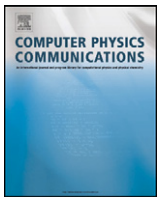
Computer Physics Communications 180 (2009) 2650–2672

Contents lists available at ScienceDirect

Computer Physics Communications

www.elsevier.com/locate/cpc



40th Anniversary Issue

The nonlinear gyro-kinetic flux tube code GKW ☆

A.G. Peeters [a,*], Y. Camenen [a], F.J. Casson [a], W.A. Hornsby [a], A.P. Snodin [a], D. Strintzi [b], G. Szepesi [a]

[a] Centre for Fusion, Space and Astrophysics, Physics Department, University of Warwick, CV4 7AL, Coventry, UK
[b] Department of Electrical and Computer Engineering, National Technical University of Athens, Association Euratom, Hellenic Republic, GR-157 73 Athens, Greece

ARTICLE INFO

Article history:
Received 1 April 2009
Received in revised form 2 July 2009
Accepted 2 July 2009
Available online 3 July 2009

PACS:
52.25.Fi
52.25.Xz
52.30.Gz
52.35.Qz
52.55.Fa
52.65.Tt

Keywords:
Gyro-kinetic
Flux tube
Drift wave
Tokamak
Plasma turbulence

ABSTRACT

A new nonlinear gyro-kinetic flux tube code (GKW) for the simulation of micro instabilities and turbulence in magnetic confinement plasmas is presented in this paper. The code incorporates all physics effects that can be expected from a state of the art gyro-kinetic simulation code in the local limit: kinetic electrons, electromagnetic effects, collisions, full general geometry with a coupling to a MHD equilibrium code, and $E \times B$ shearing. In addition the physics of plasma rotation has been implemented through a formulation of the gyro-kinetic equation in the co-moving system. The gyro-kinetic model is five-dimensional and requires a massive parallel approach. GKW has been parallelised using MPI and scales well up to 8192+ cores. The paper presents the set of equations solved, the numerical methods, the code structure, and the essential benchmarks.

Program summary

Program title: GKW
Catalogue identifier: AEES_v1_0
Program summary URL: http://cpc.cs.qub.ac.uk/summaries/AEES_v1_0.html
Program obtainable from: CPC Program Library, Queen's University, Belfast, N. Ireland
Licensing provisions: GNU GPL v3
No. of lines in distributed program, including test data, etc.: 29 998
No. of bytes in distributed program, including test data, etc.: 206 943
Distribution format: tar.gz
Programming language: Fortran 95
Computer: Not computer specific
Operating system: Any for which a Fortran 95 compiler is available
Has the code been vectorised or parallelised?: Yes. The program can efficiently utilise 8192+ processors, depending on problem and available computer. 128 processors is reasonable for a typical nonlinear kinetic run on the latest x86-64 machines.
RAM: ∼128 MB–1 GB for a linear run; 25 GB for typical nonlinear kinetic run (30 million grid points)
Classification: 19.8, 19.9, 19.11
External routines: None required, although the functionality of the program is somewhat limited without a MPI implementation (preferably MPI-2) and the FFTW3 library.
Nature of problem: Five-dimensional gyro-kinetic Vlasov equation in general flux tube tokamak geometry with kinetic electrons, electro-magnetic effects and collisions
Solution method: Pseudo-spectral and finite difference with explicit time integration
Additional comments: The MHD equilibrium code CHEASE [1] is used for the general geometry calculations. This code has been developed in CRPP Lausanne and is not distributed together with GKW, but can be downloaded separately. The geometry module of GKW is based on the version 7.1 of CHEASE, which includes the output for Hamada coordinates.
Running time: (On recent x86-64 hardware) ∼10 minutes for a short linear problem; 48 hours for typical nonlinear kinetic run.

☆ This paper and its associated computer program are available via the Computer Physics Communications homepage on ScienceDirect (http://www.sciencedirect.com/science/journal/00104655).

* Corresponding author.
E-mail address: A.G.Peeters@warwick.ac.uk (A.G. Peeters).

0010-4655/$ – see front matter
doi:10.1016/j.cpc.2009.07.001

References:

[1] H. Lütjens, A. Bondeson, O. Sauter, Comput. Phys. Comm. 97 (1996) 219, http://cpc.cs.qub.ac.uk/summaries/ADDH_v1_0.html.

1. Introduction

In this paper a new nonlinear gyro-kinetic flux tube code GKW (Gyro-Kinetics@Warwick) is presented with the aim of documenting the exact equations solved, presenting the essential benchmarks which document the proper numerical solution, and explaining the code structure and installation. This code was built through the extension of the linear code LINART [1] and can be used both in the linear as well as in the nonlinear regime.

GKW aims at solving the turbulent transport problem arising in tokamak plasmas. It solves the gyro-kinetic (the kinetic equation averaged over the fast gyro-motion) equation on a fixed grid in the 5-dimensional space using a combination of finite difference and pseudo spectral methods. The solution of the kinetic equations governing this problem is an active area of research, with many groups worldwide contributing. The two approaches used are Particle in Cell (PIC) codes [2–8], and Vlasov codes [9–14]. GKW obviously falls in the latter category, and reaches a standard that can be expected from a modern Vlasov code, i.e. it includes kinetic electrons, a general description of the geometry with a coupling to a MHD equilibrium solver, electro-magnetic effects, collisions, and the effect of $E \times B$ shearing. In addition the physics of toroidal plasma rotation is included through a formulation of the equations in the co-moving system. The physics of rotation is currently an active area of research and most of the applications of GKW up to now have concentrated on the transport of toroidal momentum [15–22] or the influence of rotation on other transport channels [23].

GKW is freely distributed with minimal restrictions to the users. Essentially you can make any changes to the source, and are not obliged to inform us or communicate these changes back to us. (Although we would welcome it if you did.) Furthermore, you can freely distribute the code to third parties as long as you communicate to them the restrictions formulated below. The only restriction that we impose is that the effort made by us is recognised. You can therefore never change the name of the code without explicit expression of the lead author (AGP) no matter how many changes you make. Also in any publication that uses the results of GKW, either directly or indirectly, you are obliged to cite this paper. (If you want to be kind to us you could additionally cite the original LINART paper [1] and/or the first paper in which GKW has been used Ref. [17].)

The remainder of this paper is structured as follows. Section 2 discusses the gyro-kinetic framework, and introduces the main equations, Section 3 then gives the normalisation, and Section 4 discusses the geometry. The spectral representation is described in Section 5 and the complete set of equations is then given in Section 6. Section 7 describes the collision operator and Sections 8–13 give technical details about numerical implementation and running the code. Section 14 presents the benchmarks which document the proper solution of the equations, and conclusions are drawn in Section 15.

2. The gyro-kinetic framework

The starting point of the equation for the evolution of the distribution function is the gyro-kinetic theory formulated in Refs. [24–27]. In GKW these equations are solved up to first order in the Larmor radius over the major radius $\rho_* \ll 1$ (with the exception of the polarisation which enters in the field equations). In order to determine which of the terms in the original formulation of the theory (which is accurate up to second order in ρ_*) must be kept, we will here briefly discuss the ordering. More details can be found in Ref. [19].

A thermal velocity (v_{th}) and a major radius (R) are used to normalise the length and time scales. For the gradients we assume

$$R\nabla_{\parallel} \approx 1, \qquad R\nabla_{\perp} \approx 1/\rho_*, \tag{1}$$

i.e. the solution is assumed to have a length scale of the order of the system size along the field, but can have a length scale of the order of the Larmor radius perpendicular to the magnetic field. This difference in length scales means that the convecting velocities must be evaluated to a different order parallel and perpendicular to the field. The velocity along the field is to be evaluated in lowest order only (i.e. of the order of v_{th}), but the perpendicular velocity must be evaluated up to first order (order $\rho_* v_{\mathrm{th}}$). In GKW the perturbations in the electro-static potential ϕ as well as the vector potential $\mathbf{A}$ are retained. For the latter it is, however, assumed that

$$\mathbf{A} = A_{\parallel}\mathbf{b}, \tag{2}$$

where $\mathbf{b}$ is the unit vector in the direction of the unperturbed magnetic field. The assumption above means that no compressional effects of the magnetic field are retained in the description. Compressional effects are currently implemented in the code but are not part of the present release. Note that the vector potential above is the perturbed vector potential not to be confused with the one connected with the equilibrium magnetic field. The equations will be formulated in the co-moving system of a toroidally rotating plasma. The $E \times B$ drift velocity connected with this toroidal rotation is formally of the order of the thermal velocity, but the transformation to the co-moving frame removes this large $E \times B$ velocity from the equations [19]. The perturbed potential and vector potential in the co-moving frame can then be taken to be of the order

$$\phi \approx \frac{T}{e}\rho_*, \qquad A_{\parallel} \approx \frac{T}{e v_{\mathrm{th}}}\rho_*, \tag{3}$$

where T is the temperature, and e is the elementary charge. From this assumption it can be derived that the $E \times B$ velocity is of the order $\rho_* v_{\mathrm{th}}$.

Gyro-centre coordinates ($\mathbf{X}, \mu, v_{\parallel}$) are used, with $\mathbf{X}$ being the gyro-centre position, $v_{\parallel}$ the parallel (to the magnetic field) velocity, and μ the magnetic moment $\mu = m v_{\perp}^2/2B$. Here $v_{\perp}$ is the velocity component perpendicular to the equilibrium magnetic field $\mathbf{B}$, m is the particle mass and B is the magnetic field strength. The magnetic moment is an invariant of motion, but the parallel velocity changes

due to the electric field acceleration and mirror force. It turns out that for consistency one needs to keep both the lowest as well as the first-order (in ρ_*) contribution to the acceleration.

Finally, the approximation known as the δf approximation is employed. This approximation assumes that the perturbed distribution f is much smaller than the background distribution F, but has a perpendicular length scale of the order of the Larmor radius whereas the background varies on the length scale of the system size:

$$f \approx \rho_* F, \qquad \frac{\partial f}{\partial v_\parallel} \approx \rho_* \frac{\partial F}{\partial v_\parallel}, \qquad \nabla f \approx \nabla F. \tag{4}$$

The ordering assumption above removes the 'velocity nonlinearity' (combinations of the electro-magnetic fields and the velocity space derivative of the perturbed distribution) from the evolution equation of the perturbed distribution. The disadvantage is that strict energy conservation no longer applies.

With the help of these approximations the gyro-kinetic equation of Refs. [24,26,27] can be rewritten. The evolution of equation for the gyro-centre position $\mathbf{X}$, and the parallel velocity are [19]

$$\frac{\mathrm{d}\mathbf{X}}{\mathrm{d}t} = v_\parallel \mathbf{b} + \mathbf{v}_D + \mathbf{v}_\chi, \qquad m v_\parallel \frac{\mathrm{d}v_\parallel}{\mathrm{d}t} = Ze \frac{\mathrm{d}\mathbf{X}}{\mathrm{d}t} \cdot \mathbf{E} - \mu \frac{\mathrm{d}\mathbf{X}}{\mathrm{d}t} \cdot \nabla B, \qquad \frac{\mathrm{d}\mu}{\mathrm{d}t} = 0. \tag{5}$$

Here Z is the particle charge, and $\mathbf{E}$ is the gyro-averaged electric field

$$\mathbf{E} = -\nabla \langle \phi \rangle - \frac{\partial \langle A_\parallel \rangle}{\partial t} \mathbf{b}, \tag{6}$$

where the angle brackets denote the gyro-average. The velocities in Eq. (5) are from left to right: the parallel motion along the unperturbed field ($v_\parallel \mathbf{b}$), the drift motion due to the inhomogeneous field ($\mathbf{v}_D$), and

$$\mathbf{v}_\chi = \frac{\mathbf{b} \times \nabla \chi}{B} \quad \text{with } \chi = \langle \phi \rangle - v_\parallel \langle A_\parallel \rangle, \tag{7}$$

which is the combination of the $E \times B$ velocity ($\mathbf{v}_E = \mathbf{b} \times \nabla \langle \phi \rangle / B$) and the parallel motion along the perturbed field line ($\mathbf{v}_{\delta B} = -\mathbf{b} \times \nabla v_\parallel \langle A_\parallel \rangle / B$). The drift due to the inhomogeneous magnetic field can be written in the form

$$\mathbf{v}_D = \frac{1}{Ze} \left[\frac{m v_\parallel^2}{B} + \mu \right] \frac{\mathbf{B} \times \nabla B}{B^2} + \frac{m v_\parallel^2}{2ZeB} \beta' \mathbf{b} \times \nabla \psi + \frac{2 m v_\parallel}{ZeB} \mathbf{\Omega}_\perp. \tag{8}$$

In the equation above ψ is a radial coordinate (flux label) and $\mathbf{\Omega}_\perp$ is the angular (toroidal) rotation vector perpendicular to the field, i.e. $\mathbf{\Omega}_\perp = \mathbf{\Omega} - (\mathbf{\Omega} \cdot \mathbf{b})\mathbf{b}$. The first term on the right is the combination of the grad-B drift and curvature drift in the low beta approximation, whereas the second term, involving

$$\beta' = \frac{2\mu_0}{B^2} \frac{\partial p}{\partial \psi}, \tag{9}$$

is the correction to the curvature drift due to the modification of the equilibrium associated with the pressure gradient. The last term is the Coriolis drift derived in Ref. [17] using the formulation of the gyro-kinetic equations in the co-moving frame [29]. The formulation of Ref. [17] was extended in Ref. [19] to include the centrifugal force and its implementation in GKW is under way but not part of the present release.

From the equations above the gyro-kinetic equation in ($\mathbf{X}$, $v_\parallel$, μ) coordinates can be derived [24]:

$$\frac{\partial f_{\mathrm{tot}}}{\partial t} + \frac{\mathrm{d}\mathbf{X}}{\mathrm{d}t} \cdot \nabla_\mu f_{\mathrm{tot}} + \frac{\mathrm{d}v_\parallel}{\mathrm{d}t} \frac{\partial f_{\mathrm{tot}}}{\partial v_\parallel} = 0. \tag{10}$$

The distribution function f_{tot} in this equation is split in a background F and a perturbed distribution f. As discussed at the beginning of the section the δf approximation will be employed. Consequently, the equation for f can be written in the form

$$\frac{\partial f}{\partial t} + (v_\parallel \mathbf{b} + \mathbf{v}_D + \mathbf{v}_\chi) \cdot \nabla f - \frac{\mu B}{m} \frac{\mathbf{B} \cdot \nabla B}{B^2} \frac{\partial f}{\partial v_\parallel} = S, \tag{11}$$

where S is determined by the background distribution function. The latter is assumed to be a Maxwellian (F_M), with density (n), temperature (T) and mean parallel velocity ($u_\parallel$) being functions of the radial coordinate only:

$$F = F_M = \frac{n}{\pi^{3/2} v_{\mathrm{th}}^3} \exp\left[-\frac{(v_\parallel - u_\parallel)^2 + 2\mu B/m}{v_{\mathrm{th}}^2} \right]. \tag{12}$$

Here $v_{\mathrm{th}} \equiv \sqrt{2T/m}$ is the thermal velocity (note that in this area of research the thermal velocity is usually defined without the $\sqrt{2}$ factor). The appearance of an explicit rotation velocity in the Maxwellian may be confusing since the equations are formulated in the co-moving system in which the rotation vanishes. However, under experimental conditions the plasma rotation has a radial gradient while the frame is chosen to rotate as a rigid body with a constant angular frequency Ω. Indeed a differential rotation of the frame would be impractical since the distance between two fixed points in the co-moving system would increase in time, i.e. the metric would be time dependent. The local model constructed here chooses the rotation of the frame to be equal to the plasma rotation at one particular radius. Therefore $u_\parallel$ in the above Maxwellian is zero at the radial location considered, but has a finite radial gradient. This gradient will enter the equations as $\nabla \Omega$ (for more details see Ref. [19]).

