A semi-Lagrangian code for nonlinear global simulations of electrostatic drift-kinetic ITG modes

M. Brunetti [a,*], V. Grandgirard [b], O. Sauter [a], J. Vaclavik [a] and L. Villard [a]

[a] *Centre de Recherches en Physique des Plasmas, Association EURATOM - Confédération Suisse, EPFL, 1015 Lausanne, Switzerland.*

[b] *DRFC Association Euratom-CEA CEA Cadarache, 19108 St Paul-lez-Durance, France.*

Abstract

A semi-Lagrangian code for the solution of the electrostatic drift-kinetic equations in straight cylinder configuration is presented. The code, CYGNE, is part of a project with the long term aim of studying microturbulence in fusion devices. The code has been constructed in such a way as to preserve a good control of the constants of motion, possessed by the drift-kinetic equations, until the nonlinear saturation of the ion-temperature-gradient modes occurs. Studies of convergence with phase space resolution and time-step are presented and discussed. The code is benchmarked against electrostatic Particle-in-Cell codes.

Key words: semi-Lagrangian, electrostatic, ion-temperature-gradient, drift-kinetic, conservation laws

PACS: 52.65, 52.35.Q, 52.25.F, 52.55

1 Introduction

Energy and particles losses observed experimentally in tokamaks are usually well above those predicted by collisional neoclassical rates. Even when magnetohydrodynamic modes have been stabilized, the free energy associated with the spatial variations of macroscopic quantities, such as mean density and temperature, gives rise to low frequency micro-instabilities which are commonly considered as leading candidates to explain anomalous transport in fusion devices. The theoretical model for studying the evolution of micro-instabilities in plasmas is provided by the nonlinear energy-conserving gyrokinetic Maxwell-Vlasov equations [1,2]. Since the solution of the gyrokinetic equations constitutes a computational physics 'Grand Challenge', numerical simulations rely in general on simplifying approximations, such as electrostatic plasmas or radially local calculations, which will be relaxed when sufficient supercomputer capabilities will be available.

Gyrokinetic equations have been solved numerically using algorithms based on Lagrangian and Eulerian descriptions of phase space dynamics. Particle in Cell (PIC) simulations [3–9] use the Lagrangian point of view since they are based on the numerical solution of 'superparticles' trajectories. The main drawback of PIC codes is that sampling the physical particles with 'superparticles' leads to the introduction of statistical noise which can be reduced but not eliminated by optimized loading techniques [10]. On the contrary, Eulerian codes [11–13], which evolve the distribution function over fixed points in phase space, are not affected by statistical noise. Moreover, by construction, the Eulerian

* Corresponding author: ph. +41-(0)21-693 65 29, fax: +41-(0)21-693 51 76
Email address: maura.brunetti@epfl.ch (M. Brunetti).

approach allows one to get information on the particle distribution function in all phase space coordinates on a structured grid, while the same is not easily reconstructed by PIC codes. The drawback of Eulerian codes is that they tend to be computationally expensive due to the use of this fixed grid and they use much smaller time-step than Lagrangian schemes. A third technique, commonly used by the meteorology community for weather forecast models, combines the Eulerian and the Lagrangian points of view and for this reason can be called 'semi-Lagrangian method' [14,15]: the distribution function is computed on a fixed grid (as in the Eulerian approach) and it is evolved (as in the Lagrangian approach) by tracing back in time the trajectories of a different set of particles at each time-step, namely the set of particles located at the grid points.

In this work, we present a semi-Lagrangian code for the solution of the nonlinear global electrostatic drift-kinetic equations in a straight cylinder configuration. Within this framework, the positions of the particles and of the guiding centers are the same, finite Larmor effects being neglected, the electrons are treated as adiabatic, and electromagnetic fluctuations are not included. The conservation properties of the physical model provide us with a powerful tool for monitoring the quality of the simulations. The code CYGNE (which is the acronym for CYlindrical Global Nonlinear Electrostatic, and it is also the French word for 'swan') has been developed in such a way as to preserve to a good accuracy the conservation of the constants of motion until the micro-instability driven by the ion temperature gradient saturates.

These results have been achieved using numerical tools which upgrade the ones described in Ref. [16,17] in several aspects. The Bulirsch-Stoer method (see [18] and Section 3) is used now for the calculation of the characteristics.

