinates is

$$\mathcal{J} = |\nabla\psi\times\nabla\theta\cdot\nabla\zeta|^{-1} = \frac{qR^2}{g} = \frac{qg+I}{B^2}.$$

The parallel gradient operator in (ψ, θ, ζ) coordinates can be written as

$$\nabla_\parallel = \boldsymbol{b}\cdot\nabla = \frac{1}{\mathcal{J}B}\left(\frac{\partial}{\partial\theta} + q\frac{\partial}{\partial\zeta}\right), \tag{1}$$

which indicates that the parallel gradient depends on partial derivatives w.r.t. two independent variables θ and ζ. The flux coordinates (ψ, θ, ζ) are previously employed in the NLT code [12], and it is straightforward to construct field-aligned coordinates (x, y, z) based on flux coordinates (ψ, θ, ζ) [8], e.g.

$$x = \psi \tag{2}$$

$$y = q(\psi)\theta - \zeta \tag{3}$$

$$z = \theta. \tag{4}$$

with the inverse transform

$$\psi = x \tag{5}$$

$$\theta = z \tag{6}$$

$$\zeta = q(x)\cdot z - y. \tag{7}$$

It can be found that the Jacobian of the field-aligned coordinates (x, y, z) is the same as that of the flux coordinates (ψ, θ, ζ). The partial derivatives w.r.t. the field-aligned coordinates can be derived by using the chain rule

$$\frac{\partial}{\partial x} = \frac{\partial}{\partial\psi} + q'\cdot z\frac{\partial}{\partial\zeta} \tag{8}$$

$$\frac{\partial}{\partial y} = -\frac{\partial}{\partial\zeta} \tag{9}$$

$$\frac{\partial}{\partial z} = \frac{\partial}{\partial \theta} + q\frac{\partial}{\partial \zeta}, \tag{10}$$

with $q' = \mathrm{d}q/\mathrm{d}x$. In this work, the field-aligned coordinates (x, y, z) are implemented instead of (ψ, θ, ζ) in the NLT code. Let us make some comments about these field-aligned coordinates. Equation (9) illustrates that the partial derivative w.r.t. y is the opposite to that w.r.t. ζ, which is along the toroidal direction. From equations (1) and (10), it can be found that the parallel gradient can be expressed in terms of the partial derivative w.r.t. z, i.e.

$$\nabla_\parallel = \frac{1}{\mathcal{J}B}\frac{\partial}{\partial z}. \tag{11}$$

This is the main advantage of the field-aligned coordinates; the parallel direction is specified by only one of the coordinates and thus can be numerically resolved by far fewer grid points for long-wavelength perturbations along the field lines. From equation (2), it can be seen that x is still a radial label, which specifies a flux surface. However, equation (8) indicates that the radial partial derivative with the field-aligned coordinates, $\partial/\partial x$, is explicitly dependent on the toroidal partial derivative and field-aligned coordinate z, as long as $q' \neq 0$. A given single mode with toroidal mode number n, equation (8) can be expressed in wave number space in radial and toroidal directions as

$$k_x = k_\psi + nq'z. \tag{12}$$

It can be seen from equation (12) that the radial wave number in field-aligned coordinates can be enlarged due to the coordinate transform. Even if the radial wave number in flux coordinates is negligible, i.e. $\partial/\partial\psi = \mathrm{i}k_\psi \sim 0$, a finite wave number in x direction does exist, which is proportional to toroidal mode number n and q'. In this case, more grid points should be used to resolve the shorter wavelength structures in x than those in ψ, especially for high-n modes. Some methods have been developed for this problem [17, 18]. In this work, a pseudo transform method is proposed for the numerical integration of the gyro-average operator of high-n modes, which will be discussed in section 3.3.

Let us consider boundary conditions on a magnetic-flux surface for a given function $f = f(\boldsymbol{x})$, where $\boldsymbol{x}$ indicates spatial position. For two angle variables of the flux coordinates, θ and ζ, periodic boundary conditions are naturally employed. That is to say, on a given flux surface ψ (or x equivalently)

$$f(\psi, \theta, \zeta) = f(\psi, \theta + 2\pi, \zeta) \tag{13}$$

$$f(\psi, \theta, \zeta) = f(\psi, \theta, \zeta + 2\pi). \tag{14}$$

For field-aligned coordinates, the periodic boundary condition is applied in the y direction, while the twisted boundary condition is applied in the z direction

$$f(x, y, z) = f(x, y + 2\pi, z) \tag{15}$$

$$f(x, y, z + 2\pi) = f(x, y - 2\pi q(x), z). \tag{16}$$

If f is expressed with Fourier series in y, i.e.

$$f(x, y, z) = \sum_n \hat{f}_n(x, z)\mathrm{e}^{\mathrm{i}ny},$$

with

$$\hat{f}_n(x, z) = \frac{1}{2\pi}\int_0^{2\pi} f(x, y, z)\mathrm{e}^{-\mathrm{i}ny}\mathrm{d}y$$

the boundary condition of $\hat{f}_n$ in the z direction is written by

$$\hat{f}_n(x, z + 2\pi) = \hat{f}_n(x, z)\mathrm{e}^{-\mathrm{i}2\pi nq(x)}. \tag{17}$$

It should be noted from equation (16) or (17) that if f has no dependence on y (or for the 0th-order Fourier component $\hat{f}_0$), f (or $\hat{f}_0$) is periodic in z.

2.2. Basic equations in the NLT code

NLT is a gyrokinetic code, which is based on the I-transform theory. It evolves the perturbed gyro-center distribution function of ion as well as perturbed electrostatic potential by solving the gyrokinetic equation (GKE) and quasi-neutrality equation (QNE) self-consistently. In this subsection, a brief introduction to the fundamental equations used in the NLT code are presented. For more detail on the I-transform theory and simulation model used in the NLT code, we refer the reader to [12, 13, 19–21].

The basic idea of the I-transform perturbation method is to transform the gyro-center coordinate variables $\boldsymbol{Z}$ to a set of new variables $\boldsymbol{z}$, so that the perturbed part of particle motion are decoupled from the unperturbed part. Hence, the GKE in the new coordinates $\boldsymbol{z}$ depends only on the unperturbed orbit determined by the equilibrium field. The contribution of the perturbed field to the distribution function, which has been decoupled by the I-transform, can be calculated by using the pull-back transform.

Let us denote $\delta F(t, \boldsymbol{Z})$ and $\delta f(t, \boldsymbol{z})$ the perturbed distribution function in coordinates $\boldsymbol{Z}$ and $\boldsymbol{z}$, respectively. In a given time interval $[t_0, t_0 + \Delta t]$, the GKE for δf can be written as

$$\frac{\mathrm{d}_0}{\mathrm{d}t}\delta f(t, z(t)) = 0. \tag{18}$$

Here, the total time derivative $\mathrm{d}_0/\mathrm{d}t$ is taken along the unperturbed orbit, which is written as

$$\frac{\mathrm{d}_0}{\mathrm{d}t} = \frac{\partial}{\partial t} + \dot{\boldsymbol{z}}_0 \cdot \frac{\partial}{\partial \boldsymbol{z}}.$$

Here, $\dot{z}_0$ are the unperturbed motion of guiding center determined by the equilibrium field, which can be expressed in terms of the Poisson matrix J, and the unperturbed Hamiltonian H_0 as

$$\dot{z}_0^i = J^{ij}\partial_j H_0,$$

with

$$H_0 = \frac{1}{2}m_s v_\parallel^2 + \mu B_0.$$

The non-zero components of the Poisson matrix J^{ij} in gyro-center phase space $(x, y, z, v_\parallel, \mu, \xi)$ can be given by

$$J^{xy} = -J^{yx} = -\frac{qg + I}{e_s D},$$
$$J^{xz} = -J^{zx} = -\frac{g}{e_s D},$$
$$J^{yz} = -J^{zy} = -\frac{q'zg + g\delta}{e_s D},$$
$$J^{xv_\parallel} = -J^{-v_\parallel x} = \frac{g\delta}{e_s D},$$
$$J^{zv_\parallel} = -J^{v_\parallel z} = \frac{B}{m_s D}\left(1 - \frac{m_s v_\parallel}{e_s}\partial_\psi \frac{g}{B}\right),$$
$$J^{\xi\mu} = -J^{\xi\mu} = \frac{e_s}{m_s},$$

with ξ the gyro-angle, $D = qg + I + \rho_\parallel(I'g - g'I) - \rho_\parallel g^2 \partial_\theta \delta$ and $\rho_\parallel = m_s v_\parallel / e_s B$.

The perturbed distribution function in gyro-center coordinates $\delta F(\mathbf{Z})$ can be found from $\delta f(\mathbf{z})$ by applying the pull-back transform as follows

$$\delta F = \delta f + G_1^i \partial_i(\delta f + F_0) + \frac{1}{2} G_1^i \partial_i (G_1^j \partial_j)(\delta f + F_0), \quad (19)$$

with $\boldsymbol{G}_1(z, t)$ the first-order generating vector field and F_0 the equilibrium particle distribution function. This pull-back transform, which indicates the contribution of the perturbed electrostatic potential to the particle distribution function, is determined by the gauge function of the I-transform $S(t, z)$, which satisfies

$$\frac{\mathrm{d}_0}{\mathrm{d}t} S = e_\mathrm{i} \overline{\delta\phi}, \quad (20)$$

with $\delta\phi$ the perturbed electrostatic potential and $\overline{(\cdots)}$ the gyro-average operator defined by

$$\overline{\delta\phi}(\mathbf{X}, \mu, t) = \frac{1}{2\pi}\int_0^{2\pi} \delta\phi(\mathbf{X} + \boldsymbol{\rho}(\mu, \xi), t)\mathrm{d}\xi. \quad (21)$$

The generating vector $G_1(t, z)$ can be calculated from $S(z, t)$ by

$$G_1^i = \partial_k S J^{ki}. \quad (22)$$

The equilibrium distribution function F_0 in an axisymmetric torus should be defined by three constants of motion [22, 23] to keep $\mathrm{d}_0 F_0/\mathrm{d}t = 0$. In this work, a local Maxwellian

$$F_0 = \left(\frac{m}{2\pi T}\right)^{\frac{3}{2}} n_0 \exp\left(-\frac{\frac{1}{2}m v_\parallel^2 + \mu B}{T}\right),$$

is used for simplicity.