The term containing the derivative of the vector potential towards time presents a numerical difficulty. In the δf formalism it can, however, easily be combined with the time derivative of f. If one defines a 'new' distribution g

$$g = f + \frac{Ze}{T} v_\parallel \langle A_\parallel \rangle F_M, \tag{13}$$

and replaces the time derivative of f in the gyro-kinetic equation with the time derivative of g, the cumbersome time derivative of $\langle A_\parallel \rangle$ is removed. Using the equations above one arrives at the following equation for the distribution function g

$$\frac{\partial g}{\partial t} + \mathbf{v}_\chi \cdot \nabla g + (v_\parallel \mathbf{b} + \mathbf{v}_D) \cdot \nabla f - \frac{\mu B}{m} \frac{\mathbf{B} \cdot \nabla B}{B^2} \frac{\partial f}{\partial v_\parallel} = S, \tag{14}$$

where S is given by

$$S = -(\mathbf{v}_\chi + \mathbf{v}_D) \cdot \left[\frac{\nabla n}{n} + \left(\frac{v_\parallel^2}{v_{\text{th}}^2} + \frac{\mu B}{T} - \frac{3}{2} \right) \frac{\nabla T}{T} + \frac{m v_\parallel}{T} \frac{R B_t}{B} \nabla \Omega \right] F_M - \frac{Ze}{T} [v_\parallel \mathbf{b} + \mathbf{v}_D] \cdot \nabla \langle \phi \rangle F_M. \tag{15}$$

Note that both g and f appear in this equation. Since any form of dissipation on the field variables leads to a numerical instability [13], it is preferable to use f in any term that requires dissipation for stable numerical implementation. It should also be noted here that the Maxwellian is not an exact solution of the gyro-kinetic equation in the limit of a zero perturbed electro-magnetic field. From the equation above it follows that S is nonzero due to the drifts connected with the magnetic field inhomogeneity in the gradients of density temperature and velocity. The solution of the evolution equation will determine the deviation away from the Maxwellian as a nonzero perturbed distribution f. This perturbation is of the order $\rho_* F_M$, and neglecting it as part of the background distribution is consistent with the ordering adopted. The drift term is $\mathbf{v}_D$ in the first term of the equation above is nevertheless implemented in GKW since the perturbation it generates is responsible for the neo-classical transport. All physics and diagnostic of the neo-classical fluxes have been implemented in the code, but the implementation has not been benchmarked and should therefore not be used.

To close the set of equations, one needs to calculate the potential and vector potential through the (low frequency ordering) Maxwell equations. These have been formulated in the literature, and will be given in Section 6 when the full set of equations in normalised form is discussed.

3. Normalisations

Of course, all quantities in the code are normalised. Below, this normalisation is discussed in detail. The velocity will be normalised by the thermal velocity of each of the species. This has the advantage that the grid in velocity space can always be defined relative to the thermal velocity. Quantities like the potential, however, must be normalised by a reference temperature since it is species independent. A similar argument applies to the vector potential and the time.

We define a reference mass m_{ref}, a reference thermal velocity v_{thref}, a reference density n_{ref}, a reference temperature T_{ref}, a reference magnetic field B_{ref} evaluated on the magnetic axis, and a reference major radius R_{ref}. These are not unrelated since

$$T_{\text{ref}} = \frac{1}{2} m_{\text{ref}} v_{\text{thref}}^2, \qquad \rho_{\text{ref}} = \frac{m_{\text{ref}} v_{\text{thref}}}{e B_{\text{ref}}}. \tag{16}$$

Furthermore we define a normalised Larmor radius to be

$$\rho_* = \rho_{\text{ref}} / R_{\text{ref}}. \tag{17}$$

The reference values are used to define, for each species, a dimensionless mass m_R, a dimensionless thermal velocity v_R, a dimensionless density n_R, and a dimensionless temperature T_R

$$m_R = \frac{m}{m_{\text{ref}}}, \qquad v_R = \frac{v_{\text{th}}}{v_{\text{thref}}}, \qquad n_R = \frac{n}{n_{\text{ref}}}, \qquad T_R = \frac{T}{T_{\text{ref}}}. \tag{18}$$

The fields are made dimensionless using only reference values

$$\phi = \rho_* \frac{T_{\text{ref}}}{e} \phi_N, \qquad A_\parallel = B_{\text{ref}} R_{\text{ref}} \rho_*^2 A_{\parallel N}, \qquad \chi = \rho_* \frac{T_{\text{ref}}}{e} \chi_N. \tag{19}$$

Here the index N refers to a normalised quantity. Note that factors of ρ_* have been added in the definitions of the normalised fields. These factors are chosen such that the normalised quantities are of the order 1.

Also time t, magnetic field B, angular rotation frequency Ω, and the major radius R, are made dimensionless using the reference values

$$\begin{aligned} t &= R_{\text{ref}} t_N / v_{\text{thref}}, & B &= B_{\text{ref}} B_N, \\ \Omega &= v_{\text{thref}} \Omega_N / R_{\text{ref}}, & R &= R_{\text{ref}} R_N. \end{aligned} \tag{20}$$

The velocity space coordinates, however, are made dimensionless with the help of the thermal velocity of the species

$$v_\parallel = v_{\parallel N} v_{\text{th}}, \qquad \mu = \frac{m v_{\text{th}}^2}{B_{\text{ref}}} \mu_N. \tag{21}$$

It is then convenient to normalise the distribution functions according to

$$f = \rho_* \frac{n}{v_{\text{th}}^3} f_N, \qquad F_M = \frac{n}{v_{\text{th}}^3} F_{MN}. \tag{22}$$

The gradient of density and temperature are made dimensionless using the density and temperature of the species under consideration, but the plasma rotation is largely a bulk motion of all the species together and its gradient is normalised using the reference thermal velocity v_{thref}

$$\frac{R}{L_N} = -\frac{1}{n}\frac{\partial n}{\partial \psi}, \qquad \frac{R}{L_T} = -\frac{1}{T}\frac{\partial T}{\partial \psi}, \qquad u'_N = -\frac{R_{\text{ref}}}{v_{\text{thref}}}\frac{\partial \Omega}{\partial \psi}. \tag{23}$$

Note that also the radial coordinate ψ is normalised:

$$\psi = \frac{R_{\max} - R_{\min}}{2R_{\text{ref}}}, \tag{24}$$

where $R_{\max}$ ($R_{\min}$) is the maximum (minimum) major radius of the flux surface. For circular surfaces, for instance, $\psi = r/R_{\text{ref}} = \epsilon$, where r is the minor radius. All gradients are normalised using the major radius R_{ref}, i.e.

$$\nabla = \frac{1}{R_{\text{ref}}}\nabla_N. \tag{25}$$

The poloidal flux Ψ used in the derivation of the metric tensors is normalised according to

$$\Psi = R_{\text{ref}}^2 B_{\text{ref}} \Psi_N. \tag{26}$$

The strength of the electromagnetic effects is determined by the plasma beta (β). Although β is dimensionless we define a different dimensionless beta (β_N) for the use in the code. This β_N is directly related to the reference values

$$\beta_N = \frac{n_{\text{ref}} T_{\text{ref}}}{B_{\text{ref}}^2/2\mu_0}. \tag{27}$$

Finally, the wave vectors introduced in the spectral representation arise from the perpendicular gradient of a fluctuating quantity and will therefore be normalised to ρ_{ref}:

$$k = \frac{k_N}{\rho_{\text{ref}}}. \tag{28}$$

In all the following sections we will, of course, use only the normalised quantities. The index N is then dropped for convenience.

4. Geometry

GKW is formulated in field aligned Hamada coordinates [30], i.e. the contravariant components of both the poloidal B^s and toroidal magnetic field B^γ are flux functions. The safety factor is then determined by the ratio $q = B^\gamma/B^s$. Starting from an orthogonal coordinate system (ψ, θ, φ), where ψ is the radial coordinate (i.e. $\mathbf{B}\cdot\nabla\psi = 0$), θ is the poloidal angle (upward on the outboard midplane), and φ is the toroidal angle (clockwise when viewed from above), and using the transformations

$$s = s(\psi, \theta), \qquad \gamma = \gamma(\psi, \theta, \varphi), \tag{29}$$

one can derive [31,32]

$$s(\theta, \psi) = \int_0^\theta \frac{\mathrm{d}\theta'}{\mathbf{B}\cdot\nabla\theta'} \Big/ \oint \frac{\mathrm{d}\theta'}{\mathbf{B}\cdot\nabla\theta'}, \tag{30}$$

$$\gamma = \frac{\varphi}{2\pi} + s_{\text{B}}\frac{RB_t}{2\pi}\int_0^\theta \frac{\mathrm{d}\theta'}{\mathbf{B}\cdot\nabla\theta'}\left[\left\{\frac{1}{R^2}\right\} - \frac{1}{R^2}\right], \tag{31}$$

with

$$B^s = 1 \Big/ \oint \frac{\mathrm{d}\theta'}{\mathbf{B}\cdot\nabla\theta'}, \qquad B^\gamma = s_{\text{B}}\frac{RB_t}{2\pi}\left\{\frac{1}{R^2}\right\}. \tag{32}$$

In the equations above, the magnetic field is decomposed as

$$\mathbf{B} = s_{\text{B}} R B_t \nabla\varphi + s_{\text{j}} \nabla\varphi \times \nabla\Psi, \tag{33}$$

where $B_t > 0$ is the toroidal component of the magnetic field, $s_{\text{B}} = \pm 1$ and $s_{\text{j}} = \pm 1$ the sign of the magnetic field and plasma current (positive in the direction of $\nabla\varphi$) and Ψ the normalised poloidal flux ($\nabla\Psi$ points from the magnetic axis to the plasma edge). The brackets $\{\}$ denote the flux surface average, which in the transformed coordinates can be written as

$$\{g\} = \oint g\,\mathrm{d}s. \tag{34}$$

The normalising constants have been chosen such that the domain $[-1/2, 1/2]$ in s corresponds to one poloidal turn and the domain $[-1/2, 1/2]$ in γ corresponds to one toroidal turn. Note that the transformations of Eq. (29) leave the angle γ to be an ignorable coordinate, since any quantity that is independent of φ will also be independent of γ after the transformation.

The coordinates given above are transformed using a simple linear transformation

$$\zeta = qs - \gamma, \quad \text{where } q = \frac{B^\gamma}{B^s} \tag{35}$$

to make them field aligned, i.e. $\mathbf{B}\cdot\nabla = B^s\partial/\partial s$ as $B^\zeta = 0$ and $B^\psi = 0$. Note that the coordinate transformation above flips the sign of the toroidal angle. The right-handed coordinate system can therefore be defined as (ψ, ζ, s). The Jacobian of the new coordinate system can be expressed in terms of the original Jacobian through

$$J_{\psi\zeta s} = 2\pi \mathbf{B}\cdot\nabla\theta \oint \frac{\mathrm{d}\theta'}{\mathbf{B}\cdot\nabla\theta'} J_{\psi\theta\varphi}. \tag{36}$$

We note here that the coordinates ψ, s and ζ are chosen dimensionless with ψ being a normalised 'minor radius' of the flux surface, see Eq. (24). We shall refer to ζ as the 'bi-normal' coordinate since it is neither strictly toroidal or poloidal.

Close inspection of the gyro-kinetic equation (14) shows that it contains parallel derivatives

$$\mathbf{b}\cdot\nabla = \mathcal{F}\frac{\partial}{\partial s} \to \mathcal{F} = \frac{B^s}{B}, \tag{37}$$

and perpendicular drifts, with the latter all involving a cross product with the magnetic field. It is then convenient to define the tensor

$$\mathcal{E}^{\alpha\beta} = \frac{1}{2B}(\nabla x_\alpha \times \nabla x_\beta)\cdot\mathbf{b}. \tag{38}$$

The convection connected with the velocity $\mathbf{v}_\chi$ can be directly expressed in this tensor times the derivatives of the potential towards x_α and the distribution towards x_β. It is worth noting that the magnetic field can be expressed as

$$\mathbf{B} = 2\pi s_\mathrm{j}\nabla\Psi \times \nabla\zeta, \tag{39}$$

which immediately leads to

$$\mathcal{E}^{\psi\zeta} = \frac{s_\mathrm{j}}{4\pi}\frac{\partial\psi}{\partial\Psi}, \tag{40}$$

showing that $\mathcal{E}^{\psi\zeta}$ is constant on a flux surface. For convenience we define two more quantities

$$\mathcal{D}^\alpha = -2\mathcal{E}^{\alpha\beta}\frac{1}{B}\frac{\partial B}{\partial x_\beta}, \qquad \mathcal{G} = \mathcal{F}\frac{\partial \ln B}{\partial s}, \tag{41}$$

with $\mathcal{D}$ related to the grad B drift, and $\mathcal{G}$ to the trapping. Finally, the Coriolis drift will enter the equations through the one form $\mathcal{H}$

$$\mathcal{H}^\alpha = -\frac{s_\mathrm{B}}{B\Omega}\boldsymbol{\Omega}_\perp\cdot\nabla x_\alpha. \tag{42}$$

The tensor elements defined above all have similar magnitude. It must be realised though that these elements are multiplied with the derivatives towards the respective coordinate. Since the derivatives along the field are much smaller than the derivatives in the perpendicular plane, the tensor elements containing the coordinate s are neglected here (i.e. $\mathcal{D}^s = \mathcal{E}^{s\psi} = \mathcal{E}^{s\zeta} = \mathcal{H}^s = 0$).

The equations in GKW are formulated for an arbitrary toroidally symmetric geometry using the tensors defined above. For an arbitrary toroidal MHD equilibrium the tensor elements can be calculated through an interface with the Grad–Shafranov solver CHEASE [33]. The simplified '$s-\alpha$' equilibrium is directly implemented in the code through pre-defined tensor elements. The latter model equilibrium is the simplest choice of the geometry of a tokamak with the flux surfaces being circular and having a small inverse aspect ratio $\epsilon = r/R \ll 1$, with r being the minor radius of the surface and $R = R_\mathrm{ref}$. Only the lowest order in an ϵ expansion is kept in the definition of all the tensors. The coordinates can then be approximated as

$$\psi = \epsilon = \frac{r}{R}, \qquad \zeta = \frac{1}{2\pi}[q\theta - \phi], \qquad s = \frac{\theta}{2\pi}, \tag{43}$$

with the (normalised) gradients

$$\nabla\psi = \mathbf{e}_r, \qquad \nabla\zeta = \frac{q}{2\pi\epsilon}[\mathbf{e}_\theta + \hat{s}\theta\mathbf{e}_r], \qquad \nabla s = \frac{1}{2\pi\epsilon}\mathbf{e}_\theta, \tag{44}$$

where $\hat{s} = (r/q)\,\mathrm{d}q/\mathrm{d}r$ is the magnetic shear. Note that the gradient of the toroidal angle is ordered small compared with the gradient of the poloidal angle (times q) and is neglected in the gradient of ζ. The contravariant components of the magnetic field are

$$B^s = s_\mathrm{j}\frac{B_p}{2\pi r}, \qquad B^\zeta = -s_\mathrm{B}\frac{B_t}{2\pi R}, \tag{45}$$

where B_p is the poloidal magnetic field. The gradient of the magnetic field strength is assumed to be in the direction of ∇R and the drift term is approximated as

$$\mathbf{B}\times\nabla B = -s_\mathrm{B}B^2[\cos\theta\mathbf{e}_\theta + \sin\theta\mathbf{e}_r]. \tag{46}$$

Using these expressions one can evaluate the tensors to be

$$\mathcal{F} = \frac{s_\mathrm{j}}{2\pi|q|}, \qquad \mathcal{D}^\psi = -s_\mathrm{B}\sin(2\pi s), \qquad \mathcal{D}^\zeta = -\frac{s_\mathrm{j}|q|}{2\pi\epsilon}\big[\cos(2\pi s) + 2\pi\hat{s}s\sin(2\pi s)\big], \tag{47}$$

$$\mathcal{G} = \frac{s_\mathrm{j}\epsilon}{|q|}\sin(2\pi s), \qquad \mathcal{E}^{\psi\zeta} = s_\mathrm{j}\frac{|q|}{4\pi\epsilon} = -\mathcal{E}^{\zeta\psi}, \qquad \mathcal{H}^\psi = \mathcal{D}^\psi, \qquad \mathcal{H}^\zeta = \mathcal{D}^\zeta. \tag{48}$$

Finally the metric elements are

$$g^{\zeta\zeta} = \left(\frac{q}{2\pi\epsilon}\right)^2 + \left(\frac{q}{\epsilon}\hat{s}s\right)^2, \qquad g^{\zeta\psi} = \frac{s_B s_j |q|}{\epsilon}\hat{s}s, \qquad g^{\psi\psi} = 1. \tag{49}$$

All numerical results presented in this paper have been obtained with the geometry coefficients given above, except the benchmark of the full geometry treatment.

GKW uses the local limit. Central to this approximation is the small scales of the turbulent fluctuations. All background plasma parameters are assumed homogeneous across the simulation domain perpendicular to the field, i.e. there is no dependence of the parameters on the coordinates ζ and ψ. The turbulence is homogeneous in the plane perpendicular to the magnetic field and periodic boundary conditions apply. Note that the dependence of equilibrium quantities on the 'parallel' coordinate s are kept. All the tensors $\mathcal{D}$, $\mathcal{E}$, $\mathcal{F}$, $\mathcal{G}$, $\mathcal{H}$ as well as the metric tensor, are therefore, functions of the parallel coordinate s only.