Non-equidistant meshes have been introduced in r and $v_\parallel$ directions in order to increase resolution where the main physics takes place while still preserving a reasonable CPU time per processor. Diagnostics have been improved in order to avoid spurious results related to unsufficient phase space resolution and negative values in the distribution function.

The code has been benchmarked against electrostatic PIC codes. The numerical schemes of CYGNE have been implemented having in mind the future developments aiming at solving gyrokinetic equations in tokamak geometry.

The paper is organized as follows. In Section 2, the governing equations and their conservation properties are described. The numerical implementation used for the results presented in this paper are discussed in Section 3. The validation of the code is reported in Section 4, and conclusions are given in Section 5. Details of the numerical implementation and results obtained with a three-time-level integration scheme will be presented in another paper [19].

2 Physical model

We consider a periodic cylindrical plasma of radius a and length L, which can be viewed as the limiting case of a stretched torus. The plasma is confined by a uniform magnetic field of the form $\vec{B} = B_0\ \vec{e}_z$, where $\vec{e}_z$ is the unit vector along the z direction. In the following, $\parallel$ denotes the direction parallel and $\perp$ the direction perpendicular to $\vec{B}$.

When finite Larmor effects are neglected, the distribution function f reduces to a four-dimensional function, the (conserved) magnetic moment $\mu = v_\perp^2/(2B_0)$ being strictly zero in this approximation. The equations of motion for the

guiding centers in cylindrical coordinates (r, θ, z) are given by

$$\dot{\vec{R}} = v_\parallel \vec{e}_z + \frac{\vec{e}_z \wedge \vec{\nabla}\phi}{B_0} = v_\parallel \vec{e}_z + \vec{v}_{GC} \tag{1}$$

$$\dot{v}_\parallel = -\frac{q_i}{m_i}\ \vec{e}_z \cdot \vec{\nabla}\phi = \frac{q_i}{m_i}\ E_z \tag{2}$$

where $q_i = Z_i e$ and m_i are the ion charge and mass, respectively, and we have used $\vec{E} = -\vec{\nabla}\phi$, $\vec{E}$ being the electric field. The drift-kinetic Vlasov equation for the ion distribution function is

$$\frac{\partial f}{\partial t} + \vec{v}_{GC} \cdot \vec{\nabla}_\perp f + v_\parallel \frac{\partial f}{\partial z} + \dot{v}_\parallel \frac{\partial f}{\partial v_\parallel} = 0 \tag{3}$$

where $\vec{\nabla}_\perp = (\partial/\partial r, (1/r)\partial/\partial\theta, 0)$. Assuming adiabatic electrons bound to the field lines and including the linearized polarization-drift term, the quasi-neutrality equation reads:

$$-\vec{\nabla}_\perp \cdot \left[\frac{n_0(r)}{B_0 \Omega_i}\vec{\nabla}_\perp \phi\right] + \frac{e n_0(r)}{T_e(r)}(\phi - \bar{\phi}) = \int f(r, \theta, z, v_\parallel, t)\ dv_\parallel - n_0(r) \tag{4}$$

where $\Omega_i = q_i B_0/m_i$ is the ion cyclotron frequency, n_0 and T_e are the initial density and electron temperature profiles, and $\bar{\phi}$ is the average of the electrostatic potential along the magnetic field lines given by

$$\bar{\phi}(r, \theta, t) = \frac{1}{L}\int_0^L \phi(r, \theta, z, t)\ dz \tag{5}$$

The Vlasov equation (3) is Hamiltonian and can be put into the form

$$\frac{df}{dt} \equiv \frac{\partial f}{\partial t} + \{H, f\} = 0 \tag{6}$$

where $\{,\}$ are the Poisson brackets and H is the Hamiltonian, $H = m_i v_\parallel^2/2 + q_i\phi$. This equation expresses the fact that the distribution function f is a Lagrangian invariant, i.e., it is constant along the particle trajectories. As a consequence, the integral over the entire phase space of the distribution

function is a constant, as well as the integral of any arbitrary (smooth) function of f, $C(f)$, since

$$\frac{dC(f)}{dt} \equiv \frac{\partial C(f)}{\partial t} + \{H, C(f)\} = 0 \tag{7}$$

Thus, the evolution of the distribution function f is constrained by an infinite number of constants of motion, such as, for example, the number of particles, $N = \int f \; d\vec{R} \; dv_{\|}$, the entropy, $S = -\int f \ln f \; d\vec{R} \; dv_{\|}$, or the L_2-norm, $L_2 = \int f^2 \; d\vec{R} \; dv_{\|}$.