Equations (18), (20), (22) and (19) are the GKEs applied to the NLT code. Note that the total time derivatives in these equations are all taken along the unperturbed orbit. This property induced by I-transform theory can be exploited in numerical computation to significantly reduce the computational effort [12, 13].

The perturbed electrostatic potential is obtained by solving the QNE with the adiabatic electron response

$$e_\mathrm{i}\nabla\cdot\left(\frac{n_{0\mathrm{i}}}{B\omega_\mathrm{i}}\nabla_\perp \delta\phi\right) = -e_\mathrm{i}\delta n_\mathrm{i}^\mathrm{g} + e^2 n_{0\mathrm{e}}\frac{\delta\phi - \langle\delta\phi\rangle}{T_\mathrm{e}}, \quad (23)$$

with $\omega_\mathrm{i} = e_\mathrm{i}B/m_\mathrm{i}$ the gyro-frequency, $\rho_\mathrm{i} = \sqrt{m_\mathrm{i}T_\mathrm{i}}/e_\mathrm{i}B$ the gyro-radius, $n_{0\mathrm{i}}$ the equilibrium ion density and T_i the ion temperature. The left-hand side of equation (23) is the ion polarization density in the long-wavelength limit. This approximation is valid for the ITG turbulence with adiabatic electrons [24]. The first term on the right-hand side of equation (23) is the gyro-center density of the ion, which can be expressed as

$$\delta n_\mathrm{i}^\mathrm{g} = \int \overline{\delta F}\frac{B_{0\parallel}^*}{m_\mathrm{i}}\mathrm{d}\xi\mathrm{d}\mu\mathrm{d}v_\parallel. \quad (24)$$

Here, the over-bar denotes the gyro-average operator defined by equation (21). The second term on the right-hand side of equation (23) indicates perturbed electron density, with $\langle\cdots\rangle$ the magnetic-flux-surface average operator defined by

$$\langle\delta\phi\rangle(x) = \frac{\int_{-\pi}^{\pi}\mathrm{d}z\int_0^{2\pi}\mathrm{d}y\mathcal{J}\delta\phi(x, y, z)}{2\pi\int_{-\pi}^{\pi}\mathcal{J}\mathrm{d}z}. \quad (25)$$

By taking the flux average on both sides of equation (23), we get the equation for $\langle\delta\phi\rangle$ as

$$\left\langle e_\mathrm{i}\nabla\cdot\left(\frac{n_{0\mathrm{i}}}{B\omega_\mathrm{i}}\nabla_\perp\delta\phi\right)\right\rangle = -e_\mathrm{i}\langle\delta n_\mathrm{i}^\mathrm{g}\rangle. \quad (26)$$

The QNE (23) in the field-aligned coordinates is numerically solved with a fast iterative algorithm, which will be discussed in section 3.2.

3. Implementation of field-aligned coordinates in the NLT code

In the NLT code, all the physics quantities are discretized on the grid points. The computational domain in phase space can be denoted by $D^5 \triangleq L_x \times L_y \times L_z \times L_{v_\parallel} \times L_\mu$, where

$$L_x \triangleq [x_a, x_b],$$
$$L_y \triangleq [0, 2\pi),$$
$$L_z \triangleq [-\pi, \pi),$$
$$L_{v_\parallel} \triangleq [-v_\mathrm{c}, v_\mathrm{c}],$$
$$L_\mu \triangleq [0, \mu_\mathrm{c}].$$

Here, x_a and x_b indicate the inner and outer boundaries of radial domain, respectively. v_c and μ_c are the cut-off velocity and cut-off magnetic moment, respectively.

3.1. Semi-Lagrangian GKE solver based on multi-dimensional (4D and 3D) B-spline interpolation in field-aligned coordinates

The GKEs in the NLT code, equations (18) and (20), are both numerically solved by using the backward semi-Lagrangian method [25–28]. The perturbed part of distribution function $\delta f(t, x, y, z, v_\parallel, \mu)$ and gauge function $S(t, x, y, z, v_\parallel, \mu)$ are discretized on uniform phase space grids, which can be denoted as

$$\delta f(t, z_I) \doteq \delta f(t, x_i, y_j, z_k, v_{\parallel l}, \mu_m),$$
$$S(t, z_I) \doteq S(t, x_i, y_j, z_k, v_{\parallel l}, \mu_m),$$

where i, j, k, l and m are grid indices for each coordinate.

For a given time interval $[t^n, t^{n+1}]$, the value of $\delta f(t^{n+1}, z_I)$ can be found by applying the semi-Lagrangian method as

$$\delta f(t^{n+1}, z_I) = \delta f(t^n, z_I^*). \tag{27}$$

Here, z_I^* is the interpolation point, which can be obtained by tracing the phase space characteristic $z(t)$ from the grid $z(t^{n+1}) = z_I$ backward in time to $z(t^n) = z_I^*$. Since the gyrocenter dynamic is in the 4D phase space $(x, y, z, v_\parallel)$ with μ a constant of motion, 4D interpolation on the domain $D^4 \triangleq L_x \times L_y \times L_z \times L_{v_\parallel}$ is needed to estimate the value of $\delta f(t^n, z_I^*)$ from grid values $\delta f(t^n, z_I)$. Note that z_I^* for every grid depends only on the equilibrium field, so that it can be calculated once and saved as numerical tables for real-time computation [12]. Moreover, it is clear that the 4D interpolation for each grid is fixed point interpolation, which can be utilized to speed up the computation in each time step [13].

For 4D B-spline interpolation by tensor product, the major difference between the flux coordinates (ψ, θ, ζ) and field-aligned coordinates (x, y, z) lies in the boundary conditions. On a flux surface, the boundary conditions in (y, z), given by equations (15) and (16), are derived from physics boundary conditions in (θ, ζ), given by equations (13) and (14). Thus, these boundary conditions should be employed by the interpolation scheme. For B-spline interpolation, it is not difficult to implement a periodic boundary condition in y, as has been previously done for the flux coordinates θ and ζ [12]. While the twisted boundary condition in equation (16) cannot be implemented directly for B-splines. If one simply replaces this physics boundary condition with another one, e.g. periodic condition, then the unphysical boundary condition in z may induce unacceptable results, such as the non-periodical solution in (θ, ζ). As for the not-a-knot boundary condition, although it does not enforce any artificial boundary values or derivatives, the problem with it is that the interpolation accuracy near the boundary is rougher than that in the inner region. To resolve this difficulty, two buffer regions including several grids are added to both ends of the z grids. Hence, the extended z grids can be expressed as

$$L_z^{\text{ext}} = [-\pi - n_b \Delta z, \pi + (n_b - 1)\Delta z], \tag{28}$$

with n_b the number of grids in each buffer region. Δz is the grid size of z. The values of δf in the buffer region are calculated from those on the non-buffer grids by using equation (17). The 4D tensor B-splines are constructed on the extended domain of z including the buffer region with not-a-knot boundary condition. Figure 1 illustrates the buffer region on a flux surface where $n_b = 2$. Note that all interpolation points can be put into the region $L_y \times L_z$ on a flux surface by using equations (15) and (16).

Here, a numerical test is shown to illustrate the relationship between the interpolation errors and number of buffer grids. We use an analytical function

$$f(\psi, \theta, \zeta, v_\parallel) = \cos(m\theta - n\zeta)\exp(-v_\parallel^2), \tag{29}$$

with the toroidal mode number $n = 30$ and the poloidal mode number $m = \text{int}(nq)$. The function values on all the grids of the extended domain $D_{\text{ext}}^4 \triangleq L_x \times L_y \times L_z^{\text{ext}} \times L_{v_\parallel}$ are analytically given and the 4D tensor splines are constructed based on these values. Then, the values on the half grids $z_k' = z_k + \Delta z/2$ are interpolated. Figure 2 plots the interpolation errors of the 4D B-splines on these half grids with a different number of buffer grids. It can be seen that since the twist boundary condition is enforced in the buffer region, the interpolation errors near the boundary of L_z reduce rapidly as n_b increases. Therefore, it is enough to choose $n_b = 4$ or $n_b = 5$ to achieve the same level of accuracy between the boundary and inner region.

The process of solving equation (18) by using 4D B-spline interpolation in field-aligned coordinates can be summarized as follows.

Step 1. The values of $\delta f(t^n, x_i, y_j, z_k, v_{\parallel l})$ on all the grids of domain D^4 are known.

Step 2. Calculate all the values on the grids in the buffer region by using 1D DFT in y (with fixed x_i, $v_{\parallel l}$ and μ_m) according to equation (17). The details of this mapping will be given below. Note that spectral precision is kept in this process.

Step 3. Construct 4D B-splines by using the values of δf on the grids of extended domain D_{ext}^4.

Step 4. Compute $\delta f(t^{n+1}, x_i, y_j, z_k, v_{\parallel l})$ on all the grids of domain D^4 by 4D B-spline interpolation according to equation (27).

In Step 2 above, the function values in the buffer region are calculated strictly from the values in the non-buffer grids according to the twisted boundary condition by 1D Fourier transform. For example, $f(x_i, y_j, -\pi - \Delta z)$, $j = 1, 2, 3, \cdots n_y$, can be calculated from $f(x_i, y_j, \pi - \Delta z)$, $j = 1, 2, 3, \cdots n_y$, with n_y the grid number in y, as follows.

1. The function in the non-extended region can be expressed by Fourier series as

$$f(x_i, y, \pi - \Delta z) = \sum_n \hat{f}_n(x_i, \pi - \Delta z)\mathrm{e}^{-\mathrm{i}ny},$$
$$n = -n_y/2, -n_y/2 + 1, \cdots, n_y/2 - 1$$

   where $\hat{f}_n$ can be calculated by 1D DFT in y from $f(x_i, y_j, \pi - \Delta z)$, $j = 1, 2, 3, \cdots n_y$.
2. The function in the buffer region can also be expressed by Fourier series as

$$f(x_i, y, -\pi - \Delta z) = \sum_n \hat{f}_n'(x_i, -\pi - \Delta z)\mathrm{e}^{-\mathrm{i}ny}.$$

3. From equation (17), $\hat{f}_n'$ can be easily found by

$$\hat{f}_n'(x_i, -\pi - \Delta z) = \hat{f}_n(x_i, \pi - \Delta z)\mathrm{e}^{\mathrm{i}2\pi nq(x_i)}.$$

4. The values in the buffer grid $f(x_i, y_j, -\pi - \Delta z)$, $j = 1, 2, 3, \cdots n_y$ can be obtained by 1D inverse DFT of $\hat{f}_n'$.