5. Spectral representation

GKW uses a combination of finite difference techniques and pseudo-spectral methods (similar to GS2 [9,10]). Due to the homogeneous nature of the turbulence all perturbed quantities in the plane perpendicular to the magnetic field are represented by a sum over Fourier modes

$$f(\psi,\zeta,s) = \sum_{k_\zeta k_\psi} \hat{f}(k_\psi,k_\zeta,s)\exp[\mathrm{i}k_\zeta\zeta/\rho_* + \mathrm{i}k_\psi\psi/\rho_*] = \mathcal{T}^{-1}(\hat{f}), \tag{50}$$

where the normalised Larmor radius (ρ_*) has been added for convenient normalisation so that

$$\rho_*\frac{\partial}{\partial x_\alpha} \to \mathrm{i}k_\alpha. \tag{51}$$

In what follows a $\widehat{\cdot}$ indicates the Fourier representation of a quantity and $\mathcal{T}$ and $\mathcal{T}^{-1}$ represent the forward and inverse Fourier transform operations, respectively. The use of Fourier harmonics means that the boundary conditions in the perpendicular plane are periodic. The periodicity of the torus is, however, not automatically satisfied. The condition of toroidal periodicity can be formulated as

$$f(\psi,\zeta+1,s) = f(\psi,\zeta,s) \to \frac{k_\zeta}{2\pi\rho_*} = N, \tag{52}$$

where N is an integer. Since ρ_* is small, this condition can in general be satisfied with very small changes to k_ζ or ρ_*. In the local limit employed here, the final equations are independent of ρ_* and it is assumed that the relation above is satisfied. The poloidal periodicity translates to

$$f(\psi,\zeta+q/2,1/2) = f(\psi,\zeta-q/2,-1/2). \tag{53}$$

Expanding the safety factor around a reference value q_R (value at the centre of the radial domain)

$$\frac{qk_\zeta}{\rho_*} = \frac{q_R k_\zeta}{\rho_*} + k_\zeta\frac{\partial q}{\partial\psi}\frac{\psi}{\rho_*} + \frac{1}{2}k_\zeta\rho_*\frac{\partial^2 q}{\partial\psi^2}\left(\frac{\psi}{\rho_*}\right)^2, \tag{54}$$

and neglecting the second derivative correction (which gives only a ρ_* variation over the size of the box) one can see that this condition can be satisfied if also $q_R k_\zeta/2\pi\rho_*$ is assumed to be an integer and if

$$\hat{f}\left(k_\psi,k_\zeta,\frac{1}{2}\right) = \hat{f}\left(k_\psi + k_\zeta\frac{\partial q}{\partial\psi},k_\zeta,-\frac{1}{2}\right), \tag{55}$$

i.e. on the boundary of the s domain one simply connects the mode with the appropriate higher k_ψ mode. This formulation is close to the 'ballooning transform' [34], and has previously been employed in GS2 [9,10]. The above boundary condition for the Fourier amplitudes implies that a convenient choice for the spacing of the k_ψ modes in any discrete Fourier representation is

$$\Delta k_\psi = k_{\zeta,\min}\frac{\partial q}{\partial\psi}\frac{1}{p}, \tag{56}$$

where p is some integer. The integer p allows for a control over the spacing between the radial modes which otherwise would be dictated by the magnetic shear.

In order to specify the wave vector in normalised form, the expression

$$(k_\theta\rho_{\mathrm{ref}})^2 = g^{\zeta\zeta}k_\zeta^2 \tag{57}$$

is evaluated at the low field side midplane to determine k_ζ from the value of $k_\theta\rho_{\mathrm{ref}}$ given as an input (here k_θ is not a normalised quantity). The full perpendicular wave vector is given by

$$k_\perp^2 = g^{\alpha\beta}k_\alpha k_\beta. \tag{58}$$

One of the main advantages of the Fourier representation is that the gyro-average becomes an algebraic operation

$$\langle\widehat{\phi}\rangle = J_0(k_\perp\rho)\widehat{\phi}, \tag{59}$$

where J_0 is the Bessel function and

$$(k_\perp\rho)^2 = \frac{m_R T_R}{Z^2 B^2}g^{\alpha\beta}k_\alpha k_\beta. \tag{60}$$

In the equation above Z is the charge number of the species considered. GKW can run with an arbitrary number of species, which have arbitrary mass and charge.

6. The complete set of equations in the collisionless limit

In this section we give the complete set of equations based on the material introduced in the previous sections. The gyro-kinetic equation can be written in the form

$$\frac{\partial g}{\partial t} = \mathrm{I} + \mathrm{II} + \mathrm{III} + \mathrm{IV} + \mathrm{V} + \mathrm{VI} + \mathrm{VII} + \mathrm{VIII}, \tag{61}$$

with

$$\mathrm{I} = -v_\parallel \mathbf{b}\cdot\nabla f \to -v_R v_\parallel \mathcal{F}\frac{\partial \hat{f}}{\partial s}, \tag{62}$$

$$\mathrm{II} = -\mathbf{v}_D\cdot\nabla f \to -\frac{\mathrm{i}}{Z}\left[T_R E_D \mathcal{D}^\alpha + T_R v_\parallel^2 \beta' \mathcal{E}^{\psi\alpha} + 2m_R v_R v_\parallel \Omega \mathcal{H}^\alpha\right] k_\alpha \hat{f}, \tag{63}$$

$$\begin{aligned}\mathrm{III} &= -\mathbf{v}_\chi\cdot\nabla g \rightharpoonup -\rho_*^2 \frac{\partial \chi}{\partial x_\beta}\mathcal{E}^{\beta\alpha}\frac{\partial g}{\partial x_\alpha} \\ &= \rho_*^2 \mathcal{E}^{\psi\zeta}\left(\frac{\partial \chi}{\partial \zeta}\frac{\partial g}{\partial \psi} - \frac{\partial g}{\partial \zeta}\frac{\partial \chi}{\partial \psi}\right) \rightharpoonup \mathcal{T}\left(\mathcal{E}^{\psi\zeta}\left[\mathcal{T}^{-1}(\mathrm{i}k_\zeta \hat{\chi})\mathcal{T}^{-1}(\mathrm{i}k_\psi \hat{g}) - \mathcal{T}^{-1}(\mathrm{i}k_\zeta \hat{g})\mathcal{T}^{-1}(\mathrm{i}k_\psi \hat{\chi})\right]\right),\end{aligned} \tag{64}$$

$$\mathrm{IV} = +\frac{\mu B}{m}\frac{\mathrm{B}\cdot\nabla B}{B^2}\frac{\partial f}{\partial v_\parallel} \to v_R \mu B \mathcal{G}\frac{\partial \hat{f}}{\partial v_\parallel}, \tag{65}$$

$$\mathrm{V} = -\mathbf{v}_\chi\cdot\nabla F_M \to \mathrm{i}k_\alpha \hat{\chi}\mathcal{E}^{\alpha\psi}\left[\frac{R}{L_N} + E_T\frac{R}{L_T} + \frac{2v_\parallel}{v_R}\frac{u' B_t}{B}\right]F_M, \tag{66}$$

$$\mathrm{VI} = -\mathbf{v}_D\cdot\nabla F_M \to \frac{1}{Z}\left[T_R E_D \mathcal{D}^\psi + 2m_R v_R v_\parallel H^\psi\right]\times\left[\frac{R}{L_N} + E_T\frac{R}{L_T} + \frac{2v_\parallel}{v_R}\frac{u' B_t}{B}\right]F_M, \tag{67}$$

$$\mathrm{VII} = -\frac{Ze}{T}v_\parallel \mathbf{b}\cdot\nabla\langle\phi\rangle F_M \to -\frac{Z}{T_R}v_R v_\parallel \mathcal{F}\frac{\partial\langle\widehat{\phi}\rangle}{\partial s}F_M, \tag{68}$$

$$\mathrm{VIII} = -\frac{Ze}{T}\mathbf{v}_D\cdot\nabla\langle\phi\rangle F_M \to -\mathrm{i}\left[E_D\mathcal{D}^\alpha + \beta' v_\parallel^2 \mathcal{E}^{\psi\alpha} + \frac{2m_R v_R}{T_R}v_\parallel \Omega\mathcal{H}^\alpha\right]k_\alpha\langle\widehat{\phi}\rangle F_M, \tag{69}$$

where the arrows represent the transformations to normalised ($\rightharpoonup$) and Fourier ($\to$) quantities and where

$$\hat{\chi} = \langle\widehat{\phi}\rangle - 2v_R v_\parallel \langle\widehat{A}_\parallel\rangle, \tag{70}$$

and

$$\hat{g} = \hat{f} + \frac{2Z}{T_\mathrm{R}}v_\mathrm{R} v_\parallel \langle\widehat{A}_\parallel\rangle F_M, \tag{71}$$

$$E_D = v_\parallel^2 + \mu B, \qquad E_T = v_\parallel^2 + 2\mu B - \frac{3}{2}. \tag{72}$$

The equations above apply to each of the species individually.

The only nonlinearity in the equations is the $\mathbf{v}_\chi\cdot\nabla g$ term (term III, Eq. (64)). Linear stability can be investigated by removing this term. In the linear case the toroidal modes are decoupled, but different radial modes can still be coupled over the parallel boundary conditions. An unstable mode will grow to infinite amplitude, and GKW therefore applies a scaling of the equations at every time-step to obtain a stationary solution. The nonlinear term (III) is implemented using FFTs. The velocity $\mathbf{v}_\chi$ and ∇g are calculated in Fourier space, then transformed to real space and multiplied. The product is transformed back into Fourier space.

The fields ϕ and $A_\parallel$ have to be obtained through a solution of the Poisson equation and Ampere's law. These can be found in the literature [28]. Using the normalisations of GKW the Poisson equation can be written in the form

$$\sum_{sp} Z_{sp} n_{Rsp}\left[2\pi B\int \mathrm{d}v_\parallel\,\mathrm{d}\mu\; J_0(k_\perp\rho_{sp})\hat{g}_{sp} + \frac{Z_{sp}}{T_{Rsp}}\left[\Gamma(b_{sp}) - 1\right]\widehat{\phi}\right] = 0, \tag{73}$$

where

$$b = \frac{1}{2}m_R T_R\left(k_\perp \rho_* R_{\mathrm{ref}}/ZB^2\right)^2 = \overbrace{\frac{1}{2}\frac{k_\perp^2 m^2 v_{\mathrm{th}}^2}{e^2B^2}}^{\text{in original units}}. \tag{74}$$

An often employed approximation for the electron dynamics is the so-called adiabatic response. In the adiabatic electron limit only the ion dynamics is simulated and the Poisson equation above is evaluated as

$$\sum_{sp,\mathrm{ions}} Z_{sp} n_{Rsp}\left[2\pi B\int \mathrm{d}v_\parallel\,\mathrm{d}\mu\; J_0(k_\perp\rho_{sp})\hat{g}_{sp} + \frac{Z_{sp}}{T_{Rsp}}\left[\Gamma(b_{sp}) - 1\right]\widehat{\phi}\right] = \frac{n_{Re}}{T_{Re}}\left(\widehat{\phi} - \{\widehat{\phi}\}\right), \tag{75}$$

where the index e refers to the electrons, and the sum on the left-hand side of the equation is over all ion species.

The vector potential must be calculated from Ampere's law

$$\left(k_\perp^2+\beta\sum_{sp}\frac{Z_{sp}^2 n_{Rsp}}{m_{Rsp}}\Gamma(b_{sp})\right)\widehat{A}_\parallel=\beta\sum_{sp}Z_{sp}v_{Rsp}n_{Rsp}\times 2\pi B\int \mathrm{d}v_\parallel\int \mathrm{d}\mu\, v_\parallel J_0\hat{g}_{sp}. \tag{76}$$

Note that both field equations are formulated for $\hat{g}$ rather than f. The reason is that it is g which is integrated forward in time. Only after the fields are calculated for the new time point, can f be reconstructed. The Maxwell correction in g leads to the second term in the brackets on the left-hand side of the equation above. This term can lead to a cancellation problem, and the integral over the Maxwellian is therefore performed numerically in a similar manner as the term on the right-hand side [13] rather than the analytic form which is given above.

7. Collisions

The previous sections have neglected the collision operator. In this section we describe the collisions as implemented in GKW. The present implementation neglects the finite Larmor radius effects. The collisions enter the evolution equation as an additional term

$$\frac{\partial\hat{f}_a}{\partial t}=C(\hat{f}_a). \tag{77}$$

Here we have added the index a to denote the species. Useful expressions for the collision operator (C) have been published in the literature, usually in the (v,θ) coordinate system, where v is the velocity and θ is the pitch angle. Since the perturbed distribution function is small $f\approx\rho_* F_M$, GKW uses a linearised collision operator with the Maxwellian as background. This operator has the following form [35]

$$C(\hat{f}_a)=\sum_b\frac{1}{v^2}\frac{\partial}{\partial v}\left[v^2\left(D_{vv}^{a/b}\frac{\partial\hat{f}_a}{\partial v}-F_v^{a/b}\hat{f}_a\right)\right]+\frac{1}{v\sin\theta}\frac{\partial}{\partial\theta}\left[\sin\theta D_{\theta\theta}^{a/b}\frac{1}{v}\frac{\partial\hat{f}_a}{\partial\theta}\right]. \tag{78}$$

Here the sum is over all species b. $D_{\theta\theta}$ represents the pitch angle scattering, while D_{vv} is the energy scattering and F_v is the slowing down force. For convenience we split this collision operator in three parts

$$C(\hat{f})=C_\theta(\hat{f})+C_{vv}(\hat{f})+C_v(\hat{f}), \tag{79}$$

where the first part is due to $D_{\theta\theta}$, the second due to D_{vv} and the third due to F_v. The normalisation of the collision operator follows from the normalisation of the gyro-kinetic equation as set out in Section 3, giving $C_N(\hat{f}_a)=\frac{R_{\mathrm{ref}}v_{\mathrm{tha}}^3}{n_a\rho_* v_{\mathrm{thref}}}C(\hat{f}_a)$ with the diffusion and friction force coefficients made dimensionless accordingly.

The coefficients can be obtained from the literature [35] (and the references cited therein) and written in normalised form, suppressing the subscript N for the normalised quantities:

$$D_{\theta\theta}^a=\sum_b\frac{\Gamma^{a/b}}{4v_a}\left[\left(2-\frac{1}{v_b^2}\right)\mathrm{erf}(v_b)+\frac{1}{v_b}\mathrm{erf}'(v_b)\right], \tag{80}$$

$$D_{vv}^a=\sum_b\frac{\Gamma^{a/b}}{2v_a}\left[\frac{1}{v_b^2}\mathrm{erf}(v_b)-\frac{1}{v_b}\mathrm{erf}'(v_b)\right], \tag{81}$$

$$F_v^a=-\sum_b\frac{\Gamma^{a/b}}{v_a^2}\frac{m_{Ra}}{m_{Rb}}\left[\mathrm{erf}(v_b)-v_b\,\mathrm{erf}'(v_b)\right], \tag{82}$$

where erf is the standard definition of the error function. The normalised velocity in these equations is defined as

$$v_\gamma=\frac{v_{Ra}}{v_{R\gamma}}\left[v_\parallel^2+2\mu B\right]^{1/2}, \tag{83}$$

with $\gamma=a$ or $\gamma=b$. Finally the constant $\Gamma^{a/b}$ is given by

$$\Gamma^{a/b}=\frac{R_{\mathrm{ref}}n_b Z_a^2 Z_b^2 e^4\ln\Lambda^{a/b}}{4\pi\epsilon_0^2 m_a^2 v_{\mathrm{th}a}^4}=6.5141\cdot 10^{-5}\frac{R_{\mathrm{ref}}n_{\mathrm{ref}}^{19}}{T_{\mathrm{ref}}^k}\frac{n_{Rb}Z_a^2Z_b^2\ln\Lambda^{a/b}}{T_{Ra}^2}, \tag{84}$$

where R_{ref} must be given in meters, n_{ref}^{19} is the reference density in units of 10^{19} m^{-3}, and T_{ref}^k is the reference temperature in units keV. Note that the reference density and temperature are defined independently for the collisions and do not need to be equal to the reference density and temperature used in Section 3 for the species parameters.