The set of equations (3) and (4) are energy conserving. Indeed, starting from the definition of the (perturbed) kinetic energy

$$E_{kin} \doteq \frac{m_i}{2} \int (f - f_M) \; v_{\|}^2 \; d\vec{R} \; dv_{\|} \tag{8}$$

(where f_M is such that $\int f_M \; dv_{\|} = n_0$) one can easily show that the total energy given by $E_{tot} = E_{kin} + E_{field}$ is conserved, provided that the field energy is given by

$$E_{field} \doteq \frac{q_i}{2} \int (f - f_M) \; \phi \; d\vec{R} \; dv_{\|} \tag{9}$$

and the electrostatic potential satisfies Dirichlet boundary conditions in a, $\phi|_{r=a} = 0$.

3 Numerical method

From a numerical point of view, it is important to note that Hamiltonian equations like (3) are known to develop, during their nonlinear evolution, increasingly smaller scales (see, e.g., [20] and references therein), which eventually cannot be resolved by the (finite) numerical grid. Thus, it is non-trivial

to simulate such Hamiltonian systems without introducing spurious dissipative effects. We have tried to take advantage of the fact that, as shown in the previous section, in the absence of collisions there is an infinite number of conserved quantities and to construct the code CYGNE in such a way as to preserve a good control of these constants of motion (namely, the number of ions N, the entropy S, the L_2-norm L_2, and, of course, the total energy E_{tot}).

3.1 Splitting scheme for the Vlasov equation

Eq. (3) is solved using a time-splitting technique. The four-dimensional (4D) equation (3) can be reduced to a sequence of two 2D equations:

$$\frac{\partial f}{\partial t} + \vec{v}_{GC} \cdot \vec{\nabla}_\perp f = 0 \tag{10}$$

$$\frac{\partial f}{\partial t} + v_\parallel \frac{\partial f}{\partial z} + \dot{v}_\parallel \frac{\partial f}{\partial v_\parallel} = 0 \tag{11}$$

A formal implicit solution of Eqs. (10)-(11) can be written as

$$f^*(r,\theta,z,v_\parallel) = f^n(r-\Delta r, \theta - \Delta\theta, z, v_\parallel) \tag{12}$$

$$f^{n+1}(r,\theta,z,v_\parallel) = f^*(r,\theta,z-\Delta z, v_\parallel - \Delta v_\parallel) \tag{13}$$

where Δr, $\Delta\theta$, Δz and $\Delta v_\parallel$ are computed solving the equations of motion (1)-(2). In this way, the integration of the Vlasov equation is reduced to successive interpolation problems, which are solved using cubic splines. Denoting by $\widehat{r\theta}/2$ the shifting over half the time-step $\Delta t/2$ in r-θ directions (see Eq. (12)), by $\widehat{zv_\parallel}$ the shifting over Δt in z-$v_\parallel$ directions (Eq. (13)) and by $\hat{Q}$ the solution of the quasi-neutrality equation (4), the time integration scheme can be written symbolically as

$$S_1 = \frac{\widehat{r\theta}}{2}\ \hat{Q}\ \widehat{zv_\parallel}\ \hat{Q}\ \frac{\widehat{r\theta}}{2}\ . \tag{14}$$

The composite operator S_1 provides second order accuracy in time if the factors $\widehat{r\theta}$ and $\widehat{zv_\parallel}$ are second order accurate, as predicted by the Strang splitting method [21]. In order to reduce the complexity of the scheme, the combination of $\widehat{r\theta}/2$ at the end of one time-step and at the beginning of the next can be replaced by the single operator $\widehat{r\theta}$, the order of accuracy remaining still two.

Note that, in straight cylinder configuration, Eq. (11) can be further splitted into two 1D equations and one can adopt equally well a time integration scheme of the form:

$$S_2 = \frac{\hat{v}_\parallel}{2}\ \frac{\hat{z}}{2}\ \hat{Q}\ \widehat{r\theta}\ \frac{\hat{z}}{2}\ \hat{Q}\ \frac{\hat{v}_\parallel}{2} \tag{15}$$

where shifting in z and $v_\parallel$ are performed over half the time-step, $\Delta t/2$ and with first order accuracy, being given by equations of the form $z(t) = z(t + \Delta t) - v_\parallel(t + \Delta t)\Delta t$ and $v_\parallel(t) = v_\parallel(t + \Delta t) - q_i E_z(t)\Delta t/m_i$. It can be shown that S_2 provides second order accuracy, using analogous arguments developed in the Cheng and Knorr splitting method [22]. Indeed, the composite operator S_2, when applied to the distribution function, gives