The process of solving equation (20) is similar where 3D B-spline interpolation for $\overline{\delta\phi}(x, y, z, t^n)$ is applied on the

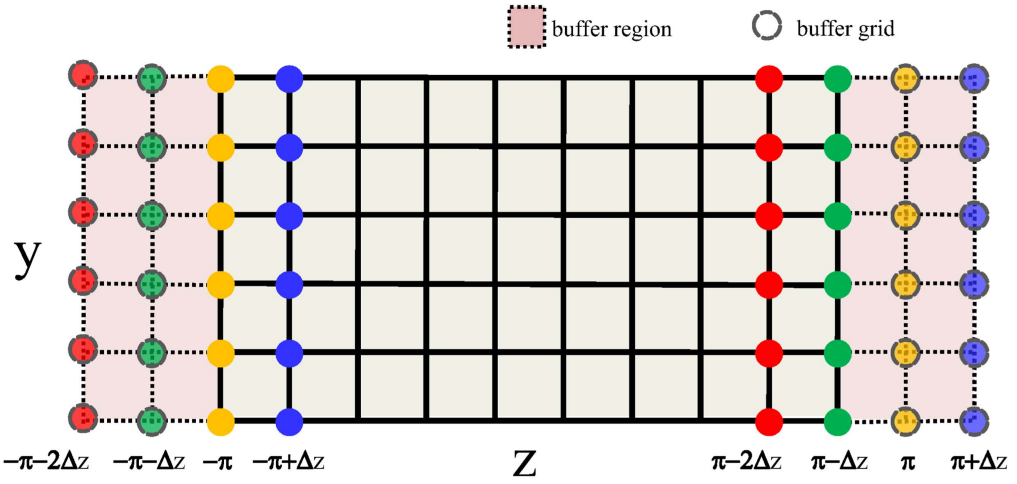

Figure 1. Computational grids of (y, z) on a flux surface. Buffer regions are indicated by dotted lines and red color. Values on the grid points in the buffer regions (circle with a dashed line) can be calculated by using corresponding values on the inner grids (circle without a line) with the same color.

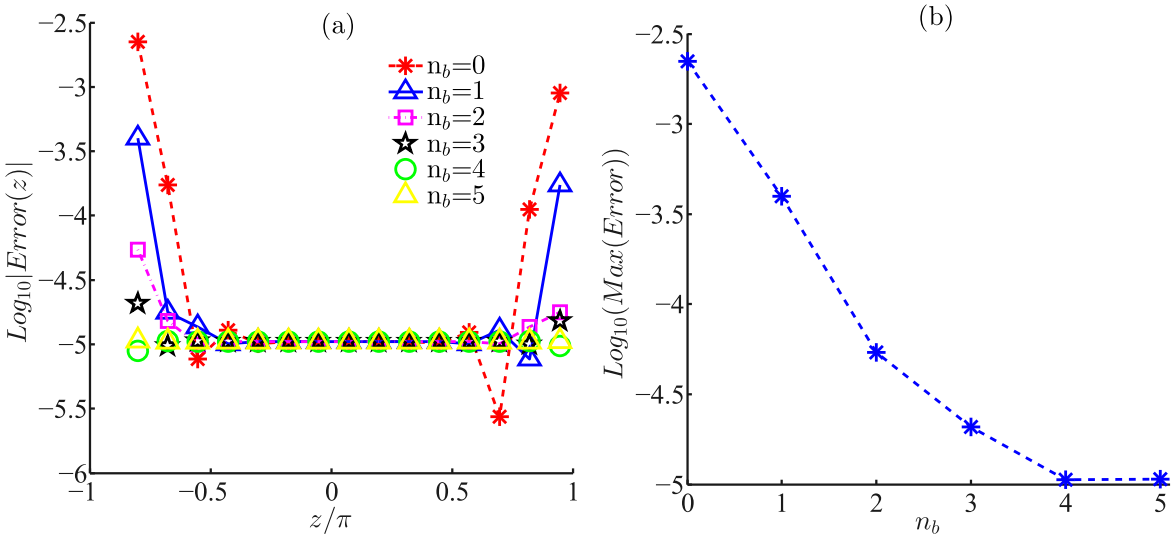

Figure 2. Relative 4D interpolation errors with a different number of buffer grids. (a) Relative errors in z with different $n_{\rm b}$. (b) Maximum errors with different $n_{\rm b}$.

extended grids. Then, the gyro-center distribution function δF can by explicitly calculated with the help of equation (19). The partial derivatives therein are approximated by the fourth-order central finite difference scheme, which can be written as

$$\frac{\partial f}{\partial \alpha}|_{\alpha=\alpha_i} = \frac{f_{i-2} - 8f_{i-1} + 8f_{i+1} - f_{i+2}}{12\Delta\alpha} + O(\Delta\alpha^4), \quad (30)$$

$$\frac{\partial f^2}{\partial^2 \alpha}|_{\alpha=\alpha_i} = \frac{-f_{i-2} + 16f_{i-1} - 30f_i + 16f_{i+1} - f_{i+2}}{12\Delta\alpha^2} + O(\Delta\alpha^4), \quad (31)$$

where α indicates x, y, z and $v_{\|}$. The midpoint predictor-corrector method [12] is applied to improve the global accuracy in time to the second order.

3.2. Fast iterative algorithm for the QNE

To obtain the self-consistent potential, the QNEs, equation (26) and equation (23) should be numerically resolved. The numerical solution of equation (26) for $\langle\delta\phi\rangle$ is straightforward and the same as before [12]. This equation comes from the adiabatic electron approximation and disappears in the kinetic electron model. In this paper, we focus on the numerical algorithm for solving the 3D QNE (23) in the field-aligned coordinates.

Equation (23) can be formally denoted by

$$\mathcal{L}[\delta\phi(x, y, z)] = s(x, y, z). \quad (32)$$

Here,

$$\mathcal{L} = e_{\rm i}\nabla\cdot\left(\frac{n_{0\rm i}}{B\omega_{\rm i}}\nabla_\perp\delta\phi\right) - e^2 n_{0\rm e}\frac{\delta\phi}{T_{\rm e}},$$

which is a second-order partial differential operator, which includes the terms of $\delta\phi$ and its partial derivatives. All these

partial derivatives of $\delta\phi$ come from the polarization density term in equation (23). s includes all the other terms independent of $\delta\phi$, which is given by

$$s = -e_{\rm i}\delta n_{\rm i}^{\rm g} - \frac{e^2 n_{0\rm e}\langle\delta\phi\rangle}{T_{\rm e}}.$$

For drift wave turbulence, the parallel derivative $\partial/\partial z$ is much smaller than the other two, $\partial/\partial x$ and $\partial/\partial y$, so that L can be divided into two parts

$$\mathcal{L} = \mathcal{L}_0 + \mathcal{L}_1,$$

where $\mathcal{L}_1$ is composed of all the partial derivatives of $\delta\phi$ w.r.t. z, i.e.

$$\mathcal{L}_1 \triangleq l_1(x,z)\frac{\partial}{\partial x}\frac{\partial}{\partial z} + l_2(x,z)\frac{\partial}{\partial y}\frac{\partial}{\partial z} + l_3(x,z)\frac{\partial^2}{\partial z^2} + l_4(x,z)\frac{\partial}{\partial z}; \tag{33}$$

and $\mathcal{L}_0$ includes all the other terms in $\mathcal{L}$

$$\mathcal{L}_0 \triangleq c_1(x,z)\frac{\partial^2}{\partial x^2} + c_2(x,z)\frac{\partial^2}{\partial y^2} + c_3(x,z)\frac{\partial}{\partial x}\frac{\partial}{\partial y} + c_4(x,z)\frac{\partial}{\partial x} + c_5(x,z)\frac{\partial}{\partial y} + c_6(x). \tag{34}$$

The details of the coefficients, c_i and l_i, and $s(x, y, z)$ can be found in the appendix. It is clear that for those modes with a high toroidal mode number, $\mathcal{L}_1 \ll \mathcal{L}_0$ is expected. In this case, $\mathcal{L}_1$ can be neglected for simplicity, as has been done in many works. However, for modes with a low toroidal mode number, $\mathcal{L}_1$ may be comparable to $\mathcal{L}_0$ and should be taken into consideration. Here, we propose an iterative method, which can solve the 3D PDE (32) efficiently by solving a set of 1D ordinary differential equations (ODEs) iteratively.

Note that all the coefficients of the partial derivatives, $c_i(x, z)$ and $l_i(x, z)$, are independent of y. The spectral method can be applied to equation (32) in the y direction. Let us denote

$$\delta\phi(x,y,z) = \sum_{n=-\frac{N}{2}}^{\frac{N}{2}} \hat{\delta\phi}_n(x,z)\mathrm{e}^{\mathrm{i}ny},$$

$$s(x,y,z) = \sum_{n=-\frac{N}{2}}^{\frac{N}{2}} \hat{s}_n(x,z)\mathrm{e}^{\mathrm{i}ny}.$$

The operators $\mathcal{L}_0$ and $\mathcal{L}_1$ in spectral space can be written as

$$\hat{\mathcal{L}}_0 \triangleq c_1\frac{\partial^2}{\partial x} + (\mathrm{i}nc_3 + c_4)\frac{\partial}{\partial x} - n^2c_2 + \mathrm{i}nc_5 + c_6,$$
$$\hat{\mathcal{L}}_1 \triangleq l_1\frac{\partial}{\partial x}\frac{\partial}{\partial z} + l_3\frac{\partial^2}{\partial z^2} + (\mathrm{i}nl_2 + l_4)\frac{\partial}{\partial z}.$$

Hence, equation (32) can be reduced to a set of 2D PDEs in spectral space as

$$(\hat{\mathcal{L}}_0 + \hat{\mathcal{L}}_1)\hat{\delta\phi}_n = \hat{s}_n. \qquad n = -\frac{N}{2}, -\frac{N}{2}+1, \cdots, \frac{N}{2}-1, \frac{N}{2} \tag{35}$$

Since $L_1 \ll L_0$, the above equation can be solved iteratively by using the following scheme

$$\hat{\mathcal{L}}_0\hat{\delta\phi}_n^{(m+1)}(x,z_k) = \hat{s}_n(x,z_k) - \hat{\mathcal{L}}_1\hat{\delta\phi}_n^{(m)}(x,z_k). \tag{36}$$

Here, the superscript (m), with $m = 0, 1, 2, 3\cdots$, indicates the number of iterations. Now, the $\hat{\mathcal{L}}_1$ operator, which includes all the partial derivatives w.r.t. z, acts on the known quantity $\hat{\delta\phi}_n^{(m)}$ so that the last term in equation (36) can be calculated explicitly. Note again that all the partial derivatives w.r.t. z are excluded from $\hat{\mathcal{L}}_0$, so equation (36) is in fact a set of ODEs w.r.t. x for given toroidal mode number n and z_k.