Using the relation between (v,θ) and the normalised coordinates $(v_\parallel,\mu)$ used in GKW, and performing the coordinate transform, one arrives at the explicit expressions given below. The pitch angle scattering term is

$$C_\theta\to 2\nu_R\frac{\partial}{\partial v_\parallel}\left[\frac{\mu D_{\theta\theta}}{v_\parallel^2+2\mu B}\left(B\frac{\partial\hat{f}}{\partial v_\parallel}-v_\parallel\frac{\partial\hat{f}}{\partial\mu}\right)\right]+\frac{2\nu_R}{B}\frac{\partial}{\partial\mu}\left[\frac{v_\parallel\mu D_{\theta\theta}}{v_\parallel^2+2\mu B}\left(-B\frac{\partial\hat{f}}{\partial v_\parallel}+v_\parallel\frac{\partial\hat{f}}{\partial\mu}\right)\right]. \tag{85}$$

The energy scattering yields

$$C_{vv}\to\nu_R\frac{\partial}{\partial v_\parallel}\left[\frac{v_\parallel D_{vv}}{v_\parallel^2+2\mu B}\left(v_\parallel\frac{\partial\hat{f}}{\partial v_\parallel}+2\mu\frac{\partial\hat{f}}{\partial\mu}\right)\right]+\nu_R\frac{\partial}{\partial\mu}\left[\frac{2\mu D_{vv}}{v_\parallel^2+2\mu B}\left(v_\parallel\frac{\partial\hat{f}}{\partial v_\parallel}+2\mu\frac{\partial\hat{f}}{\partial\mu}\right)\right]. \tag{86}$$

The slowing down can be written as

$$C_v \to -\nu_R \frac{\partial}{\partial v_\parallel}\left[\frac{v_\parallel F_v}{[v_\parallel^2+2\mu B]^{1/2}}\hat{f}\right] - \nu_R \frac{\partial}{\partial \mu}\left[\frac{2\mu F_v}{[v_\parallel^2+2\mu B]^{1/2}}\hat{f}\right]. \tag{87}$$

The linearised form of the Fokker–Planck collision operator above conserves particle number but does not conserve momentum or energy. The conservation of parallel momentum can be reintroduced using the following simplified model. We are interested in momentum conservation for like particle collisions. The change in parallel momentum due to the collision operator above is

$$\begin{aligned} 2\pi B\int \mathrm{d}v_\parallel \int \mathrm{d}\mu\, v_\parallel C(\hat{f}) = &-4\pi B\nu_R \int \mathrm{d}v_\parallel\, \mathrm{d}\mu\, \frac{\mu D_{\theta\theta}^{a/a}}{v_\parallel^2+2\mu B}\left(B\frac{\partial \hat{f}}{\partial v_\parallel} - v_\parallel \frac{\partial \hat{f}}{\partial \mu}\right) \\ &- 2\pi B\nu_R \int \mathrm{d}v_\parallel\, \mathrm{d}\mu \frac{v_\parallel D_{vv}^{a/a}}{v_\parallel^2+2\mu B}\left(v_\parallel \frac{\partial \hat{f}}{\partial v_\parallel} + 2\mu \frac{\partial \hat{f}}{\partial \mu}\right) \\ &- 2\pi B\nu_R \int \mathrm{d}v_\parallel\, \mathrm{d}\mu\, \frac{v_\parallel F_v^{a/a}\hat{f}}{[v_\parallel^2+2\mu B]^{1/2}}. \end{aligned} \tag{88}$$

This loss of momentum can be corrected for by adding an ad hoc correction term

$$\frac{\partial \hat{f}}{\partial t} = C_{\mathrm{mom}} v_\parallel F_M. \tag{89}$$

The constant C_{mom} then follows from working out the momentum change due to this term.

$$2\pi B C_{\mathrm{mom}} \int \mathrm{d}v_\parallel\, \mathrm{d}\mu\, v_\parallel^2 F_M + 2\pi B \int \mathrm{d}v_\parallel \int \mathrm{d}\mu\, v_\parallel C(\hat{f}) = 0. \tag{90}$$

No attempt is made in GKW to conserve the energy in the collision operator. The lack of energy conservation is considered less crucial compared with momentum conservation since the present form of the collision operator drives the distribution towards a Maxwellian with the specified temperature. In the local problem one specifies the temperature as an input parameter and no evolution in the temperature is retained. A collision operator that maintains this temperature can therefore be considered acceptable.

8. Background $E \times B$ shear flow

The co-moving system employed in GKW corresponds to a frame that rotates as a rigid body with constant frequency Ω. The possible radial gradient in the toroidal rotation is then treated through a radial gradient in the averaged parallel velocity of the background. The radial gradient of the perpendicular velocity has so far been ignored, but is known to play an important role in the saturation of the turbulence. The sheared $E \times B$ velocity leads to a stabilisation of turbulence through the breaking up of eddy structures. This $E \times B$ shearing is always present for a purely toroidally rotating plasma but can, of course, also be related to a sheared poloidal rotation. In this section it is described how the $E \times B$ shearing is treated in GKW.

The equilibrium $E \times B$ rotation

$$\mathbf{v}_s(\psi) = \frac{\mathbf{b}\times\nabla\Phi}{B} \tag{91}$$

lies inside the flux surface. The shearing rate in the normalised units is defined to be

$$\gamma_E^N = \frac{1}{2}\rho_*^2 \frac{\partial^2 \Phi_N}{\partial \psi^2}, \tag{92}$$

where the normalisation $\Phi = \rho_* \frac{T_{\mathrm{ref}}}{e}\Phi_N$ is as for the perturbed potential equation (19). The factor 1/2 is present due to the definition of the reference thermal velocity. In physical units the shearing rate is

$$\gamma_E = \frac{v_{\mathrm{thref}}}{R_{\mathrm{ref}}}\gamma_E^N = \frac{1}{B_{\mathrm{ref}}}\frac{\partial^2 \Phi}{\partial r^2}, \tag{93}$$

where $r = (R_{\max} - R_{\min})/2$. The GKW shearing rate is assumed radially constant and is defined as a flux function. At the radial centre of the flux tube $\mathbf{v}_s$ is chosen to be zero, with the result that there is no net flow over the domain. In the local limit with the approximations of the '$s-\alpha$' model geometry the shearing rate is then equivalent to the familiar definition [39,40]:

$$\gamma_E \approx \frac{(RB_p)^2}{B_t}\frac{\partial^2 \Phi}{\partial \Psi^2} \approx \frac{\mathrm{d}v_s}{\mathrm{d}r}. \tag{94}$$

As before the index N is dropped in what follows. The sign convention here is opposite to Eq. (23), so $\gamma_E < 0$ corresponds to ∇E_0 radially outwards.

Only the Doppler shift of the background $E \times B$ rotation is kept in the description, i.e. the background $E \times B$ rotation is added as an additional convective term for the perturbed distribution $(\mathbf{v}_s \cdot \nabla g)$ in the gyrokinetic equation (61), and we neglect the acceleration due to the background potential. With zero net flow across the flux tube one obtains

$$\mathrm{IX} = -\mathbf{v}_s \cdot \nabla g \rightharpoonup -\rho_*^2 \frac{\partial \Phi}{\partial \psi}\mathcal{E}^{\psi\zeta}\frac{\partial g}{\partial \zeta} = -2\gamma_E \psi \mathcal{E}^{\psi\zeta}\frac{\partial g}{\partial \zeta}. \tag{95}$$

Defining $\bar{\gamma}_E = 2\gamma_E \mathcal{E}^{\psi\zeta}$ and taking the Fourier transform, the term can be written as a derivative in Fourier space

$$\mathcal{T}\left(-\bar{\gamma}_E \psi \frac{\partial g}{\partial \zeta}\right) = \bar{\gamma}_E k_\zeta \frac{\partial \hat{g}}{\partial k_\psi}. \tag{96}$$

The derivative represents a continuous advection (and shearing) in Fourier space in k_ψ.

There are a number of subtleties associated with numerical evaluation of this derivative. A finite difference approximation cannot resolve the fine scale structure in Fourier space. The 'harmonic derivative' [41,42] is a full convolution over all modes and is equivalent to a pseudo-spectral implementation using FFTs (as was used for the nonlinear term in Eq. (64)). Although it correctly implements the $E \times B$ shearing term inside the computational domain, it does not represent a homogeneous shear flow because the flow profile does not satisfy the periodicity of the discrete Fourier transform; any turbulent structure passing through the radial boundary will experience a discontinuity in the flow profile (which in the periodic domain has a sawtooth form). Since the local limit requires that the turbulence has no preference for the boundary, periodicity at the boundary must be preserved. The 'shear-periodic' boundary condition [43–45] that moves with the mean flow accomplishes this for a finite difference representation in the radial direction, but does not naturally translate to the spectral representation.

In coordinates that move with the flow [46–48], the 'radial' wavenumbers become time dependent:

$$\psi' = \psi, \qquad \zeta' = \zeta - \psi \bar{\gamma}_E t, \tag{97}$$

$$k'_\zeta = k_\zeta, \qquad k'_\psi = k_\psi - k_\zeta \bar{\gamma}_E t. \tag{98}$$

In these coordinates, the derivative in Fourier space becomes part of the time derivative.

For a gyro-kinetic code, a time dependent wave vector requires the re-evaluation of the linear terms and Bessel functions at every time-step and would be computationally prohibitive. By discretising the time dependence of the wavenumbers as a remapping of the solution vector between the original fixed wavevectors, this expensive re-evaluation can be avoided [49]. Using only the fixed wavevector grid, the advection in Fourier space occurs in jumps only when the boundary periodicity is satisfied for a given mode. Explicitly, the remapping $\hat{g}(k_\psi, k_\zeta, s) \to \hat{g}(k_\psi - \Delta k_\psi, k_\zeta, s)$ occurs when

$$k_\zeta \bar{\gamma}_E (t - t_{\text{last remap}}) = \Delta k_\psi / 2. \tag{99}$$

At the high k_ψ limit the solution vector is discarded. This means that the method is nonconservative, but since the turbulence is characterised by a peaked spectrum (Fig. 2), the losses are negligible if the range of radial wavenumbers is suitably wide. For numerical stability, the remapping must occur at the same time for all points along the flux tube. As the metric tensor $\mathcal{E}^{\psi\zeta}$ is a flux function, see Eq. (39), this condition is always satisfied for a constant shear rate along the field line.

The method in GKW differs from the standard spectral methods to model homogeneous shear flows in fluid turbulence of Refs. [46–48], where the wavenumbers must be recalculated and re-meshing occurs for all wavevectors at the same time. In GKW the method used is the one proposed by Hammett et al. in Ref. [49]. Here the 're-meshing' occurs continually (and at different times for each k_ζ), and the wavenumbers stay on their original fixed grid. The accuracy of the method has been argued to be second order on average [49] and is able to capture the physics effects of a background shear flow whilst allowing the desirable features of the flux tube model to be retained.

Convergence of the remap method can be (and should be) checked (particularly by for the modes with lowest k_ζ) by decreasing $\Delta k_\psi / k_{\zeta\min}$ (increasing N_x whilst holding N_{mod} constant (defined in the next section)).

For purely toroidal rotation (with no poloidal flows) the poloidal components of the parallel flow $u_\parallel = R\Omega B_t / B$ and the $E \times B$ flow $\mathbf{v}_s$ must cancel:

$$v_s = -\frac{B_p}{B_t} u_\parallel = -\frac{B_p R \Omega}{B}. \tag{100}$$

For $s_B = 1$, $v_s > 0$ indicates a flow in increasing poloidal direction, $\mathbf{v}_s \cdot \nabla\theta > 0$. For $s_j = 1$, $\mathbf{B} \cdot \nabla\theta > 0$, with the poloidal component of the magnetic field positive, $B_p > 0$. For the gradients

$$\gamma_E = -\frac{\partial}{\partial \psi}\left(\frac{B_p R_{\text{ref}}}{|\nabla\psi|}\Omega\right) \approx \frac{B_p u'}{B}, \tag{101}$$

since the definitions (23) and (92) have opposite sign.

9. Numerical implementation

In this section we outline the details of the numerical solution of the equations. GKW uses a combination of finite difference and spectral methods. The turbulence in the plane perpendicular to the magnetic field is homogeneous and the solution in the ζ, ψ plane is represented by Fourier modes, as mentioned previously. All other directions are treated using finite difference techniques. In what follows N_{mod} and N_x are the number of bi-normal (ζ direction) and radial (ψ direction) modes respectively, N_{sp} is the number of kinetic species, N_s the number of grid points along the field line. $N_{v\parallel}$ is the number of grid points in the parallel velocity direction and N_μ the number of magnetic moment grid points.

9.1. Derivatives along the magnetic field line and in the parallel velocity direction

Terms I, IV and VII on the right-hand side of Eq. (61), involving a derivative along the magnetic field line or in the parallel velocity direction are considered here. These terms correspond to an advection with a spatially dependent advective velocity and can be written in the form $-v(s)\frac{\partial g}{\partial s}$. The second- and fourth-order centred-differences used for these terms can then be written

$$v\frac{\partial g}{\partial s} \to v_i \frac{-g_{i-1}+g_{i+1}}{2\Delta s} - D|v_i|\frac{g_{i-1}-2g_i+g_{i+1}}{2\Delta s}, \tag{102}$$

$$v\frac{\partial g}{\partial s} \to v_i \frac{g_{i-2}-8g_{i-1}+8g_{i+1}-g_{i+2}}{12\Delta s} - D|v_i|\frac{-g_{i-2}+4g_{i-1}-6g_i+4g_{i+1}-g_{i+2}}{12\Delta s}, \tag{103}$$

where the terms involving a dissipation coefficient D correspond to (hyper)diffusive upwind dissipation. This dissipation is not included in the case of term VII (containing derivatives of ϕ) for numerical stability reasons.

Rather than apply this central differencing scheme separately to terms I and IV, the code user can choose an alternative scheme, which combines the derivative along the field line with the trapping term. Defining $H(s, v_\parallel) = \frac{1}{2}v_\parallel^2 + \mu B$, and noting that $B\mathcal{G} = \mathcal{F}\frac{\partial B}{\partial s}$ in IV, we can combine terms I and IV to give

$$v_R\mathcal{F}\left(\mu\frac{\partial B}{\partial s}\frac{\partial \hat{f}}{\partial v_\parallel} - v_\parallel\frac{\partial \hat{f}}{\partial s}\right) = v_R\mathcal{F}\{H, \hat{f}\}, \tag{104}$$

where $\{H, \hat{f}\}$ is a Poisson bracket (note that we have neglected the redundant dependence of μ in the definition of H for simplicity). Following the second- and fourth-order schemes of Arakawa [56], we allow the Poisson bracket to be differenced directly. The terms containing D in Eqs. (102) and (103) are also added to allow some dissipation. The advantage of this scheme over the separate differencing scheme is that usually zero (or relatively small) dissipation is needed to obtain a stable solution. Presently this is implemented only as a fourth-order scheme.

9.2. Boundary conditions along a field line

We here discuss the implementation of the boundary conditions in the parallel direction s, that are presented in Section 5. After one poloidal turn the Fourier mode connects to a mode with a different radial wave vector. The grid in the s direction is therefore constructed such that at the end of the field line it connects smoothly to the start of the field line (i.e. the grid spacing remains constant). Of course, the radial wave vector to which the mode has to connect might not be represented on the grid. At the end (or beginning) of the field line a boundary condition is then employed. GKW allows for a choice in this boundary condition, setting the perturbed distribution either to zero or to have a zero derivative in the direction upwind with respect to the convection. The down wind direction is always treated with a zero derivative boundary condition, allowing the distribution to 'flow of the grid', without reflections. Close to the end point boundaries a staggered change in differencing order accuracy is used to remove ghost cells and to allow the flow of the distribution function out of the domain. Reflection or the allowed build up at the upper velocity boundaries is seen as unphysical and therefore undesirable. Because of the use of up-winding the boundary condition is dependent on the sign of the advective velocity.

When the advective velocity $v(s)$ is positive the distribution function will move toward the left boundary, and the following scheme is used

$$\text{Boundary cell:}\quad \text{Second-order scheme}\quad -\frac{\partial g}{\partial s} \to \frac{-\frac{3}{2}g_n + 2g_{n+1} - \frac{1}{2}g_{n+2}}{\Delta s},$$

$$\text{Adjacent cell:}\quad \text{Third order}\quad -\frac{\partial g}{\partial s} \to \frac{-\frac{1}{3}g_{n-1} - \frac{1}{2}g_n + g_{n+1} - \frac{1}{6}g_{n+2}}{\Delta s}$$

with the right-hand boundary using the centrally differenced schemes. When the advective velocity is negative the opposite is true and the right-hand boundary is differenced in an identical way. To obtain the scheme for the opposing boundary simply reverse the signs of the equation above. When using the second-order scheme, special consideration is needed only for the boundary cell. Depending on the direction of the advective velocity, a first-order back-winded scheme is used at the boundary. The effect of these open boundary conditions is to remove the need for ghost cells in the differencing scheme while maintaining a high order scheme. Compared to other alternatives we have found that the chosen scheme removes rapid fluctuations near the end points, and therefore allows for a minimum dissipation. In the Poisson bracket formulation (104), the above convention is integrated into the scheme at the boundaries to have similar properties.