$$\begin{aligned} & f^{n+1}(r,\theta,z,v_\parallel) = S_2\ f^n(r,\theta,z,v_\parallel) \\ \equiv\ & f^n(\bar{r},\bar{\theta},z-[v_\parallel - E_z(t)\Delta t/2]\Delta t, v_\parallel - [E_z(t)+E_z(t+\Delta t)]\Delta t/2) \end{aligned} \tag{16}$$

where $\bar{r}$ and $\bar{\theta}$ are the feet of the characteristics in r-θ which are supposed to be calculated to second order accuracy (see the next subsection). Eq. (16) is equivalent to the following integrated equations:

$$z(t) = z(t+\Delta t) - [v_\parallel(t+\Delta t) - E_z(t)\Delta t/2]\ \Delta t \tag{17}$$

$$v_\parallel(t) = v_\parallel(t+\Delta t) - [E_z(t) + E_z(t+\Delta t)]\ \Delta t/2 \tag{18}$$

It is easily seen that Eq. (18) provides a second order accurate scheme, since

$$
\begin{aligned}
v_{\parallel}(t+\Delta t) - v_{\parallel}(t) &= [E_z(t) + E_z(t+\Delta t)]\ \Delta t/2 \\
&= E_z(t+\Delta t/2)\Delta t + O(\Delta t^3)
\end{aligned} \tag{19}
$$

Eq. (17) corresponds to the following two equations:

$$
z(t) = z(t+\Delta t) - [v_{\parallel}(t+\Delta t) - E_z(t)\Delta t/2]\ \Delta t \tag{20}
$$

$$
z(t-\Delta t) = z(t) - [v_{\parallel}(t) - E_z(t-\Delta t)\Delta t/2]\ \Delta t \tag{21}
$$

the difference of which, using Eq. (19), results to be second order accurate in time:

$$
\begin{aligned}
z(t-\Delta t) - 2z(t) + z(t+\Delta t) &= [v_{\parallel}(t+\Delta t) - v_{\parallel}(t)]\ \Delta t \\
&\quad + [E_z(t-\Delta t) - E_z(t)]\ \Delta t^2/2 \\
&= [E_z(t+\Delta t) + E_z(t-\Delta t)]\ \Delta t^2/2 + O(\Delta t^3) \\
&= E_z(t)\ \Delta t^2 + O(\Delta t^3)
\end{aligned} \tag{22}
$$

3.2 Equations of motion

We have tested two different methods for solving the equations of motion (1)-(2).

The first method, implemented for the sequence (15), is based on the Newton-Raphson algorithm which is used to obtain the starting points of the particle trajectories in r-θ plane, as described in detail in Ref. [15]. We solve the following equation in Cartesian coordinates with second order accuracy in time

$$
\frac{\vec{X}(t+\Delta t) - \vec{X}(t)}{\Delta t} = \vec{v}_{GC}(\vec{X}(t+\Delta t/2), t+\Delta t/2)) \tag{23}
$$

where $\vec{X} = (x, y)$, z being a parameter in this context. Instead of using linear interpolation for the guiding centre velocity $\vec{v}_{GC} = (E_y/B_0, -E_x/B_0, 0)$ as suggested in Ref. [15], we use quadratic splines, the electric potential being

calculated with cubic splines in our code. Since $\vec{v}_{GC}$ is not known at time $t + \Delta t/2$, a two-time-level predictor-corrector scheme is used: with the predictor, the particle positions $\vec{X}(t + \Delta t/2)$ are given with first order accuracy in time. Thus, from the interpolated values of the distribution function at these positions and the solution of the quasi-neutrality equation, one can obtain the electric field components at time $t + \Delta t/2$. Finally, with the corrector, these values are used for the right-hand side of Eq. (23).

The second method, implemented for both the sequences (14) and (15), is a modified version of the Bulirsch-Stoer method. In order to solve first order differential equations of the form

$$\dot{w}(t) = g(w, t) \tag{24}$$

where the function g on the right-hand side is known, one can use the powerful Bulirsch-Stoer technique [18] which is based on the *modified midpoint rule*, which advances the vector $w(t)$ from t to $t + \Delta t$ by a sequence on n substeps $h = \Delta t/n$, and *rational function extrapolation*.