The iterative algorithm for the QNE can be summarized as follows.

Step 1. Given $\hat{s}_n(x_i, z_k)$. Set $m = 0$ and $\hat{\delta\phi}_k^{(0)} = 0$. Set desired convergence tolerance $\epsilon_{\rm tol}$ and maximal number of iterations $n_{\rm it}$.

Step 2. Calculate the second term on the right-hand side of equation (36) explicitly from $\hat{\delta\phi}_n^{(m)}$ with the finite difference method.

Step 3. For each toroidal mode number n and grid z_k, solve equation (36) w.r.t. x to get $\hat{\delta\phi}_n^{(m+1)}(x_i, z_k)$ by using the finite difference method.

Step 4. Calculate the difference between $\hat{\delta\phi}^{(m+1)}$ and $\hat{\delta\phi}^{(m)}$ as

$$\epsilon^{(m)} = \frac{\sum_{n,z_k,x_i}|\hat{\delta\phi}_n^{(m+1)}(x_i,z_k) - \hat{\delta\phi}_n^{(m)}(x_i,z_k)|}{\sum_{n,z_k,x_i}|\hat{\delta\phi}_n^{(m+1)}(x_i,z_k)|}.$$

Step 5. If $\epsilon^{(m)} > \epsilon_{\rm tol}$, set $m = m + 1$ and go back to Step 2. If $\epsilon^{(m)} \leqq \epsilon_{\rm tol}$ or $m = n_{\rm it}$, let $\hat{\delta\phi}_n = \hat{\delta\phi}_n^{(m+1)}$ and exit the loop.

Note that $\hat{\delta\phi}_k^{(0)} = 0$ is assumed in Step 1 to start the iteration, which indicates that all the parallel derivatives are neglected for the calculation of $\hat{\delta\phi}_k^{(1)}$. This approximation has been used in many gyrokinetic simulation codes. Numerical practices show that a fast convergence rate can be achieved for $\epsilon^{(m)}$ and one typical case is shown in figure 3 for ITG turbulence simulation.

The total computational cost of solving equation (36) for one iteration can be evaluated as $O(N_x^2 \times N_y \times N_z)$, with $O(N_x^2)$ the operations cost for each ODE and $N_y \times N_z$ the number of ODEs. The cost of FFT for computing $\hat{s}_n$ from s and $\delta\phi$ from $\hat{\delta\phi}_n$ is $O(N_y log_2 N_y \times N_z \times N_x)$. In the previous QN solver, which employs flux coordinates [12], the 3D PDE is decomposed into a set of 2D PDEs in (ψ, θ) and solved by using a combination of the pseudo-spectral method in the poloidal direction and the finite difference method in the radial direction. Thus, the total computational cost is $O((N_\psi \times N_\theta)^2 \times N_\zeta)$, with $O((N_\psi \times N_\theta)^2)$ the cost for each 2D PDE and N_ζ the number of 2D PDEs. Here, (N_x, N_y, N_z) and $(N_\psi, N_\theta, N_\zeta)$ are the number of grid points for each coordinate. It is clear that the new QN solver described in this paper is much more efficient than before.

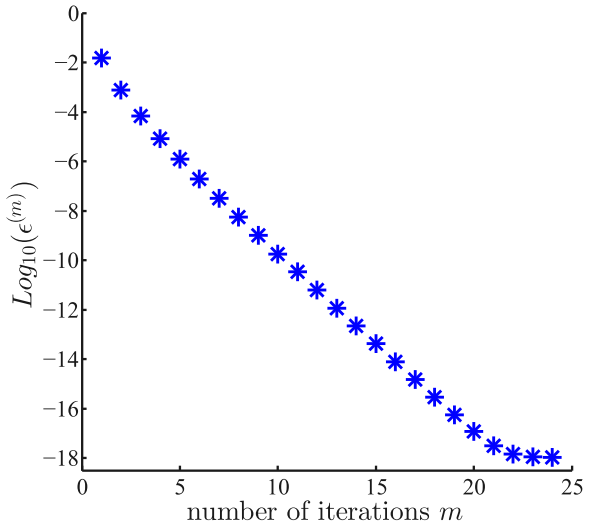

Figure 3. Convergence of the iterative algorithm for the QNE.

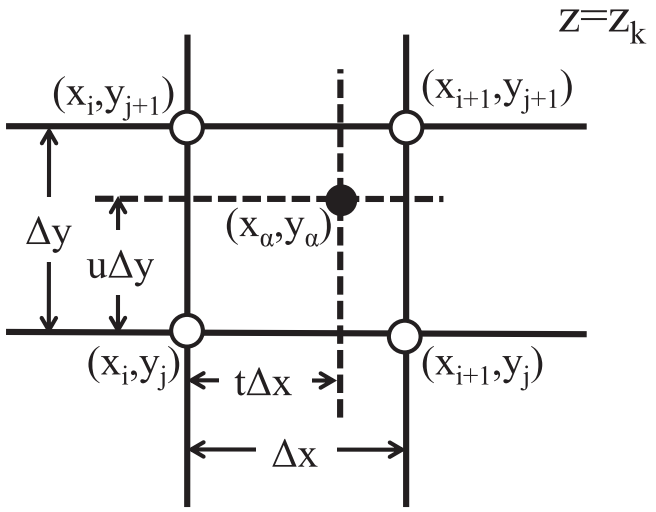

Figure 4. Bilinear interpolation on the (x, y) surface.

3.3. Pseudo transform method for the numerical integration of the gyro-average operator

The gyro-average operator is very important for gyrokinetic theory and simulation. The finite Larmor-radius effects can provide stabilizing effects on small-scale turbulence [29]. In gyrokinetic simulation code, there are mainly two kinds of numerical method for the gyro-average integration defined in equation (21) [30]. The first is the spectral method [10, 31], which transforms the integration to the spectral space. In simple geometry or flux-tube simulation, the gyro-average operator in Fourier space can be simply calculated by multiplying the spectra by the Bessel function. The other is the numerical integration method [9, 32, 33], which approximates the integration by the evaluation of several points on the gyro-ring. In the NLT code, a four-point discrete sum method [32] is used to approximate the integration efficiently.

It is easy to specify four points on a gyro-ring by using the field-aligned coordinates. The equilibrium magnetic field can be expressed as $\boldsymbol{B} = \nabla x \times \nabla y$. Thus, the unit vectors $\boldsymbol{e}_1 = \nabla x/|\nabla x|$, $\boldsymbol{e}_2 = \boldsymbol{b} \times \boldsymbol{e}_1$ and $\boldsymbol{b}$ are orthogonal to each other. Given gyro-center position $\boldsymbol{X}(x_g, y_g, z_g)$ and the gyro-radius ρ, the four points on the gyro-ring $\boldsymbol{X}_\alpha(\alpha = 1, 2, 3, 4)$ can be identified as

$$\boldsymbol{X}_\alpha(x_\alpha, y_\alpha, z_\alpha) = \boldsymbol{X} + \boldsymbol{\rho}_\alpha,$$

with

$$\begin{aligned}\boldsymbol{\rho}_1 &= \rho\boldsymbol{e}_1,\\ \boldsymbol{\rho}_2 &= -\rho\boldsymbol{e}_1,\\ \boldsymbol{\rho}_3 &= \rho\boldsymbol{e}_2,\\ \boldsymbol{\rho}_4 &= -\rho\boldsymbol{e}_2.\end{aligned}$$

x_α, y_α, z_α can be approximated by

$$\begin{aligned}x_\alpha &= x_g + \boldsymbol{\rho}_\alpha \cdot \nabla x \triangleq x_g + \Delta x_\alpha,\\ y_\alpha &= y_g + \boldsymbol{\rho}_\alpha \cdot \nabla y \triangleq y_g + \Delta y_\alpha,\\ z_\alpha &= z_g + \boldsymbol{\rho}_\alpha \cdot \nabla z \triangleq z_g + \Delta z_\alpha.\end{aligned}$$

The gyro-average integration of a function $f(\boldsymbol{X})$ can be approximated by using the four-point discrete sum as

$$\frac{1}{2\pi}\int_0^{2\pi} f(\boldsymbol{X} + \boldsymbol{\rho})\mathrm{d}\xi = \frac{1}{4}\sum_\alpha f(x_\alpha, y_\alpha, z_\alpha). \tag{37}$$

The off-grid values $f(x_\alpha, y_\alpha, z_\alpha)$ can be obtained through interpolation. The nearest-grid-point interpolation [10, 34, 35] can be applied to the z direction to improve the computational efficiency. Thus, $f(x_\alpha, y_\alpha, z_\alpha) \simeq f(x_\alpha, y_\alpha, z_k)$ with z_k is the grid point nearest to z_α. Usually, the bilinear interpolation can be applied to x and y

$$f(x_\alpha, y_\alpha, z_k) = \sum_{i=1}^{4} w_i f_i, \tag{38}$$

where, f_i are function values on the nearest four grids and w_i the weight for each grid value given by

$$\begin{aligned}w_1 &= (1 - t)(1 - u),\\ w_2 &= t(1 - u),\\ w_3 &= tu,\\ w_4 &= (1 - t)u,\end{aligned}$$

with $t = (x_\alpha - x_i)/\Delta x$ and $u = (y_\alpha - y_j)/\Delta y$. The bilinear interpolation on the (x, y) surface is illustrated in figure 4.