9.3. The collision operator

The collision operator can be thought of as a convection/diffusion equation in velocity space and thus is differenced accordingly. The velocity space boundary conditions are taken to be zero flux, which occurs naturally at the $\mu = 0$ boundary and is artificially imposed on the other three boundaries. These boundary conditions guarantee the conservation of mass. Considering that the diffusion coefficient is velocity space dependent, and the grid can be nonuniform in the μ-direction, the differencing scheme of the equations is chosen as follows for the parallel velocity direction

$$\frac{\partial}{\partial v_\parallel}\left(D\frac{\partial \hat{f}}{\partial v_\parallel}\right) \to \frac{1}{v^j_{i+1/2\parallel} - v^j_{i-1/2\parallel}}\left(D^j_{i+1/2}\left(\frac{f^j_{i+1} - f^j_i}{v^j_{i+1\parallel} - v^j_{i\parallel}}\right) - D^j_{i-1/2}\left(\frac{f^j_i - f^j_{i-1}}{v^j_{i\parallel} - v^j_{i-1\parallel}}\right)\right), \tag{105}$$

where i denotes the point in the parallel velocity direction and j the μ direction. When considering the cross-terms a four point interpolation method is used to calculate the half point values. Four points are needed on a two-dimensional grid. The friction term is differenced in the same way, but with a two point interpolation.

9.4. The Fourier representation

In a nonlinear run a rectangular grid of wave vectors (k_ζ, k_ψ) is used. The nonlinearity is treated using fast Fourier transforms (FFT). Different choices of FFT libraries are possible, but at present we use mostly the FFTW library [38]. For the nonlinear term a de-aliasing method is applied, i.e. the transformation to real space is constructed on a grid finer (number of grid points at least 3/2 larger in real

space) than the Fourier modes represent. The de-aliasing method is nonconservative and has a dissipative effect on the finer grid scales, with the consequence that a numerical dissipation term in the spectral directions is generally not required.

Since the distribution function is a real quantity the number of complex Fourier modes can be reduced by a factor two and a half plane of wavevectors with $k_\zeta \geqslant 0$ is used. At present, grid sizes in real space are chosen to be a power of 2. The grid sizes in real space for bi-normal direction is

$$M_{\mathrm{mod}} = 2^{\mathrm{int}[\log_2(3(N_{\mathrm{mod}}-1))+1]} \tag{106}$$

(where N_{mod} is the number of bi-normal modes) and for the radial direction

$$M_x = 2^{\mathrm{int}[\log_2(\frac{3}{2}(N_x+1))+1]} \tag{107}$$

(where N_x is the number of radial modes). The consequence is that some choices of N_{mod} and N_x make more efficient use of the FFT than others. FFTW can treat grid sizes that are not a power of two, and future work will lift the limitation described above.

9.5. Time integration

GKW has several options for the explicit time integration: modified midpoint, fourth-order Runge–Kutta (recommended), and a second-order scheme that is stable for hyperbolic equations. An implicit scheme is implemented, but insufficiently tested and not part of the present release.

10. The code

The package GKW consists of a README file (where a description of the full contents can be found), Fortran 95 source files, a simple makefile, some documentation and some sample code input files. This section describes what the code can do in terms of parallelisation and how it is structured with respect to the solution of the equations presented in the previous sections. In order to do this, we follow the various tasks the code performs as it runs and discuss the relevant modules to that task. Instructions for building, installing and running the code can be found in Section 12.

Each source code file contains a module, except the main program file '`linart.f90`'. Each source file has either the extension '`.f90`' or '`.F90`', the latter denoting the need to be pre-processed. For the modules, each module name corresponds directly to the file name (which is always lower case), minus the suffix. Therefore, in the following subsections, when referring to the module *GRID*, the corresponding source code can be found in the file '`grid.F90`'.

The code does not come with any routines to perform FFTs, which are required for the computation of the nonlinear term in Eq. (64), and to transform some of the diagnostics into real space. No FFT's are required for a linear run and the code can be compiled without FFT functionality. The FFT's are presently performed via the external FFTW3 library [38], for which the *FFT* module provides an interface. Although more interfaces for various other portable Fortran FFT libraries could be provided, the performance of the FFTW3 library has been found to exceed that of other alternatives significantly. In addition, the FFTW3 library is fairly portable, so to date there has been no good reason to consider alternatives.

GKW is parallelised using the Message Passing Interface (MPI). Each parallel process is responsible for solving the equations over a subdomain of the space and a subset of the species. A process usually only has knowledge of the part of the solution it is responsible for, unless explicitly communicated between processes using the MPI library. The parallelisation is discussed in more detail in a later section.

10.1. Initialisation

At the start of a run, the code will first call various basic MPI routines. These give each process an individual rank (*processor_number*) and inform each process of the total number of processors available for the computation (*number_of_processors*). Checks are then performed to see if various output or restart files are present.

After this, the main input file is read. A single processor is responsible for doing this. The input file contains several Fortran namelists, which are described later. The first namelist is called *control* and is read in the module *CONTROL*, and is responsible for controlling the basic switches for the physics, numerics and some of the code output. Each module requiring input is responsible for reading its input from this file. After each namelist is read, the processor that read the file broadcasts the information to all the other processors via MPI. All processors can then individually check that the input is allowed or satisfactory, and if necessary, abort the run.

10.2. Parallelisation: local grid

The sizes of the local and global grids and associated quantities are dealt with in the module *GRID*. The namelist *gridsize* contains the input variables for the grid sizes. The code can decompose the calculation over the s, $v_\parallel$ and μ coordinates and the species. Given the number of processors as input, the routines of *GRID* decide how the global solution will be subdivided between the processors. Alternatively, a user can specify explicitly the number of processors over which to subdivide the problem in each divisible direction. Presently, the automatic choice made by the code does not include the possibility of parallelising over the s grid. All other possibilities are considered, apart from some restrictions for the case of collisions, and the Arakawa style differencing scheme. If a user explicitly sets an invalid combination, the code will abort at this point or earlier. The minimum number of grid points per processor in a given direction is limited to either 1 (when using the second-order finite differences) or 2 (when using the fourth-order scheme). For the decomposition over the species, 1 species per processor is the minimum. These minimum grid sizes dictate the maximum number of processors that can be used in a given direction; the maximum is either the number of grid points in that direction (for the second-order scheme) or half that value (for the fourth-order scheme). The parallelisation is discussed in more detail in Section 11.

A processor is provided with various logical variables denoting if it is at a particular boundary in the global domain together with the ranks of the processors that it will need to communicate with (which are usually the ones responsible for adjacent parts of the computational domain).

Once the local grid sizes have been confirmed, some MPI communicators are set up to facilitate communication between processes responsible various subsets of the whole domain. These are found in the *MPICOMMS* module. They are particularly useful when performing a sum over a subset of the coordinates when all the relevant points are not present on the local processor.

10.3. Memory layout and allocation

The solution of the equation, $g(k_\zeta, k_\psi, s, \mu, v_\parallel)$ for each species, is stored in the one-dimensional array *fdisi* in the module *DIST*. This module manages how the solution is ordered within the array. The memory layout can be re-ordered to (primarily) make the code parallelisation more efficient and to (possibly) improve cache efficiency. The function *indx* is responsible for providing the memory location information to the rest of the code. Given a grid point in the 5-dimensional space and species number, the function will return the array index for that solution point. This is used extensively in the *LINEAR_TERMS* module.

In general, each module is responsible for allocating any arrays they require. In addition to the local solution, the *fdisi* array contains any fields that are evolved and parts of the solution from other processors which has been communicated via MPI. In the case of parallelising over the s grid, the fields are also decomposed, so parts of the potential from other processors must also be present in order to calculate the derivative of $\langle\phi\rangle$. The *indx* function deals with referencing the right part of *fdisi* so that other parts of the code do not need to make special arrangements for processor boundary points.

10.4. Linear terms

The linear terms are calculated in the *LINEAR_TERMS* and *COLLISIONOP* module and put into a matrix which is in *MATDAT*. The *LINEAR_TERMS* module is designed so that more new terms can be added by only adding one new routine and calling it from within that module.

11. Code parallelisation

At present, the parallelisation of the code is done via decomposition of the distribution function grid and the various species in a given run. Here we describe the various parallelisation options and their potential consequences for code performance. A parallel scaling test is also shown for a Cray XT4.

11.1. Simple parallelisation via the fields

The equation for the distribution function (without collisions) contains no derivatives in the magnetic moment μ, which means that the distribution function for any point in the μ grid can be updated independently of the other points in the μ grid. Similarly, the distribution function for each species can be updated independently. Therefore, one can run up to *number of grid points in mu* $\times$ *number of species* parallel processes (typically 32–64 processes) to update the distribution function if parallelising over these two 'directions'. However, these points are coupled via the fields, which must be updated before the distribution function can be calculated in an explicit time step. The field equations involve a summation over the whole of the velocity space and the species, for which the code requires an inter-process reduction sum. In parallelising over these 'trivial' directions, a process can first perform a partial sum over local species and μ-grid points, before participating in a sum involving all the parallel processes. For each field, this sum must be performed at every point in the 3-dimensional space and the result must be then known to every process. As the number of utilised processor cores increases, the amount of data per core that is involved in the sum remains constant. How much the parallel efficiency of the code decreases as the number of utilised processors is increased will depend on the algorithm used by MPI_ALLREDUCE and underlying system. In this simple parallelisation case (and potentially in other cases) this ALLREDUCE operation is the bottleneck in parallel performance.

11.2. Parallelisation over the $v_\parallel$ grid

The equations have derivatives in parallel velocity, which in their numerical implementation involve points of the distribution function along lines in the parallel velocity grid. The derivative at the local point requires at most the value of the distribution function at 1 or 2 adjacent points (second- or fourth-order scheme) in each direction along the parallel velocity grid (see Section 9.1). When decomposing the $v_\parallel$ grid, the calculation of a $v_\parallel$ derivative at the end of a local grid will require 1 or 2 more points managed by another process. These points need to be communicated from the adjacent processor before the derivatives are performed. Typically, the 2 points at either end of the local parallel velocity grid must be communicated to the corresponding adjacent process, and 2 points must be received from it. Therefore, if we parallelise over the $v_\parallel$ grid we have this communication to perform in addition to that required to calculate the fields as discussed in the previous subsection. On systems which support performing computation at the same time as communication (such as a Cray XT4), this communication can be overlapped with the relatively expensive nonlinear terms computation. However, in such a case we would still be left with the same field calculation bottleneck as the number of processors responsible for the parallel velocity grid is increased.

11.3. Parallelisation over the s grid

In terms of the communication required for the derivatives along the s direction, parallelisation over the s grid is much the same as for the $v_\parallel$ grid. However, splitting up the s direction decomposes the space over which the fields are calculated. This means that the ALLREDUCE operation for a given processor only needs to be performed between processors responsible for the same part of the s grid. If the number of processors over which the s grid is decomposed is doubled, then the data involved in the reduction sum is halved, and the number of processors that partake in the sum remains the same. A consequence of this is that when parallelising only in the s direction, no communication is needed to calculate the fields. This is typically very favourable for parallel performance; one potential bottleneck would be effectively removed (and the other can usually be overlapped with computation).

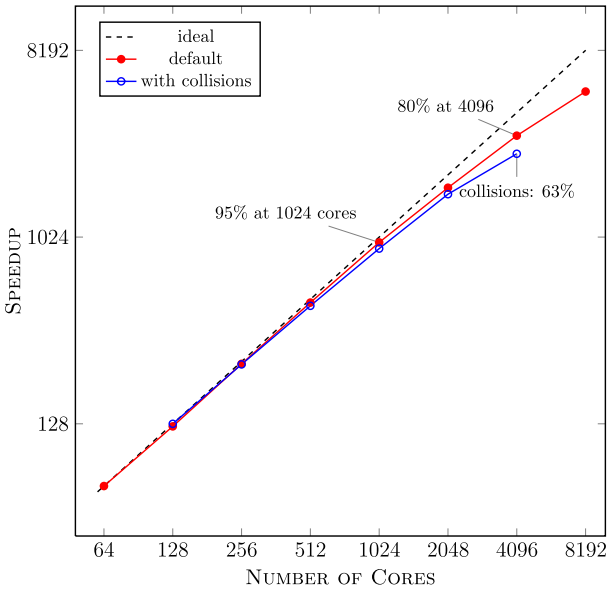


Fig. 1. Parallel performance for a typical sized problem on HECToR (Cray XT4), both without and with the collision operator turned on. Due to machine memory constraints, this problem can run with a minimum of 64 cores without collisions and 128 cores with. The performance/speedup is based on the time taken to perform the main time integration loop of the code and is scaled such that it is equal to the number of cores when the smallest possible number of cores is used. The parallel efficiency, based on 100% for the smallest number of cores, is noted at several points on the plot.

11.4. Parallelisation over the μ (and $v_\parallel$) grid with collisions

If the collision operator is used, the parallelisation over the μ grid is not simply via the fields as derivatives are present. The second-order local derivative for the collision operator uses all the adjacent points in the velocity space (see Section 9.3). Although only 1 point must be exchanged between adjacent processors, any 1 processor may need to perform this exchange between 8 processors. If parallelising over the $v_\parallel$ and μ grids at the same time, this would mean communication between 6 additional processors (when the grid is decomposed over more than 2 processors in each direction) when compared to a run without collisions. Ideally we would like to avoid parallelising over both the $v_\parallel$ and μ grids till other options have been exhausted. Parallelising over the $v_\parallel$ grid, but not the μ grid should be equally expensive with or without collisions, since there is no difference in the grid points communicated. Parallelising instead over just the μ grid, but not the $v_\parallel$ grid would be more expensive with collisions than without, although typically cheaper (when using the default fourth-order finite-differences) than the previous case using only $v_\parallel$.

11.5. Parallel performance on HECToR (Cray XT4)

Here we make a simple assessment of the parallel performance of the code on a Cray XT4. We define the performance as the amount of main time loop computation done by the code per unit time. In the ideal case, a doubling in the number of processor cores used will double the amount of computation done per unit time (or halve the time required to do a fixed number of time steps). We consider a nonlinear problem of fairly typical size that we believe to be reasonably well converged with respect to grid resolution. As mentioned in Section 10.3, we can change the data memory layout to potentially tweak the parallelisation and cache efficiency of the code. We have experienced some variation in performance due to these adjustments. However, we do not yet have enough experience of how to optimally prescribe the layout depending on grid size and parallelisation options, so for the purpose of this test, we keep the layout fixed to the default. We perform a test with a resolution of 83 radial wave vectors, 21 bi-normal modes, 4 species, 16 s grid points, 16 μ grid points and 64 $v_\parallel$ grid points, using fourth-order finite-differences in double precision. Two cases are considered, one using the collision operator and one without. Due to the memory available per processor core, a minimum of 64 processors are required for the former case and 128 for the latter. Using a second-order finite-difference scheme and/or a single precision calculation can reduce this minimum. From the parallel algorithm perspective, the upper limit on the number of parallel processes used for this case is 16 384. Using second-order finite differences would increase this maximum to 65 536 cores. However, only up to 8192 cores were available on the hardware at the time of testing.

Before deciding how to producing a scaling plot for this testcase (such as in Fig. 1), some preliminary investigations on the effects of parallelising in each direction were made. Using the 'trivial' directions alone, the test case parallelises over the minimum 64 processors for the default case. The difference in performance between this parallelisation and parallelising primarily over the parallel velocity grid is fairly small, the 'trivial' directions giving less than 10% better performance than with the $v_\parallel$ direction. Parallelising primarily over the s grid at 64 cores (using the maximum number of cores in the s direction before using the simple direction) yields a performance which is almost identical to that using only the 'trivial' directions. It seems that for 64 processors, the communication needed for the derivatives has little (if any) effect on the parallel performance. To get an idea of the relative expense of parallelising over s and $v_\parallel$, two tests were made by starting with 64 cores (using only the trivial parallelisations), then varying either the number of processors in the $v_\parallel$ direction or the s direction until a total of 512 cores were used. The reduction in efficiency with increasing processors was greater with $v_\parallel$, with the s parallelisation performing around 6% better at 512 cores (at $>95\%$ of the efficiency at 64 processors).