The linear interpolation precision is first order and the interpolation error is proportional to the partial derivatives of the interpolated function. For example, suppose f does not depend on y, then the linear interpolation error can be estimated by [36]

$$E_x \leqslant \frac{1}{8}\max_{x\in[x_i,x_{i+1}]}\left|\frac{\partial^2 f}{\partial x^2}\right|(\Delta x)^2. \tag{39}$$

From equations (8) and (12), it can be seen that the interpolation error, proportional to the radial derivatives, may be larger with field-aligned coordinates than that with flux coordinates. In this case, it is wise to do the interpolation in the flux coordinates. Here, we propose a pseudo coordinate transform method to resolve this problem, which does not

need to map the interpolated function $f(x, y, z)$ to the flux coordinates $f(\psi, \theta, \zeta)$.

The interpolation point $(x_\alpha, y_\alpha, z_k)$ can be mapped to the flux coordinates as $(\psi_\alpha, \theta_k, \zeta_\alpha)$ according to equations (5)–(7) with a virtual grid of $\psi_i = x_i$, $\zeta_j = y_j$ and $\theta_k = z_k$. Here, the virtual grid of (ψ, θ, ζ) is only used for interpolation error analysis; the function value is indeed not mapped to the grid points in the flux coordinates. Then, $f(\psi_\alpha, \theta_k, \zeta_\alpha)$ can be interpolated linearly in ψ

$$f(x_\alpha, y_\alpha, z_k) = f(\psi_\alpha, \theta_k, \zeta_\alpha) = (1 - t)f(\psi_i, \theta_k, \zeta_\alpha) + tf(\psi_{i+1}, \theta_k, \zeta_\alpha), \tag{40}$$

with an interpolation error

$$E_\psi \leqslant \frac{1}{8}\max_{\psi\in[\psi_i,\psi_{i+1}]}\left|\frac{\partial^2 f}{\partial\psi^2}\right|(\Delta\psi)^2. \tag{41}$$

This interpolation error is smaller than that in equation (39) if $\frac{\partial^2 f}{\partial x^2} > \frac{\partial^2 f}{\partial \psi^2}$. Now, what is needed is to evaluate $f(\psi_i, \theta_k, \zeta_\alpha)$ and $f(\psi_{i+1}, \theta_k, \zeta_\alpha)$. This will not be directly done in the flux coordinates. Note from equations (2)–(4) that the mapping between (ψ, θ) and (x, y) is identical mapping, so (x_i, z_k) are still on the grid point if (ψ_i, θ_k) are on the grid point. By remapping the two points $(\psi_i, \theta_k, \zeta_\alpha)$ and $(\psi_{i+1}, \theta_k, \zeta_\alpha)$ back to the field-aligned coordinates, which can be denoted by (x_i, y_{α_1}, z_k) and (x_i, y_{α_2}, z_k), respectively, the function value can be evaluated by 1D interpolation in y by

$$\begin{aligned} f(\psi_i, \theta_k, \zeta_\alpha) = f(x_i, y_{\alpha_1}, z_k) &= (1 - u_1)f(x_i, y_{j_1}, z_k) \\ &\quad + u_1 f(x_i, y_{j_1+1}, z_k), \\ f(\psi_{i+1}, \theta_k, \zeta_\alpha) = f(x_{i+1}, y_{\alpha_2}, z_k) &= (1 - u_2)f(x_{i+1}, y_{j_2}, z_k) \\ &\quad + u_2 f(x_{i+1}, y_{j_2+1}, z_k), \end{aligned}$$

with interpolation error

$$E_y \leqslant \frac{1}{8}\max_{y\in[y_j,y_{j+1}]}\left|\frac{\partial^2 f}{\partial y^2}\right|(\Delta y)^2.$$

Here, $y_{j_{1(2)}}$ and $y_{j_{1(2)}+1}$ are the two nearest grid points to the interpolation point $y_{\alpha_{1(2)}}$ and $u_{1(2)} = (y_{\alpha_{1(2)}} - y_{j_{1(2)}})/\Delta y$. Thus, the whole interpolation scheme in the field-aligned coordinates can be written by

$$\begin{aligned} f(x_\alpha, y_\alpha, z_k) &= (1 - t)[(1 - u_1)f(x_i, y_{j_1}, z_k) \\ &\quad + u_1 f(x_i, y_{j_1+1}, z_k)] + t[(1 - u_2)f(x_{i+1}, y_{j_2}, z_k) \\ &\quad + u_2 f(x_{i+1}, y_{j_2+1}, z_k)], \end{aligned} \tag{42}$$

with interpolation error

$$E_{\text{tot}} \leqslant E_\psi + E_y,$$

which is dependent on E_ψ but not E_x. The pseudo transform method is illustrated in figure 5.

With the pseudo transform method, the interpolation error can reduce from E_x to E_ψ. This is useful for the gyro-average integration of perturbations with a high toroidal mode number and if $k_x \gg k_\psi$. The key point is to perform radial interpolation in ψ instead of x to avoid the large wave number in x. The interpolation error in the toroidal direction is the same for ζ and y, as can be seen from equation (9). It should also be pointed out that for the continuum code, all the interpolation points and weights are independent of time. Thus, they can be computed and saved in the initialization stage.

The numerical tests are performed to illustrate the advantage of the new method. The test function is given by

$$f(\psi, \theta, \zeta) = \exp\left[-\frac{(\psi - \psi_0)}{\Delta(\psi_b - \psi_a)}\right]\cos(m\theta - n\zeta),$$

where ψ_0 is the reference flux surface with safety factor $q = 1.4$ and magnetic shear $\hat{s} = 0.8$. The toroidal mode number $n = 30$ and poloidal mode number $m = \text{int}(nq)$. The simulation domain in (x, y, z) is $[0.4a, 0.6a] \times [0, 2\pi/n] \times [0, 2\pi]$ with a the minor radius. In figure 6, the function is plotted with different radial variables ψ and x. It can be seen that the wave number is much larger in field-aligned coordinates, which is consistent with equation (12).

For given numbers of grid points (n_x, n_y, n_z), the numerical errors of the numerical gyro-average integration can be defined by

$$E = \frac{\Sigma_{i,j,k}|\bar{f}_{\text{num}}(x_i, y_j, z_k) - \bar{f}(x_i, y_j, z_k)|}{\Sigma_{i,j,k}|\bar{f}(x_i, y_j, z_k)|},$$

where $\bar{f}_{\text{num}}$ and $\bar{f}$ are the numerical and analytical gyro-averaged function of f. The summation is taken over all the grid points in the simulation domain. Figure 7 compares the numerical errors of the gyro-average integration between the newly-proposed method using equation (42) and the direct integration method using equation (38) for given grid numbers $n_y = 32$ and $n_z = 32$. It can be seen that to achieve the same numerical precision, many more grid points must be employed for the direct integration method. This is easy to understand; more grid points are needed to resolve the finer structure in the x direction.

In figure 8, the contour-plot of the numerical errors with the direct method are shown. It is clear that the error is dominant in the large z region, which is consistent with equations (8) and (12).

4. Simulation results

4.1. Linear ITG simulation

In this subsection, some linear ITG simulations are performed with the cyclone-test [37] parameters for code verification. The simulation parameters used here are set as close as possible to those in 10]. An analytical equilibrium with concentric circle flux surface is used with $R_0 = 1.67$ m, $a = 0.60$ m, $B_0 = 1.9$ T. The safety factor profile is given by

$$q(r) = \frac{0.86 + 2.27\left(\frac{r}{a}\right)^2}{\sqrt{1 - \left(\frac{r}{R_0}\right)^2}}.$$

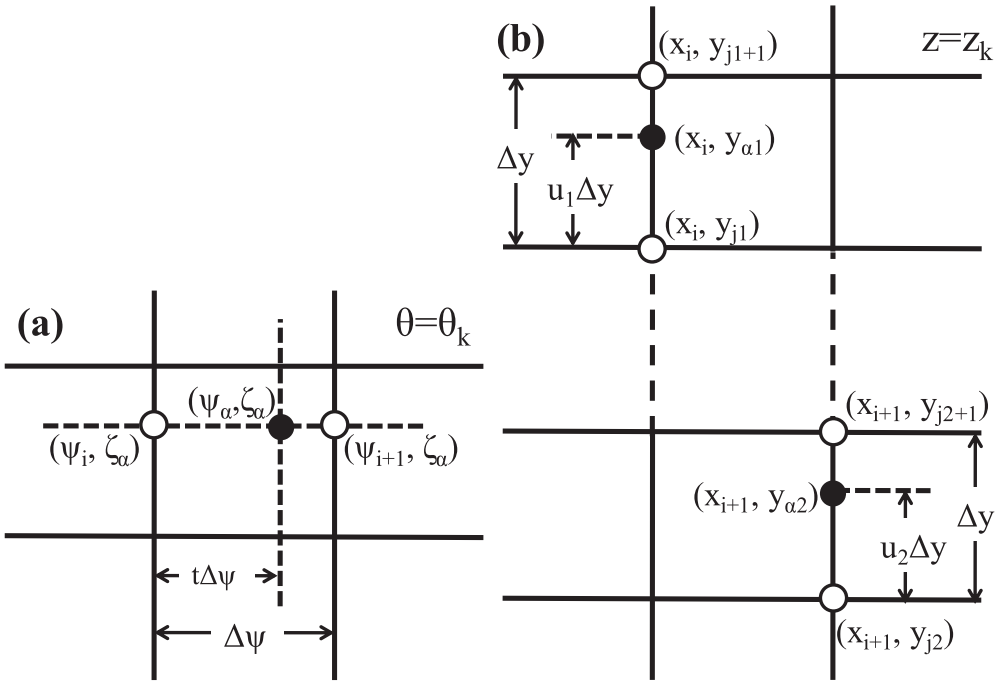

Figure 5. Pseudo transform method. (a) 1D radial interpolation in the flux coordinate. $f(x_\alpha, y_\alpha, z_k)$ is mapped to $f(\psi_\alpha, \theta_k, \zeta_\alpha)$, which is interpolated by $f(\psi_i, \theta_k, \zeta_\alpha)$ and $f(\psi_{i+1}, \theta_k, \zeta_\alpha)$. (b) 1D toroidal interpolation in field-aligned coordinates. $f(\psi_i, \theta_k, \zeta_\alpha)$ and $f(\psi_{i+1}, \theta_k, \zeta_\alpha)$ are mapped back to $f(x_i, y_{\alpha_1}, z_k)$ and $f(x_{i+1}, y_{\alpha_2}, z_k)$, respectively.