A scaling test was performed for the collisionless case up to 8192 cores. Fig. 1 shows how the performance increases with the number of cores. For the first point at 64 cores, only the 'trivial' directions are used for the parallelisation, since that combination provides the

best performance (but only by around 1%). Since the s direction is the preferred parallelisation direction (least expensive in terms of performance), the number of processors in s doubled between each point on the graph up to 512 cores, where the maximum of 8 processors in s is used. Beyond that point, the more expensive $v_\parallel$ grid is decomposed until 8192 cores are used. At 8192 cores, the code runs at 63% of the efficiency at 64 cores.

Following the discussion in the previous subsections, we should expect that with collisions, parallelisation over the μ direction is more expensive than over the s direction. Therefore, the scaling test for collisions (as shown in Fig. 1) starts at 128 processors fully parallelised over the s direction and over the species, using only 4 processors in μ. Then, the number of processors in μ is doubled, until 512 cores are used. At 1024 cores, it is cheaper to use 32 processors in $v_\parallel$ and only 1 in μ, since this avoids using parallelisation in both those directions together. For the final 2 points on the graph, the number of processors in the $v_\parallel$ and μ are either identical, or there are twice as many in $v_\parallel$. Since these parallelisation combinations give the best performance (considerably better than using the same ones as without collisions), careful consideration should be given to the number of processors used in each direction when using collisions. At 4096 cores the efficiency is 63% of that at 64 cores.

It is perhaps not surprising that in both cases considered we see decreasing performance gains when we begin to parallelise in less favourable directions. Running the tests in single precision would mean less data communicated, so may improve scaling (but at the same time would likely reduce the time between communication, or the time available for overlapping communication with computation). Running with the second-order finite difference scheme would allow double the number of cores in the s direction and may improve the efficiency at a large number of cores. If different test cases are considered, it may well be that more resolution in a particular direction is needed, which would then lend itself to further scaling.

12. Building, installing and running the code

12.1. Compiling the source code

Detailed instructions on building the code from source can be found in the top-level directory file '`README`' and the makefile '`src/makefile`'; here only a brief overview is provided. A Fortran 95 compiler is required to build the code. Most Fortran compilers are able to pre-process the source files containing C pre-processor directives; in the unlikely event that this is not the case, and an appropriate pre-processor is not available, some editing of the source files may be necessary. It is assumed here that the program *make* is available (as it is on most Unix-like systems). If not, then the files can be concatenated into a single file in a certain order to facilitate the build; see the '`README`' file.

You should modify the file '`src/makefile`' to suit your compiler and to set up the linking to any desired MPI and FFT implementations. On successful build you get '`gkw.x`' in the top level.

12.2. Running GKW

Here it is assumed you have the necessary MPI environment set up and running on the machine (consult your MPI library documentation) and that any run time dependencies can be satisfied.

In order to run GKW, the code requires a single input file '`input.dat`' to be present in the run directory. Optionally, a file containing equilibrium geometry information from the CHEASE code can be provided as input. To get started, there are a few example input files found in the '`samples/`' directory, together with corresponding '`README`' files. The file '`samples/general/input.dat`' lists every input switch together with a description of its purpose. '`samples/minimal/input.dat`' is a bare minimum input file containing the essential inputs required for the code to run.

To run the code in parallel, you will probably need to use the program provided by the MPI implementation; this is usually called *mpirun* or *mpiexec*.

GKW will try to select an appropriate parallelisation given the gridsize input but this can be overridden (by setting `N_procs_[sp|mu|vpar|s]` in the gridsize namelist). Nevertheless it is sensible to have in mind a parallelisation setup when you set the gridsize, as the user controls the number of processors the code will be run on and should choose a number that fits with the gridsize. For the largest parallel runs, the user is recommended to manually choose the parallel setup.

The rest of this section describes a basic input file. Many of the input switches are optional and default values will be used if they are not provided. The input variables are provided in Fortran namelists, for example the namelist 'control' contains the main switches for the code. The namelist examples presented here provide you with a basic set needed to run the code. The namelists are sensitive to bad formatting, in which case the code should inform you which namelist has the problem.

GKW tries to catch as many invalid input combinations as possible. Most of the time if you input a set of incompatible switches the code will tell you and abort if necessary. For more minor errors the code will generate a warning that it is ignoring an input parameter and continue. The actual input parameters the code runs with are written to '`input.out`'. By renaming '`input.out`' to '`input.dat`' the code should run without the warnings (although some compilers give badly formatted output, so that the file needs to be fixed by hand first).

12.3. Modes (Table 1)

The spacing and location of the wavevectors is determined by the 'mode' namelist. For `mode_box=.false.` there is only one radial wavevector considered, but the field line can be extended to take more than one poloidal turn by setting `nperiod`. The total number of full poloidal turns is then `2*nperiod - 1`. The toroidal modes are determined by the input `kthrho` which specifies the value(s) of $k_\theta \rho_{\mathrm{ref}}$, see Eq. (57). For `mode_box=.false.` the toroidal modes are given as a list i.e. `kthrho=0.1,0.2,0.3` (the variable `krhomax` is ignored). For `mode_box=.true.`, the field line must have one poloidal turn, i.e. one must set `nperiod=1`. In this case the radial modes are coupled at the end of the field line according to the ballooning shift connected with the magnetic shear. The toroidal modes will be equally spaced from $k_\theta \rho_{\mathrm{ref}} = 0$ to `krhomax` (the `kthrho` input is ignored). The integer p in Eq. (56) is input into the code as `ikxspace` and determines the balance between radial resolution and radial box size.

Table 1
Equivalent formulations example with `mode_box` true and false.

NS	ikxspace	NX	NPERIOD	mode_box
80	N/A	1	3	false
16	1	5	1	true
$n_s.(2n_p-1)$	N/A	1	n_p	false
n_s	1	$2n_p-1$	1	true

12.4. Species

There is no default species data, hence the user must provide some. The species input must satisfy quasi-neutrality. This requires the user to set ratios of densities and charge appropriately. Multiple ion species can be input but only one negatively charged species is allowed. The number of species read in is controlled by the number of species parameter in the 'gridsize' namelist and hence it is possible that later species data you specify will not be read in if this is set incorrectly. The `number_of_species` input variable does not include adiabatic electrons (the species data for the negative charge species will still be read).

12.5. Example GKW input file

```
&CONTROL                    !Namelist read in module CONTROL
 NON_LINEAR = .false.       !Include the non linear terms, computationally expensive fft
 zonal_adiabatic = .false.,!If zonal flows correction included for adiabatic electrons
 order_of_the_scheme = 'fourth_order' !or 'second_order' in space
 DTIM   = 0.02          !Time step size (normalised time)
 NTIME  = 25,           !Number of iterations of NAVERAGE.
 NAVERAGE = 100,        !No of timesteps between re-normalisation, data output
 nlapar = .false.,      !true = keep the electro-magnetic potential. false = electrostatic
 collisions = .false.   !Turn on the collision operator
 disp_par = 1.0         !Dissipation coefficient for parallel derivatives.
 disp_vp  = 0.0         !Dissipation coefficient for parallel velocity space
 max_seconds = 1000.    !Internal program kill time.
 gamatol     = 0.       !tolerance in the linear growth rate for which the code should stop
/
&GRIDSIZE        !Namelist read in module GRID
 NX = 83,        !Number of radial wave vectors - needs to be an odd number for fft
 N_s_grid = 50,  !Number of grid points along the field line
 NPERIOD = 1,    !Integer - the field line makes 2*nperiod - 1 poloidal turns.
 N_mu_grid   = 8,        !Number of magnetic moment grid points
 N_vpar_grid = 32,       !Number of grid points for parallel velocity
 NMOD = 21,              !Number of bi-normal modes (k_zeta)
 number_of_species = 1   !Number of kinetic species.
/
&MODE                !Namelist read in module MODE
 KTHRHO  = 0.5,1.0,  !'Poloidal' wavevector(s) if mode_box=.false.
 mode_box = .true.,  !Determines if there is a 2D grid of modes. If true use nperiod = 1
                     !Also determines if radial modes are coupled at end of field line.
 krhomax = 1.0,      !For mode_box, the maximum 'poloidal' wavevector (k_perp rho_i) used
                     !k_perp evaluated on the low field side of the outboard midplane
                     !rho_i evaluated on the magnetic axis
 ikxspace = 7,       !Spacing between radial modes (resolution against box size)
/
&SPCGENERAL                  !Namelist read in module COMPONENTS
 beta = 0.000,               !Plasma beta (not used if nlapar = .false.)
 adiabatic_electrons = .true. !Kinetic electrons require a smaller timestep
/
&SPECIES       !Namelist read in module COMPONENTS
 MASS  = 1.0,  !Species mass in terms of reference mass
 Z     = 1.0,  !Species charge. If negative, assumed to be the electrons.
 TEMP  = 1.,   !Temperature of species scaled by reference temperature
 DENS  = 1.,   !Density of species scaled by reference density
 rlt   = 8,    !Temperature gradient R/LT
 rln   = 3.5,  !Density gradient R/Ln
 uprim = 0.0,  !Gradient of the toroidal rotation Rref^2 grad Omega / v_thref
/
&SPECIES
 MASS  = 2.72e-4, !Electrons
 Z     = -1.0,    !Only one negatively charged species is allowed.
 TEMP  = 1.0,
 dens  = 1.0,
 rlt   = 7.9,
 rln   = 3.5,
 uprim = 0.0,
/
&GEOM          !Namelist read in module GEOM, s-alpha is default geometry
 shat = 1.0,   !Magnetic shear
 q    = 2.0,   !Safety factor
```

```
 eps  = 0.1    !Radial Coordinate
/
&ROTATION                    !Namelist read in module ROTATION
 VCOR = 0.0                  !Rotation of the plasma vcor = Rref Omega / vthref = \Omega_N
 shear_rate = 0.0            !Normalised shearing rate for the E x B perpendicular shear
 shear_profile  = 'none'     !To include a shear flow, use 'wavevector_remap'
 toroidal_shear = .false.    !Override uprim inputs to be toroidally consistent with shear_rate
/
&COLLISIONS    !Namelist read in module COLLISONOP if collisions = .true.
 rref = 1  !reference major radius used in collision operator
 tref = 1  !reference temperature in units of kev used for the collision operator
 nref = 1  !reference density in units 10^19 m^-3 used for the collision operator
 mom_conservation = .false.     !Use the correction to conserve momentum
 mass_conserve = .true.,        !Forces zero flux boundary on velocity space so no out flow of mass
 freq_override=.false.          !Manually specify ii collision frequency to override ref values
 coll_freq=6.515E-5             !Ion-ion collision frequency used if freq_override=.true.
/
```

12.6. Stability

The timestep required for numerical stability can vary greatly depending on input parameters, and for kinetic electrons will need to be much smaller. For a linear run, there is a time step estimate made at the start of the run and the code will post a warning if the chosen timestep is larger than this. For a nonlinear run, the minimum time step is estimated based on the nonlinear terms, then the time step is adapted as necessary. A minimum time step size can be set in the code (`dt_min`) so that the run stops when the time step becomes too small. The parallel structure of the solution is a good indicator of a the stability of a solution.

For linear runs a stable solution will converge to a constant growth rate. A tolerance (`gamatol`) for the growth rate can be specified in the code input, so that the run ends when it has converged. GKW outputs the global growth rate in `time.dat` so a growth rate spectrum can be generated from a batch of linear runs with `NMOD=1` and `mode_box=.false.`. Typical grid sizes can be seen in the example input files.

13. Diagnostics

The fluxes are calculated as guiding centre fluxes (flux surface averaged) and are given by the equations

$$\Gamma_i^{\psi} = \left\{ \int \mathrm{d}^3\mathbf{v}\, \tilde{\mathbf{v}}_E \cdot \nabla\psi\, \alpha_i f \right\} + \left\{ \int \mathrm{d}^3\mathbf{v}\, \tilde{\mathbf{v}}_{\delta B} \cdot \nabla\psi\, \alpha_i f \right\}, \tag{108}$$

for the electro-static and electro-magnetic (due to magnetic flutter) fluxes, respectively. Here $i = (1, 2, 3)$ with $\alpha_i = (1, mv^2/2, v_\parallel)$, i.e. $i = 1$ is the particle flux, $i = 2$ is the heat flux and $i = 3$ is the parallel momentum flux.

Both the potential and the distribution function are represented as a sum over Fourier modes. Since the flux contains an average over space it will be zero unless the wave vectors of the potential and distribution function add up to be zero. In the representation of modes used in this manuscript this means that a particular wave vector of the representation of f must be combined with the complex conjugate of the same wave vector in the representation of ϕ or $A_\parallel$, i.e.

$$\langle \hat{v}_E f \rangle = 2 \sum_m \mathrm{Re}\big[\langle \hat{\mathbf{v}}_{Em}^{\dagger} \cdot \nabla\psi\, \hat{f}_m \rangle\big], \tag{109}$$

where $\hat{v}_{Em}$ and $\hat{f}_m$ are the Fourier amplitudes of the $E \times B$ velocity and distribution function of the mode m respectively, and the dagger denotes the complex conjugate. Substituting the various definitions one can derive the expression for the normalised fluxes (electro-static and electro-magnetic)

$$\mathcal{I}_i = \sum_m \left\{ 2\pi B \mathcal{E}^{\psi\beta} k_{\beta m} \int \mathrm{d}\mu\, \mathrm{d}v_\parallel\, \alpha_i\, \mathrm{Im}\big[\langle \hat{\phi}_m \rangle^{\dagger} \hat{f}_m\big] \right\} = \sum_m \bar{\mathcal{I}}_i(k_\psi, k_\zeta, s), \tag{110}$$

$$\mathcal{J}_i = -2 \sum_m \left\{ 2\pi B \mathcal{E}^{\psi\beta} k_{\beta m} \int \mathrm{d}\mu\, \mathrm{d}v_\parallel\, \alpha_i v_R v_\parallel\, \mathrm{Im}\big[\langle \hat{A}_{\parallel m} \rangle^{\dagger} \hat{f}_m\big] \right\} = \sum_m \bar{\mathcal{J}}_i(k_\psi, k_\zeta, s), \tag{111}$$

where $\sum_m \{\} = \sum_{k_\zeta} \sum_{k_\psi} \sum_s$ is a sum over all modes and along the full s length of the flux tube.

The fluxes in physical units can be obtained from

$$R_{\mathrm{ref}} \Gamma_s^{\psi} = n_s \rho_*^2 v_{\mathrm{thref}} (\mathcal{I}_1 + \mathcal{J}_1), \tag{112}$$

$$R_{\mathrm{ref}} Q_s^{\psi} = n_s T_s \rho_*^2 v_{\mathrm{thref}} (\mathcal{I}_2 + \mathcal{J}_2), \tag{113}$$

$$R_{\mathrm{ref}} \Gamma_{\phi s}^{\psi} = m_s n_s v_{ths} \rho_*^2 v_{\mathrm{thref}} (\mathcal{I}_3 + \mathcal{J}_3). \tag{114}$$

For a linear run, the dominant mode growth rate is calculated as

$$\gamma^{\max}(t) = \ln\left[\frac{\sqrt{\sum_m |\hat{\phi}(t)|^2}}{\sqrt{\sum_m |\hat{\phi}(t - \Delta t)|^2}} \right] \Big/ \Delta t. \tag{115}$$

Table 2
Time trace diagnostics output by GKW.

Filename	Description	Quantity	Cols, (rows)
time.dat	Dominant linear mode growth rate	$t_N, \gamma_N^{\max}, \omega_N$	3
fluxes.dat	Total v_E fluxes ($i=1,2,3$) by species	$\mathcal{I}_i = \sum_{k_\zeta}\sum_{k_\psi}\sum_s \bar{\mathcal{I}}_i$	$3N_{sp}$
fluxes_em.dat	Total δB fluxes ($i=1,2,3$) by species	$\mathcal{J}_i = \sum_{k_\zeta}\sum_{k_\psi}\sum_s \bar{\mathcal{J}}_i$	$3N_{sp}$
kyspec	Bi-normal mode potential spectrum	$\sum_{k_\psi}\sum_s \lvert\hat{\phi}(k_\psi,k_\zeta,s)\rvert^2$	N_{mod}
kxspec	Radial mode potential spectrum	$\sum_{k_\zeta}\sum_s \lvert\hat{\phi}(k_\psi,k_\zeta,s)\rvert^2$	N_x
p/e/v_flux_spectra	Bi-normal spectral v_E flux for each species	$\sum_{k_\psi}\sum_s \bar{\mathcal{I}}_i(k_\psi,k_\zeta,s)$	$N_{\mathrm{mod}}N_{sp}$
p/e/v_flux_xspec	Radial spectral v_E fluxes for each species	$\sum_{k_\zeta}\sum_s \bar{\mathcal{I}}_i(k_\psi,k_\zeta,s)$	$N_{\mathrm{mod}}N_{sp}$
parallel_phi.dat	Parallel potential	$\sum_{k_\zeta}\sum_{k_\psi} \lvert\hat{\phi}(k_\psi,k_\zeta,s)\rvert^2$	N_s
spc#	2D potential spectrum slice # by timestep	$\lvert\hat{\phi}(k_\psi,k_\zeta,s=0)\rvert$	$N_x, (N_{\mathrm{mod}})$
phi#	2D potential slice # by timestep	$\phi(\psi,\zeta,s=0)$	$M_x, (M_{\mathrm{mod}})$

Table 3
Initial grid diagnostics output by GKW. The grid outputs are scaled so that ψ and ζ coordinate scales can be compared directly on the input scale of $k_\perp\rho_{\mathrm{ref}}$ (Eq. (57)).

Filename	Description	Quantities	Cols, rows
krho	Bi-normal wavevector grid	$(q/2\pi\psi)k_\zeta\rho_{\mathrm{ref}} = k_\perp(k_\psi=0,s=0)\rho_{\mathrm{ref}}$	N_x, N_{mod}
kxrh	Radial wavevector grid	$k_\psi\rho_{\mathrm{ref}}$	N_x, N_{mod}
yphi	Real space bi-normal grid	$2\pi\psi(\zeta-\zeta_{\min})/q\rho_{\mathrm{ref}}$	M_x, M_{mod}
xphi	Real space radial grid	$(\psi-\psi_{\min})/\rho_{\mathrm{ref}}$	M_x, M_{mod}
parfun.dat	Perpendicular wavevector	$k_\perp\rho_{\mathrm{ref}}$	$2, N_sN_{\mathrm{mod}}N_x$
par.dat	Curvature and Coriolis functions	$k_\perp\rho_{\mathrm{ref}}$, $\mathcal{D}^\psi k_\psi + \mathcal{D}^\zeta k_\zeta$, $\mathcal{H}^\psi k_\psi + \mathcal{H}^\zeta k_\zeta$,	$3, N_sN_{\mathrm{mod}}N_x$
geom.dat	Geometry tensors	Labelled in file	N/A
kx_connect	Parallel boundary connections	Labelled in file	N/A

The mode frequency (not used in nonlinear runs) is calculated as

$$\omega(t) = \left[\mathrm{Arg}\left(\sum_s \hat{\phi}(t)\right) - \mathrm{Arg}\left(\sum_s \hat{\phi}(t-\Delta t)\right)\right] \Big/ \Delta t. \tag{116}$$

Positive values of $\omega(t)$ indicate propagation in the direction opposite to $\nabla\zeta$ (which for $s_j = 1$ is in the ion diamagnetic drift direction). Note that in an electromagnetic run, each sum in the growth rate and mode frequency expressions above becomes a double sum, the additional sum being over the two fields $\widehat{\phi}$ and $\widehat{A}_\parallel$.

Although GKW uses the thermal velocity ($\sqrt{2T/m}$) for normalisation, it is more common in the literature to use the gyro-Bohm units of sound speed $c_s = \sqrt{T_e/m_i}$ and the corresponding Larmor radius $\rho_s = c_s/\omega_{ci}$, where $\omega_{ci} = eB/m_i$ is the ion cyclotron frequency, and the index i (e) corresponds to the ions (electrons). In the case where $T_e = T_{\mathrm{ref}}$ and $m_i = m_{\mathrm{ref}}$, the sound speed is $c_s = v_{\mathrm{thref}}/\sqrt{2}$, the Larmor radius is $\rho_s = \rho_{\mathrm{ref}}/\sqrt{2}$, and the GKW wavevectors have $k_\perp\rho_i = k_\perp\rho_s/\sqrt{2}$. The time normalisation converts as

$$t_N = \frac{v_{\mathrm{thref}}}{R_{\mathrm{ref}}}t = \sqrt{2}\frac{c_s}{R_{\mathrm{ref}}} = \left(\frac{\sqrt{2}a}{R_{\mathrm{ref}}}\right)\left(\frac{c_s}{a}\right)t, \tag{117}$$

and the heat fluxes convert to a gyro-Bohm diffusivity as

$$\chi_i^N = \frac{\Gamma_i^N}{R/L_T} = \frac{R_{\mathrm{ref}}}{\rho_{\mathrm{ref}}^2 v_{\mathrm{thref}}}\chi_i = \left(\frac{1}{2\sqrt{2}}\frac{R_{\mathrm{ref}}}{a}\right)\left(\frac{a}{\rho_s^2 c_s}\right)\chi_i. \tag{118}$$

Note that N in the two previous expressions is used to denote a normalised quantity, as used in most of this paper, and the equivalent quantity without the N is a dimensional one. A summary of GKW diagnostic outputs is given in Tables 2 and 3.

14. Benchmarks

In this section we describe a set of benchmarks. The benchmark cases have been chosen to maximise the effect of each of the different implemented terms as much as possible. Together they give evidence for the correct implementation of the entire model. Linear benchmarks have been made against the GS2 code which is well established. Indeed the model equations are equivalent, with the exception that GKW at present does not allow for compressional effects in the magnetic field, while GS2 does, and that GKW includes the effect of a rigid body toroidal rotation, while GS2 does not. The implementation is to some extent similar; both use a spectral approach for the plane perpendicular to the magnetic field, and finite differences for the other directions. The main difference lies in the velocity space discretisation where GKW uses $(v_\parallel, \mu)$ coordinates, whereas GS2 uses pitch angle and energy. Also the finite differences scheme used for GKW in the benchmarks is fourth order while GS2 has a second-order scheme. In this section all quantities have been converted to use gyro-Bohm units.

14.1. Linear benchmarks

The standard benchmark for linear problems is the growth rate of the Ion Temperature Gradient mode (ITG) as a function of the poloidal wave vector $k_\theta\rho_s$ for the so-called Cyclone base case [36] ($q = 1.4$, $\hat{s} = 0.78$, $\epsilon = 0.19$, $R/L_T = 6.9$, $R/L_N = 2.2$, $T_e/T_i = 1$, electrostatic (zero $A_\parallel$), and adiabatic electrons). The growth rate for this case is shown in the right panel of Fig. 2 as a function of $k_\theta\rho_s$ for various values of the normalised ion temperature gradient ($R/L_T = 6.9$, 8.28, 10.35, 12.44, and 15.18) calculated with both GKW and GS2.

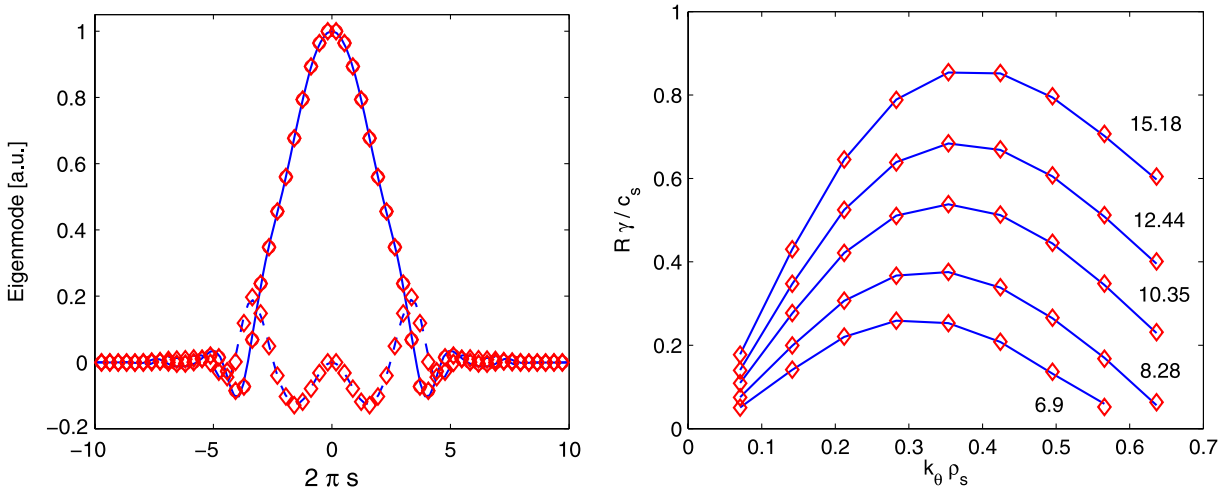


Fig. 2. Left: Mode structure of the Cyclone base case. The full line is the real part and the dotted line is the imaginary part of the potential calculated with GS2. The corresponding results of GKW are given by the diamonds. Right: Growth rate as a function of $k_\theta \rho_s$ for various values of R/L_{Ti} (numbers given in the figure). The full lines give the results of GS2 while the diamonds are the corresponding results of GKW.

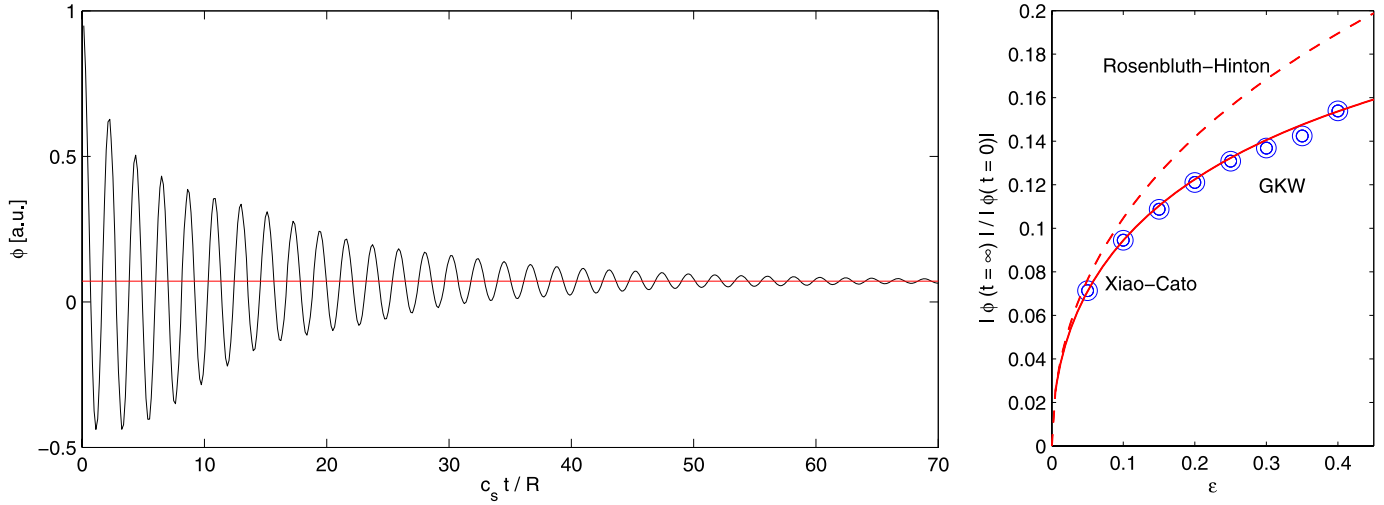


Fig. 3. Zonal flow benchmark. Left: The potential perturbation (normalised to the potential at $t = 0$) as a function of normalised time, for $q = 1.3$, $\epsilon = 0.05$. The horizontal line gives the residual (0.0711) as predicted in Ref. [52]. For this value of ϵ a numerical residual of 0.0713 is obtained, i.e. in agreement with the analytic result to within 0.3%. Right: Residual $\phi(t = \infty)/\phi(0)$ as a function of ϵ for $q = 1.3$. The circles are the result of GKW, the dotted line is the Rosenbluth–Hinton result [51] and the full line gives the analytic result of Xiao and Catto [52].

As can be seen from the figure, good agreement is obtained. The left panel, furthermore, shows the comparison of the potential as a function of the coordinate along the field line (s) for both codes.

For nonlinear runs the proper response of the zonal flows is of utmost importance. The standard benchmark [7,13,50] that addresses the physics of the zonal flow/geo-acoustic mode is the Rosenbluth–Hinton test which was described analytically in Ref. [51]. In this benchmark the initial condition is an ion density perturbation with a finite (small) radial wave vector, and no dependence on either s or ζ. The adiabatic electron response is used, keeping the correction due to the flux surface average of the potential. The density perturbation generates a potential perturbation and excites the so-called geo-acoustic mode. This mode is damped and a small residual poloidal flow remains. Fig. 3 gives the potential perturbation as a function of time. The relevant parameters used for this benchmark are $q = 1.3$, $\epsilon = 0.05$, $k_\psi \rho_s = 0.02$, electrostatic, collisionless. Rather large grid sizes ($N_s = 128$ and $N_{v_\parallel} = 128$) are used to avoid the recurrence problem [13]. The residual potential is given by the equation

$$\frac{\phi(t=\infty)}{\phi(t=0)} = \frac{1}{1+q^2\Theta/\epsilon^2} \quad \text{with } \Theta = 1.6\epsilon^{3/2} + 0.5\epsilon^2 + 0.36\epsilon^{5/2}. \tag{119}$$

The original derivation of Ref. [51] is accurate to lowest order in ϵ only, i.e. it retained only the first term ($1.6\epsilon^{3/2}$) in Θ. The higher-order terms have been calculated in Ref. [52]. For the parameters of the simulation shown in the left panel the residue is $\phi(t = \infty)/\phi(t = 0) = 0.0713$ smaller than the result of Ref. [51] (0.0764) but in very good agreement with the results of Ref. [52] (0.0710). The right panel shows the residue as a function of ϵ. Good agreement with the analytic theory is obtained provided the finite ϵ effects are kept.

The Cyclone base case with adiabatic electrons does not check the correct implementation of the kinetic electron response, and in general is not very sensitive to the effects associated with trapped particles. The physics effects associated with kinetic electrons can be well benchmarked by considering a Trapped Electron Mode. A benchmark with GS2 is shown on the left in Fig. 4. The parameters are taken

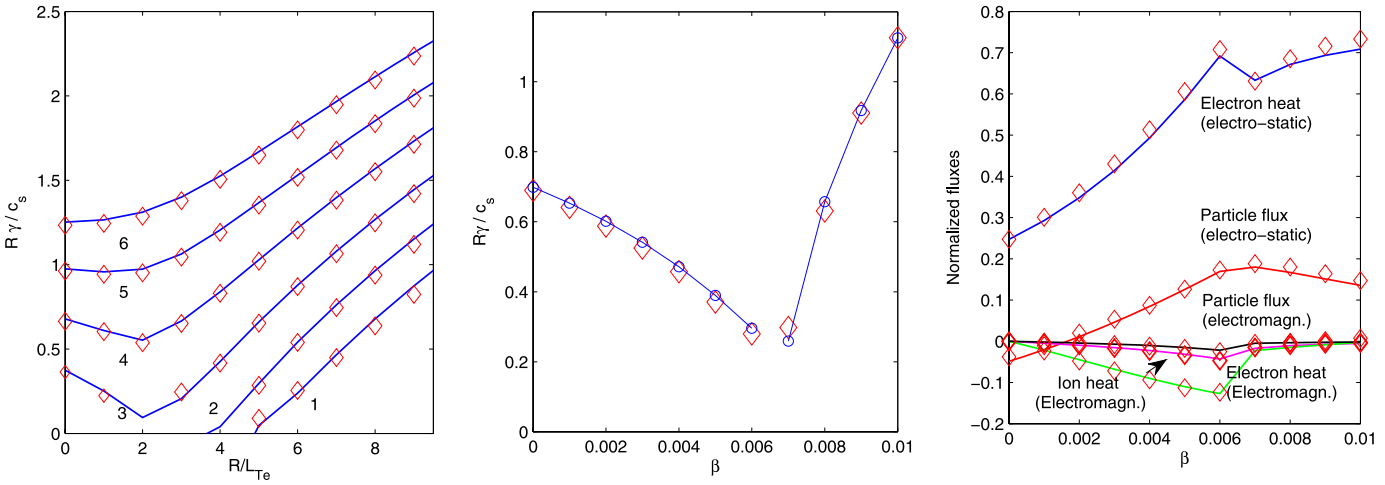


Fig. 4. Left: Benchmark of a trapped electron mode case. Shown is the normalised growth rate $R\gamma/c_s$ as a function of R/L_{Te} for various values of R/L_N indicated in the figure. Blue lines are the results of GS2 while red diamonds give the results of GKW. Middle: A benchmark of the growth rates of the ITG/Kinetic ballooning mode versus the plasma beta. Blue lines and circles are results from GS2 while red diamonds give the results of GKW. Right: The fluxes, normalised to the ion heat flux as a function of beta. Lines give the results of GS2, while the diamonds give the results of GKW. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

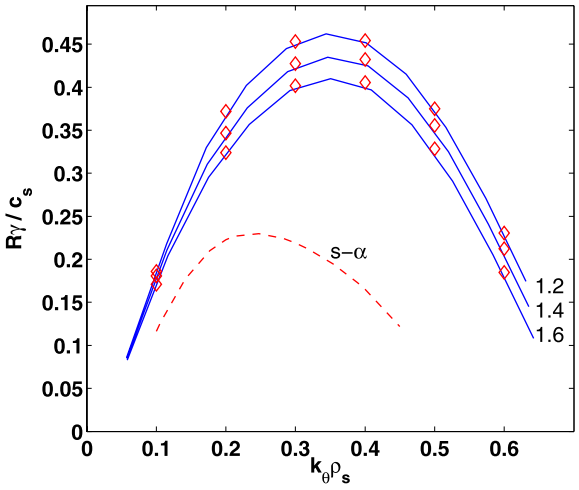


Fig. 5. Benchmark of the geometry treatment. The normalised growth rate $R\gamma/c_s$ is shown as a function of $k_\theta\rho_s$ for various values of the last closed flux surface elongation κ indicated in the figure. Blue lines are the results of GS2 while red diamonds give the results of GKW. The results obtained for the simplified '$s-\alpha$' equilibrium with GKW are indicated by the red dashed line. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

from a discharge in the ASDEX Upgrade tokamak [53]: $\hat{s}=1.07$, $q=1.57$, $\epsilon=0.177$, $T_e/T_i=3$, $R/L_{Ti}=0$, electro-static, collisionless, and with an ion to electron mass ratio of a Deuterium plasma. The values of the density gradient as well as the temperature gradient are scanned. The agreement for the growth rates is good.