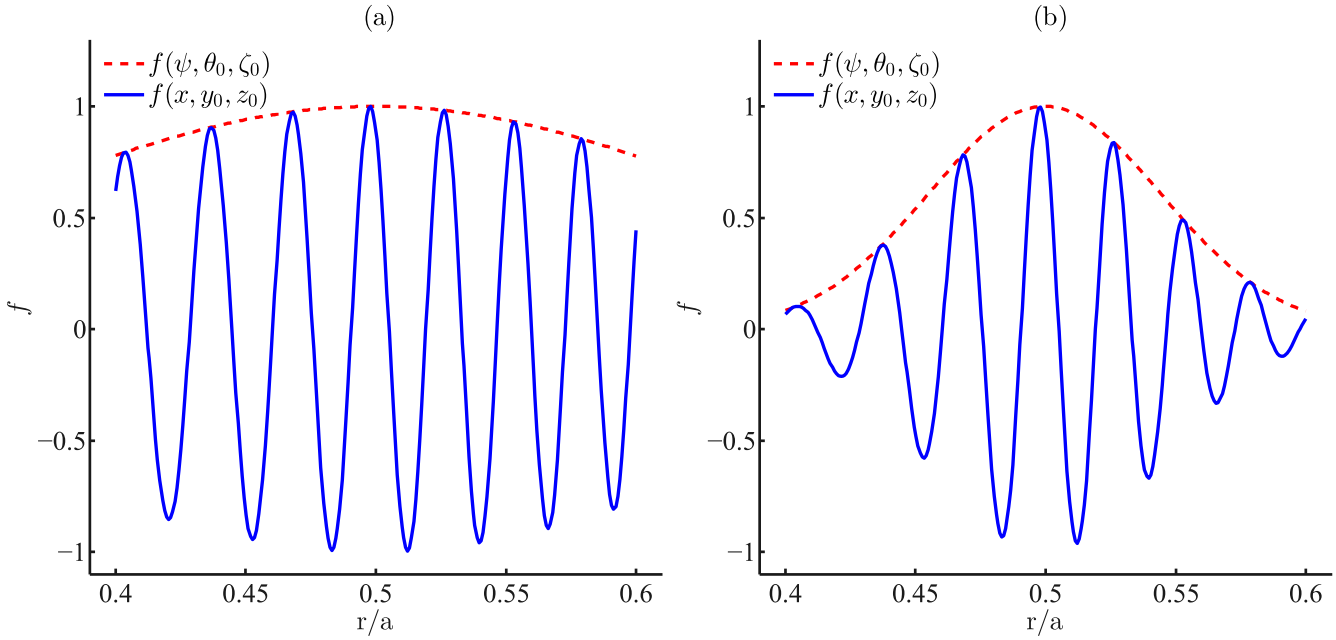

Figure 6. Test function f in the radial direction in the flux coordinates and field-aligned coordinates with the other two coordinates fixed. (a) $\Delta = 1$. (b) $\Delta = 0.1$.

The temperature and density profiles are given by

$$A(r) = A_0 \exp\left[-\kappa_A \frac{a}{R_0}\Delta_A \tan h\left(\frac{(r - r_0)/a}{\Delta_A}\right)\right],$$

with A indicating T and N. Here, $T_0 = 1.97$ keV, $N_0 = 10^{19}\,\mathrm{m}^{-3}$, $\kappa_T = 6.96$, $\kappa_N = 2.23$, $\Delta_T = \Delta_N = 0.3$. $\tau = T_e/T_i = 1$ is assumed. For more details on the simulation parameters and set-up, we refer the reader to [12]. In figure 9, the frequency and growth rate of the linear ITG modes given by the new version of the NLT code is plotted and compared with the results from the previous version of the NLT (with the flux coordinates) and GENE code. It can be seen that the linear simulation results from these three codes are consistent with each other. The linear mode structure with toroidal mode number $n = 19$ on a poloidal section is shown in figure 10.

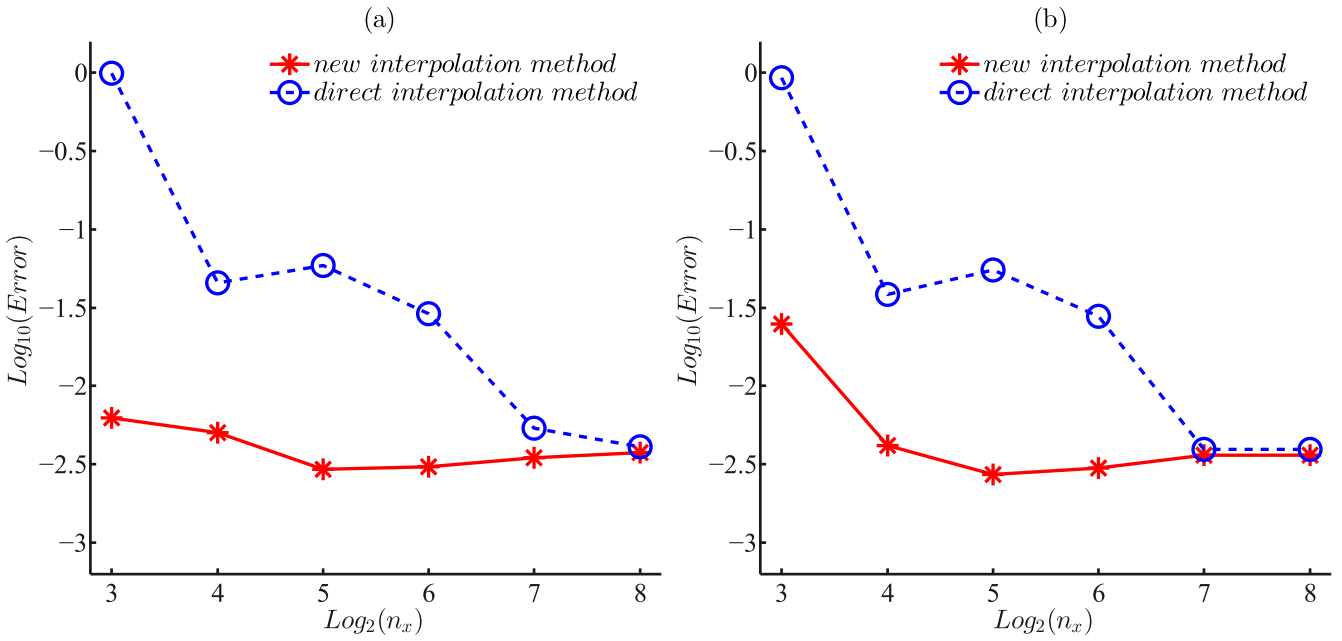

Figure 7. Numerical errors of the gyro-average integration on the reference flux surface with different number of grids in the radial direction. (a) $\Delta = 1$. (b) $\Delta = 0.1$.

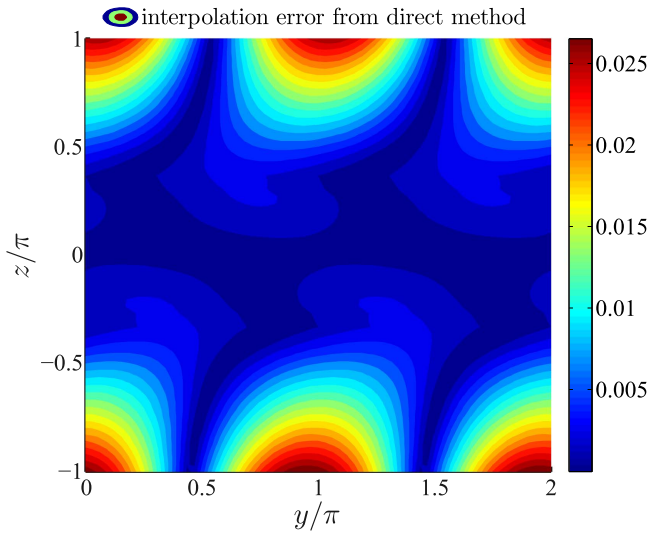

Figure 8. Interpolation errors on the reference flux surface for the direct method with $n_x = 64$ and $\Delta = 1.0$. Numerical error is larger near $z = \pm\pi$.

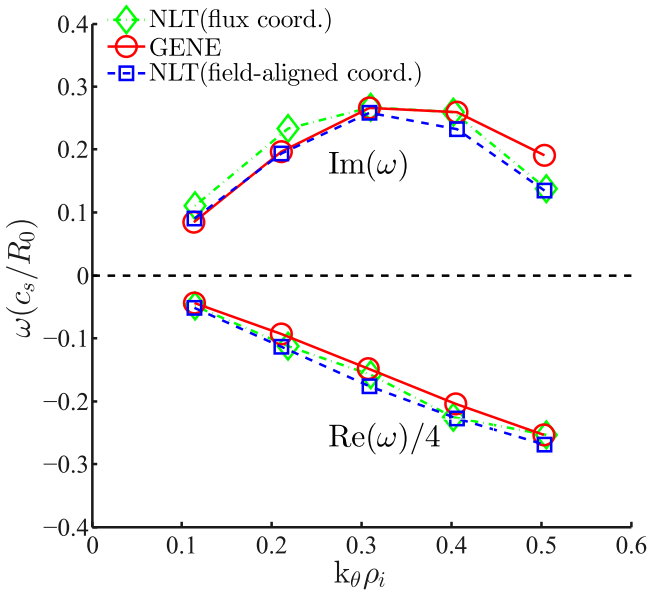

Figure 9. Comparison of frequency and growth rate for linear ITG mode.

4.2. Nonlinear ITG turbulence simulation

In this subsection, some simulations of nonlinear ITG turbulence are performed. The simulation parameters are chosen according to [38]. The equilibrium configuration is the same as that in the above linear test, except that the q profile is given by

$$q = 0.85 + 2.12\left(\frac{r}{a}\right)^2.$$

In this case, $q(r_0) = 1.40$ and $\hat{s}(r_0) = 0.78$, with $r_0 = 0.5a$ and $\hat{s} = (r/q)\mathrm{d}q/\mathrm{d}r$ the magnetic shear. The temperature and density profiles are given by

$$A(r) = A_0 \exp\left\{-\kappa_A \frac{a}{R_0}\left[(r - r_0)/a - \Delta_A \tan h\left(\frac{(r - r_i)/a}{\Delta_A}\right) - \Delta_A \tan h\left(\frac{(r - r_0)/a}{\Delta_A}\right)\right]\right\},$$

with $r \in [r_i, r_0]$ the simulation region in the radial direction. The relevant parameters are $T_0 = 1.97$ keV, $N_0 = 10^{19}$ m^{-3}, $\kappa_T = 6.91$, $\kappa_N = 2.22$, $\Delta_T = \Delta_N = 0.04$ and $\tau = 1$. The

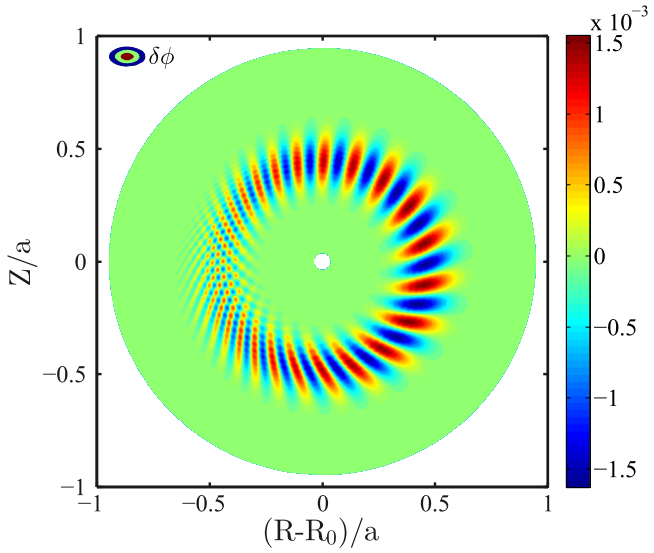

Figure 10. Linear ITG mode structure with single toroidal mode ($n = 19$).

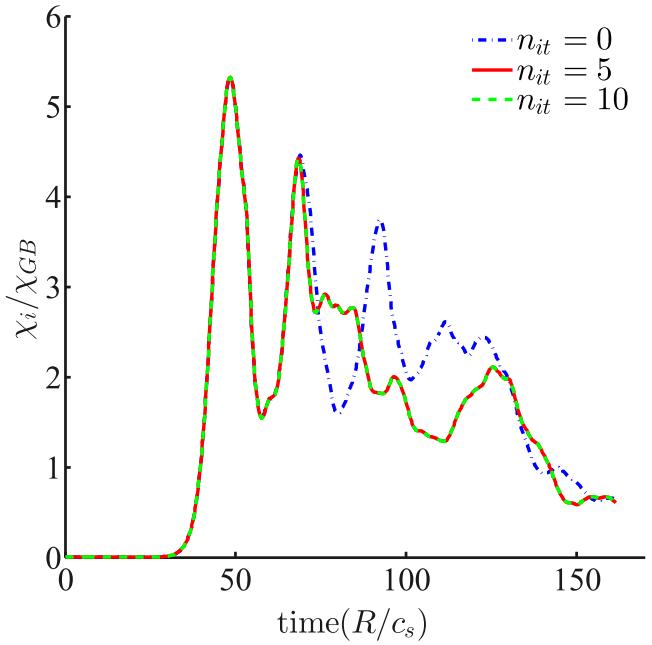

Figure 11. Time evolution of ion heat diffusivity with different iteration number $n_{\rm it}$ for the QN solver.

grid resolution $n_x \times n_y \times n_z \times n_{v_\parallel} \times n_\mu$ is set as $(129 \times 144 \times 16 \times 64 \times 32)$.

In figure 11, the ion heat diffusivity $\chi_{\rm i}$ with different iteration number $n_{\rm it}$ in the QN solver are shown. The mode structures in the linear and nonlinear phase are drawn in figure 12. It can be seen from figures 11 and 12 that the same results are obtained with $n_{\rm it} = 5$ and $n_{\rm it} = 10$ in this test case. The difference between the case with $n_{\rm it} = 0$ and the other two indicates that the derivatives w.r.t. z, included by the iterative algorithm, may affect the energy flux as well as the mode structure in the nonlinear phase, while in the linear and quasi-linear phase these effects are negligible. Some other benchmark results of nonlinear ITG turbulence for the new version of the NLT code can be found in [39].

In figure 13, time evolution of the particle number $\delta N/N_{\rm eq}$ in ITG turbulence simulations is plotted. Here, the perturbed particle number is given by

$$\delta N = \int \delta n \mathcal{J} {\rm d}x{\rm d}y{\rm d}z, \tag{43}$$

with $\delta n = \int \delta F \frac{B^*_{0\parallel}}{m_{\rm i}} {\rm d}\xi {\rm d}\mu {\rm d}v_\parallel$. $N_{\rm eq} = \int N(x)\mathcal{J}{\rm d}x{\rm d}y{\rm d}z$ is the equilibrium particle number. It can be seen that the particle number conservation is comparable to other gyrokinetic codes, such as the results shown in [24].

4.3. Computational efficiency

In this subsection, the computational efficiency of the new version of the NLT code is compared with the previous version by running single ITG mode simulations with $k_\theta \rho_{\rm i} = 0.3$. The numbers of grid points used for this test are $(N_\psi, N_\zeta, N_{v_\parallel}, N_\mu) = (160, 32, 64, 32)$ and $(N_x, N_y, N_{v_\parallel}, N_\mu) = (160, 32, 64, 32)$ for the previous version and the new version, which are required for numerical convergence in this test. Figure 14 shows the numerical convergence rate in the number of grid points of $N_\theta(N_z)$ for the previous (new) version of the code. It can be seen that a minimal number $N_\theta = 128$ or $N_z = 16$ is required to get both the convergent frequency and growth rate for each version. Table 1 compares the computational cost of the new and previous version of the NLT code. It is clear that in this case of high-n ITG mode, the computational efficiency of the NLT code has been greatly improved for a given accuracy by using field-aligned coordinates.

5. Summary

In this work, we have implemented field-aligned coordinates in the semi-Lagrangian code NLT to improve the computational efficiency for the gyrokinetic turbulence simulation. By introducing field-aligned coordinates, the computational cost can be greatly reduced by using a small number of grid points along the field line to resolve perturbations with small $k_\parallel$. For ITG turbulence simulation, the grid number of z is typically set as 16 or 32. The gyrokinetic Vlasov equation is solved by using the backward semi-Lagrangian method with 4D B-spline interpolation by tensor product in the field-aligned coordinates. Two buffer regions are imposed at both ends of z grid to ensure the twisted boundary condition along the field line and keep the interpolation accuracy near the boundary. The QNE is also solved with the field-aligned coordinates. The partial derivatives w.r.t. z, which are usually dropped in many codes, can be preserved by applying the iteration method. A new transform method is

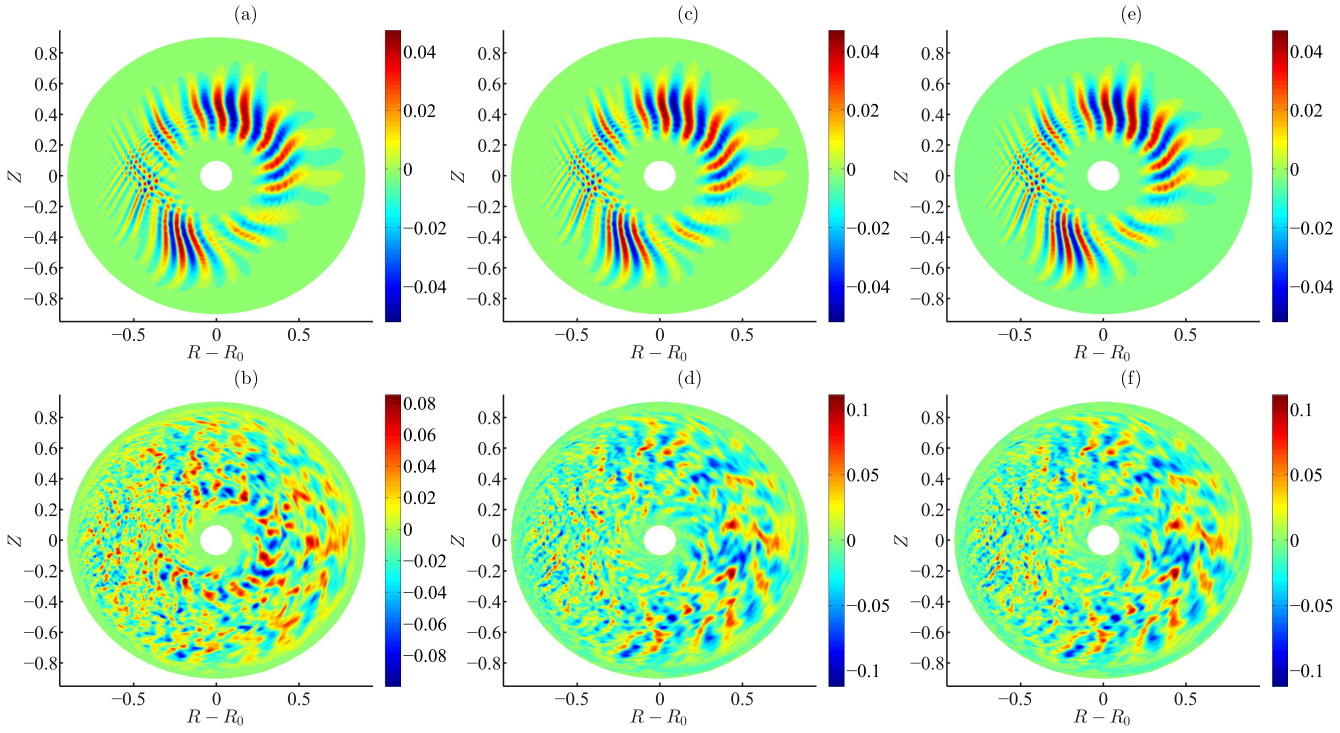

Figure 12. Mode structure of $\delta\phi$ in linear phase $t = 42c_s/R$ (up) and nonlinear phase $t = 80c_s/R$ (down) with different number of iterations: $n_{\rm it} = 0$ (left), $n_{\rm it} = 5$ (middle) and $n_{\rm it} = 10$ right.