Finite beta effects are tested for an ITG case increasing the plasma beta. The middle panel of Fig. 4 shows the normalised growth rate as a function of the electron beta $\beta=n_eT_e/(B^2/2\mu_0)$ for the Waltz standard case $R/L_{Ti}=R/L_{Te}=9$, $R/L_N=3$, $q=2$, $\hat{s}=1$, $\epsilon=0.166$, $T_e/T_i=1$, and with a mass ratio of a Deuterium plasma. The ITG is stabilised by the finite beta effects and, at sufficient high values of beta, a kinetic ballooning mode is destabilised. The agreement for the growth rate between GKW and GS2 is again good. The electromagnetic case also provides for a good benchmark of the calculation of the fluxes, since all fluxes (also the ones due to the magnetic flutter) are nonzero in this case. The right panel of Fig. 4 shows the comparison of the fluxes calculated by GS2 and GKW. Since for a linear problem the amplitude of the mode and, therefore, the magnitude of the fluxes is arbitrary, all fluxes have been normalised to the ion heat flux. This allows for a straightforward comparison between the codes. All calculated fluxes are again in good agreement.

The collisions have been benchmarked against GS2 in Ref. [21], and good agreement is obtained for the particle flux as a function of collisionality. Also the $E\times B$ shear stabilisation has been successfully benchmarked against GYRO results and is reported in Ref. [54].

The treatment of arbitrary toroidal geometry has been benchmarked against GS2 by performing an elongation scan. The parameters considered are $R/L_{Ti}=R/L_{Te}=8.9$, $R/L_N=2.85$, $q=1.42$, $\hat{s}=1.25$, $\epsilon=0.182$, $T_e/T_i=1$ for deuterium ions and adiabatic electrons. The elongation of the last closed flux surface is varied from $\kappa=1.2$ to $\kappa=1.6$ and the corresponding MHD equilibrium is calculated using the CHEASE code [33]. This equilibrium is then used for the calculation of the metric tensors both in GKW and GS2. The results for the linear growth rate as a function of $k_\theta\rho_s$ are shown in Fig. 5 showing a good agreement between the two codes. Note that the results are significantly different from the ones obtained with the simplified '$s-\alpha$' equilibrium shown by the dashed line (obtained with GKW).

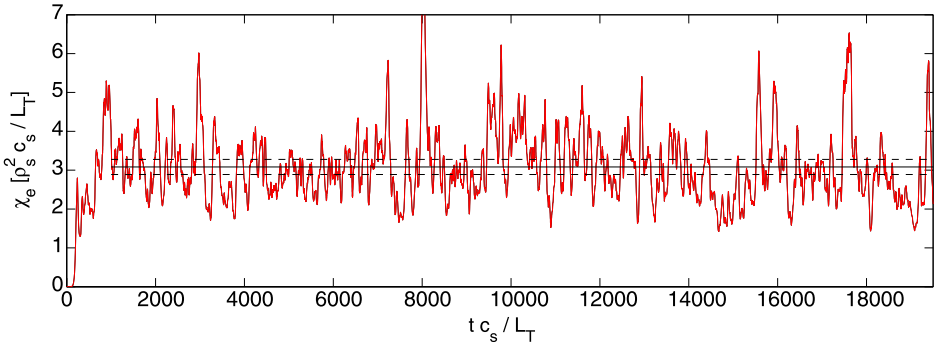


Fig. 6. GKW results of the electron heat flux expressed as a heat conduction coefficient (normalised to $\rho_e^2 c_e / L_T$ with $c_e = \sqrt{T_e/m_e}$) as a function of normalised time ($t_{NN} = t c_e / L_T$) for the Nevins ETG benchmark.

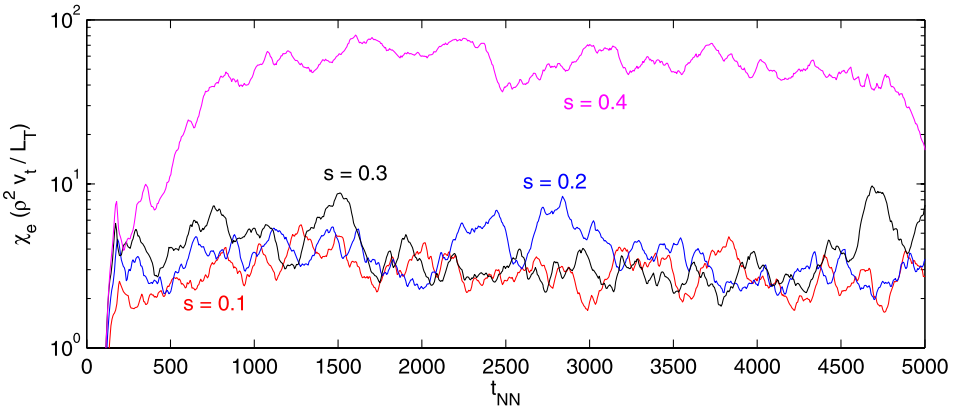


Fig. 7. GKW results of the electron heat conduction coefficient for different values of the magnetic shear.

14.2. Nonlinear benchmarks

For the nonlinear solution there are only two benchmark cases for which several codes have been compared: the cyclone base case, and the Nevins ETG benchmark. GKW has been benchmarked against the European gyrokinetic codes for the Cyclone case in Ref. [37]. Below we therefore describe the alternative ETG benchmark of Nevins [55]. The parameters for this case are the same as those of the Cyclone base case ($R/L_T = 6.9$, $R/L_N = 2.2$, $q = 1.4$, $T_e/T_i = 1$, $\epsilon = 0.18$) with only the magnetic shear changed to $\hat{s} = 0.1$. For this case the electron dynamics is simulated while the ions are assumed adiabatic. The latter response does not include the flux surface average of the electro-static potential, i.e. the response is proportional to $n_i\phi/T_i$. To obtain the same results as published in Ref. [55] we have used (and must use) the same number of toroidal modes $N_{\mathrm{mod}} = 8$, and mode spacing $(k_\perp \rho_s)_{\max} = 0.69$. The radial grid spacing is $\Delta(k_r\rho_s) = 0.0619$ with $N_x = 41$ (both positive and negative wave vectors counted). Finally, $N_\mu = 8$, $N_{v_\parallel} = 16$, and $N_s = 16$.

Fig. 6 shows the normalised electron heat flux as a function of the normalised time. Averaging over the time interval ($1000 < t c_e/L_T < 19\,500$), GKW predicts a heat diffusion coefficient $\chi_e = 3.08 \pm 0.19$ in good agreement with the value $\chi_e = 2.95 \pm 0.15$ reported in Ref. [55]. The ETG case is sensitive to the magnetic shear. For values $\hat{s} > 0.3$ no convergence is found in the simulations reported in Ref. [55]. GKW reproduces this result as shown in Fig. 7.

15. Conclusion

A new nonlinear gyro-kinetic flux tube code (GKW) for the simulation of micro instabilities and turbulence in magnetic confinement plasmas has been presented. The code has the following features: local limit, kinetic electrons, electromagnetic effects, collisions, full general geometry with a coupling to a MHD equilibrium code, and $E \times B$ shearing. In addition the physics of plasma rotation has been implemented through a formulation of the equations in the co-moving system. The code passes all the necessary benchmarks and is optimised to run on 8192+ processors. These features together, make GKW a powerful tool for the study of micro-instabilities and turbulence in magnetised plasmas.

Acknowledgements

The authors want to thank W. Dorland and M. Kotschenreuther for making the GS2 code available, as well as A. Marinoni for his contribution to the geometry benchmark. One of the authors (AGP) would also like to thank Bruce D. Scott for his tutorials and insights into the subject which he has been able to follow over many years.

References

[1] A.G. Peeters, D. Strintzi, Phys. Plasmas 11 (2004) 3748.

[2] S.E. Parker, W.W. Lee, R.A. Sandtoro, Phys. Rev. Lett. 71 (1993) 2042.
[3] Z. Lin, T.S. Hahm, W.W. Lee, et al., Science 281 (1998) 1835.
[4] Y. Idomura, S. Tokuda, Y. Kishimoto, Nucl. Fusion 43 (2003) 234.
[5] J.A. Heikkinen, J.S. Janhunen, T.P. Kiviniemi, et al., Contr. Plasma Phys. 44 (2004) 13.
[6] A. Bottino, A.G. Peeters, R. Hatzky, et al., Phys. Plasmas 14 (2007) 010701.
[7] S. Jolliet, A. Bottino, P. Angelino, et al., Comput. Phys. Comm. 177 (2007) 409.
[8] V. Grandgirard, M. Brunetti, P. Bertrand, et al., J. Comput. Phys. 217 (2006) 395.
[9] M. Kotschenreuther, G. Rewoldt, W.M. Tang, Comput. Phys. Comm. 88 (1995) 128.
[10] W. Dorland, F. Jenko, M. Kotschenreuther, et al., Phys. Rev. Lett. 85 (2000) 5579.
[11] F. Jenko, Comput. Phys. Comm. 125 (2000) 196.
[12] T. Dannert, F. Jenko, Phys. Plasmas 12 (2005) 072309.
[13] J. Candy, R.E. Waltz, J. Comput. Phys. 186 (2003) 545.
[14] B.D. Scott, in: 22nd IAEA Fusion Energy Conference (Geneva, Swiss), Paper TH/P8-13, http://www-pub.iaea.org/MTCD/Meetings/FEC2008/th_p8-13.pdf.
[15] A.G. Peeters, C. Angioni, Phys. Plasmas 12 (2005) 072515.
[16] A.G. Peeters, C. Angioni, A. Bottino, et al., Plasma Phys. Contr. Fusion 48 (2006) B413.
[17] A.G. Peeters, C. Angioni, D. Strintzi, Phys. Rev. Lett. 98 (2007) 265003.
[18] A.G. Peeters, C. Angioni, D. Strintzi, Phys. Plasmas 16 (2009) 034703.
[19] A.G. Peeters, D. Strintzi, C. Angioni, et al., Phys. Plasmas 16 (2009) 042310.
[20] Y. Camenen, A.G. Peeters, C. Angioni, Phys. Rev. Lett. 102 (2009) 125001.
[21] A.G. Peeters, C. Angioni, Y. Camenen, et al., Phys. Plasmas 16 (2009) 062311.
[22] Y. Camenen, A.G. Peeters, C. Angioni, et al., Phys. Plasmas 16 (2009) 062501.
[23] Y. Camenen, A.G. Peeters, C. Angioni, et al., Phys. Plasmas 16 (2009) 012503.
[24] R.G. Littlejohn, J. Plasma Phys. 29 (1983) 111.
[25] T.S. Hahm, Phys. Fluids 31 (1988) 043207.
[26] A. Brizard, Phys. Plasmas 41 (1988) 541.
[27] H. Sugama, Phys. Plasmas 7 (2000) 466.
[28] A. Brizard, Rev. Modern Phys. 79 (2007) 421.
[29] A.J. Brizard, Phys. Plasmas 2 (1995) 459.
[30] S. Hamada, Kakuyugo Kenkyu 1 (1958) 542.
[31] B.D. Scott, Phys. Plasmas 5 (1998) 2334.
[32] B.D. Scott, Phys. Plasmas 8 (2001) 447.
[33] H. Lütjens, A. Bondeson, O. Sauter, Comput. Phys. Comm. 97 (1996) 219.
[34] J.W. Connor, R.J. Hastie, J.B. Taylor, Phys. Rev. Let. 40 (1978) 396.
[35] C.F.F. Karney, Comput. Phys. Rep. 4 (1986) 183.
[36] A.M. Dimits, Phys. Plasmas 7 (2000) 969.
[37] G.L. Falchetto, B.D. Scott, P. Angelino, A. Bottino, T. Dannert, V. Grandgirard, S. Janhunen, F. Jenko, S. Jolliet, A. Kendl, B.F. McMillan, V. Naulin, A.H. Nielsen, M. Ottaviani, A.G. Peeters, M.J. Pueschel, D. Reiser, T.T. Ribeiro, M. Romanelli, Plasma Phys. Controll. Fusion 50 (2008) 124015.
[38] M. Frigo, S.G. Johnson, Proc. IEEE 93 (2005) 216.
[39] T.S. Hahm, K.H. Burrell, Phys. Plasmas 2 (1995) 1648.
[40] K.H. Burrell, Phys. Plasmas 4 (1997) 1499.
[41] R.E. Waltz, G.D. Kerbel, J. Milovich, Phys. Plasmas 1 (1994) 2229.
[42] R.L. Miller, R.E. Waltz, Phys. Plasmas 1 (1994) 2835.
[43] F. Baron, PhD in physics, Univ. Pierre et Marie Curie, Paris, France, 1982.
[44] U. Schumann, Algorithms for Direct Numerical Simulation of Shear-Periodic Turbulence, in: Lecture Notes in Physics, vol. 218, Springer-Verlag, Berlin, 1985, p. 492.
[45] T. Gerz, U. Schumann, S.E. Elghobashi, J. Fluid Mech. Digital Arch. 200 (1989) 563.
[46] R.S. Rogallo, Numerical experiments in homogeneous turbulence, NASA STI/Recon Technical Report N, 1981, 81:31508.
[47] T.A. Zang, Appl. Numer. Math. 7 (1991) 27.
[48] Alain Pumir, Phys. Fluids 8 (1996) 3112.
[49] G.W. Hammett, W. Dorland, N.F. Loureiro, T. Tatsuno, Implementation of large scale $E \times B$ shear flow in the GS2 gyrokinetic turbulence code, in: APS Meeting Abstracts, 2006, p. 1136.
[50] F. Jenko, T. Dannert, C. Angioni, Plasma Phys. Contr. Fusion 47 (2005) B195.
[51] F.L. Hinton, M.N. Rosenbluth, Phys. Rev. Lett. 80 (1998) 724.
[52] Y. Xiao, P.J. Catto, Phys. Plasmas 13 (2006) 102311.
[53] A.G. Peeters, C. Angioni, M. Apostoliceanu, et al., Phys. Plasmas 12 (2005) 022505.
[54] F.J. Casson, A.G. Peeters, Y. Camenen, et al., Anomalous parallel momentum transport due to $E \times B$ flow shear in a tokamak plasma, Phys. Plasmas (2009), submitted for publication.
[55] W. Nevins, et al., Phys. Plasmas 13 (2006) 122306.
[56] A. Arakawa, J. Comput. Phys. 1 (1966) 119.