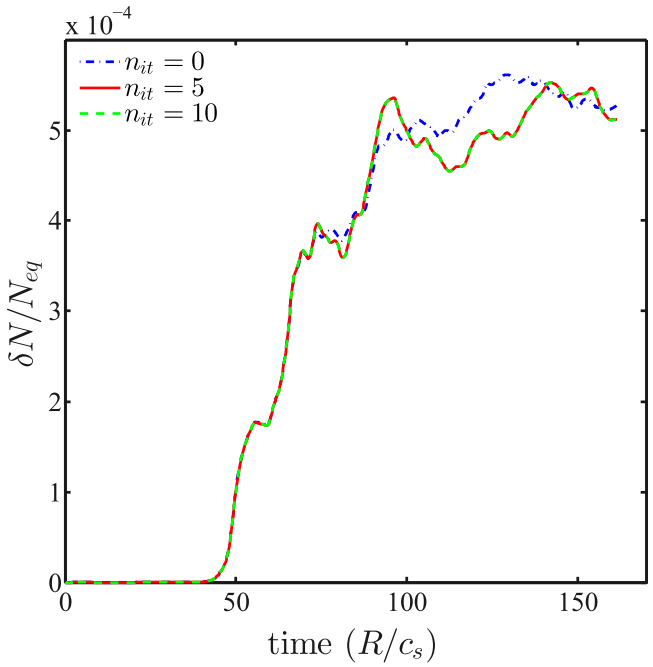

Figure 13. Time evolution of the particle number ($\delta N/N_{\rm eq}$) in the ITG turbulence simulation.

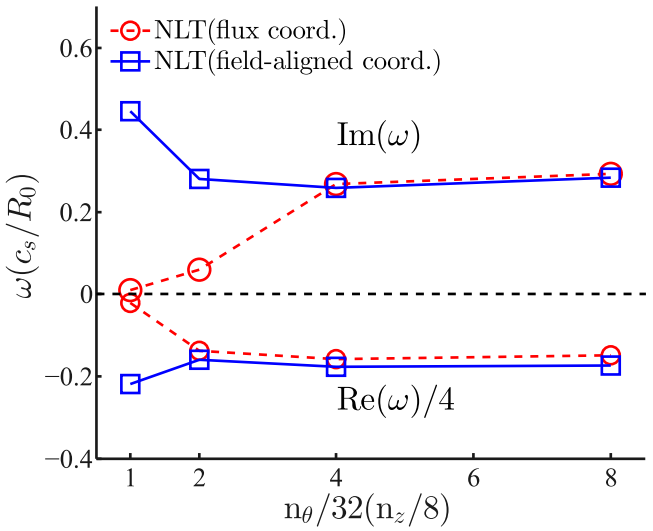

Figure 14. Convergence rate in the number of grid points of N_θ (for the previous version) and N_z (for the new version) for single ITG mode simulation with $k_\theta\rho_{\rm i} = 0.3$.

Table 1. Computational cost for single ITG mode simulations by the NLT code with different grid numbers of θ and z. 384 CPUs are employed in each case.

	Grid number of $\theta(z)$	Computational cost
previous version	$N_\theta = 128\ (k_\theta \rho_\mathrm{i} < 0.3)$	7.36 s/step
previous version	$N_\theta = 256\ (k_\theta \rho_\mathrm{i} \geqslant 0.3)$	15.44 s/step
new version	$N_z = 16$	0.60 s/step
new version	$N_z = 32$	1.120 s/step

proposed for the numerical integration of the gyro-average operator.

Acknowledgments

This work is supported by National Natural Science Foundation of China under Grant Nos. 11505240, 11375196 and 11405174, and the National ITER program of China under Contract No. 2014GB113000.

Appendix. Coefficients in QNE

Let $\beta(x, z) = e_\mathrm{i} n_{0\mathrm{i}}/B\omega_\mathrm{i}$; the coefficients in equations (34) and (33) can be given as

$$
\begin{aligned}
c_1 &= \beta g^{xx},\\
c_2 &= \beta g^{yy},\\
c_3 &= 2\beta g^{xy},\\
c_4 &= \frac{1}{J}\partial_x(J\beta g^{xx}) + \frac{1}{J}\partial_z(\beta J g^{zx}),\\
c_5 &= \frac{1}{J}\partial_x(J\beta g^{xy}) + \frac{1}{J}\partial_z(\beta J g^{zy}),\\
c_6 &= -\frac{e^2 n_{0\mathrm{e}}}{T_\mathrm{e}},\\
s &= -e_\mathrm{i}\delta n_\mathrm{i}^\mathrm{g} - \frac{e^2 n_{0\mathrm{e}}\langle\delta\phi\rangle}{T_\mathrm{e}}
\end{aligned}
$$

and

$$
\begin{aligned}
l_1 &= 2\beta g^{xz},\\
l_2 &= 2\beta g^{zy},\\
l_3 &= \beta\left(g^{zz} - \frac{1}{\mathcal{J}^2 B^2}\right),\\
l_4 &= \frac{1}{J}\partial_x(J\beta g^{xz}) + \frac{1}{J}\partial_z\left[\beta\left(Jg^{zz} - \frac{1}{JB^2}\right)\right].
\end{aligned}
$$

The metric coefficients $g^{\alpha\beta}$ are defined by

$$g^{\alpha\beta} \triangleq \nabla\alpha\cdot\nabla\beta.$$

References

[1] Garbet X, Idomura Y, Villard L and Watanabe T 2010 *Nucl. Fusion* **50** 043002
[2] Batchelor D *et al* 2007 *Plasma Sci. Technol.* **9** 312
[3] Lin Z, Tang W and Lee W 1995 *Phys. Plasmas* **2** 2975
[4] Watanabe T H, Sugama H and Ferrando-Margalet S 2008 *Phys. Rev. Lett.* **100** 195002
[5] Chen Y, Parker S E, Lang J and Fu G Y 2010 *Phys. Plasmas* **17** 102504
[6] Zhang W, Holod I, Lin Z and Xiao Y 2012 *Phys. Plasmas* **19** 022507
[7] Horton W 1999 *Rev. Mod. Phys.* **71** 735
[8] Beer M A, Cowley S and Hammett G 1995 *Phys. Plasmas* **2** 2687
[9] Chen Y and Parker S E 2003 *J. Comput. Phys.* **189** 463
[10] Görler T, Lapillonne X, Brunner S, Dannert T, Jenko F, Merz F and Told D 2011 *J. Comput. Phys.* **230** 7053
[11] Hariri F and Ottaviani M 2013 *Comput. Phys. Commun.* **184** 2419
[12] Ye L, Xu Y, Xiao X, Dai Z and Wang S 2016 *J. Comput. Phys.* **316** 180
[13] Xiao X, Ye L, Xu Y and Wang S 2017 *Commun. Comput. Phys.* **22** 789
[14] Strang G 1968 *SIAM J. Numer. Anal.* **5** 506
[15] Ye L, Guo W, Xiao X and Wang S 2013 *Phys. Plasmas* **20** 072501
[16] White R B 2001 *The Theory of Toroidally Confined Plasmas* (Singapore: World Scientific)
[17] Scott B 2001 *Phys. Plasmas* **8** 447
[18] Ottaviani M 2011 *Phys. Lett.* A **375** 1677
[19] Wang S 2012 *Phys. Plasmas* **19** 062504
[20] Wang S 2013 *Phys. Rev.* E **87** 063103
[21] Wang S 2014 *Phys. Plasmas* **21** 072312
[22] Wang S 2001 *Phys. Rev.* E **64** 056404
[23] Idomura Y, Tokuda S and Kishimoto Y 2003 *Nucl. Fusion* **43** 234
[24] Idomura Y, Ida M, Kano T, Aiba N and Tokuda S 2008 *Comput. Phys. Commun.* **179** 391
[25] Cheng C Z and Knorr G 1976 *J. Comput. Phys.* **22** 330
[26] Grandgirard V *et al* 2006 *J. Comput. Phys.* **217** 395
[27] Kwon J M, Yi D, Piao X and Kim P 2015 *J. Comput. Phys.* **283** 518
[28] Grandgirard V *et al* 2016 *Comput. Phys. Commun.* **207** 35
[29] Wang Z, Li J, Dong J and Kishimoto Y 2009 *Phys. Rev. Lett.* **103** 015004
[30] Crouseilles N, Mehrenberger M and Sellama H 2010 *Commun. Comput. Phys.* **8** 484
[31] Candy J and Waltz R 2003 *J. Comput. Phys.* **186** 545
[32] Lee W 1987 *J. Comput. Phys.* **72** 243
[33] Idomura Y, Wakatani M and Tokuda S 2000 *Phys. Plasmas* **7** 3551
[34] Kim C C and Parker S E 2000 *Massively Parallel Three-Dimensional Toroidal Gyrokinetic Flux-Tube Turbulence Simulation* (Academic)
[35] Parker S E 2002 *J. Comput. Phys.* **178** 520
[36] Atkinson K E 2008 *An Introduction to Numerical Analysis* (New York: Wiley)
[37] Dimits A *et al* 2000 *Phys. Plasmas* **7** 969
[38] Falchetto G *et al* 2008 *Plasma Phys. Control. Fusion* **50** 124015
[39] Xu Y, Ye L, Dai Z, Xiao X and Wang S 2017 *Phys. Plasmas* **24** 082515