In our problem, the equations of motion in r and θ directions are solved in Cartesian coordinates and have the form of Eq. (24)

$$\dot{x}(t) = \frac{E_y(x, y, t)}{B_0}, \qquad \dot{y}(t) = -\frac{E_x(x, y, t)}{B_0} \tag{25}$$

Thus, in our case, g corresponds to the electric field components which are determined numerically, requiring (i) to calculate the distribution function, (ii) to solve the quasi-neutrality equation and (iii) obtaining $\vec{E} = -\vec{\nabla}\phi$. Performing all these operations at every intermediate time $t + h$ within each time-step $H = \Delta t$ would be numerically too expensive. For this reason, we cannot use the modified midpoint rule but the following first-order scheme. Denoting by

z_m (with $m = 0, \ldots, n$) the intermediate approximations of w in Eq. (24) which march along in steps of h, the formulas are

$$\begin{aligned} z_0 &= w(t+H) \\ z_1 &= z_0 - hg(t, z_0) \\ z_{m+1} &= z_{m-1} - 2hg(t, z_m), \qquad \forall m = 1, 2, \ldots, n-1 \\ w(t) \sim w_n &= \frac{1}{2}[z_n + z_{n-1} - hg(t, z_n)] \end{aligned} \tag{26}$$

Thus, in this scheme the intermediate approximations are calculated explicitly by interpolating $g(z_m, t)$ (since these functions are known in CYGNE only on the grid points). We use then rational functions to extrapolate the values obtained with different n to the limit $h \to 0$.

In the case of sequence (14), we need to solve the second-order equation $\ddot{z} = q_i E_z / m_i$. In order to do this in CYGNE, modifications analogous to those performed to the *modified midpoint method* must be applied to the *Stoermer's rule* [18], since the knowledge of E_z at intermediate times is numerically too expensive.

Despite the fact that the Bulirsch-Stoer method is formally first order accurate in time (without considering the extrapolation procedure), we have found that it gives more accurate results with respect to the Newton-Raphson method (with predictor-corrector), as we will show in the next section. A possible explanation relies on the spatial accuracy of these two methods. Indeed, while the Bulirsch-Stoer technique requires only the calculation of the first derivatives of the potential field ϕ, the Newton-Raphson algorithm is based on the values of both the first and the second derivatives of ϕ, the latter being calculated with linear splines. Moreover, the Bulirsch-Stoer method is slightly faster as shown in Table 1, and the required memory is half that used in the

predictor-corrector scheme which needs to store the distribution function at two different times. The latter property is to be taken into consideration when we will move to a 5D distribution function to include finite Larmor radius effects.

3.3 Quasi-neutrality equation

Eq. (4) is Fourier transformed in θ and z, and it is solved along r by a standard finite element procedure where the electrostatic potential is discretized as a sum over a finite element (cubic spline) basis $\Lambda_\alpha(r)$ as

$$\phi_{mn}(r,t) = \sum_\alpha \varphi_\alpha(t)\ \Lambda_\alpha(r) \tag{27}$$

where ϕ_{mn} are the Fourier components of ϕ. The spline coefficients $\varphi_\alpha(t)$ are determined by the following matrix equation

$$\sum_\beta A_{\alpha\beta}\ \varphi_\beta(t) = b_\alpha(t) \tag{28}$$

$$A_{\alpha\beta} \doteq \int_0^a \left[\frac{n_0}{\Omega_i B_0}\Lambda'_\alpha\Lambda'_\beta + \left(\frac{m^2 n_0}{r^2\Omega_i B_0} + (1-\delta_{n0})\frac{e n_0}{T_e}\right)\Lambda_\alpha\Lambda_\beta\right] r\ dr \tag{29}$$

$$b_\alpha(t) \doteq \int_0^a [n_{mn}(r,t) - n_0(r)]\ \Lambda_\alpha\ r\ dr \tag{30}$$

where $\Lambda'_\alpha = d\Lambda_\alpha/dr$, δ_{mn} is the Kronecker symbol and $n_{mn}(r,t)$ are the Fourier components of the ion density $n_i = \int f\ dv_\parallel$. While b_α must be calculated at every time-step, the matrix $A_{\alpha\beta}$, which is time-independent, can be calculated only at initialisation. The matrix $A_{\alpha\beta}$ is real, symmetric and positive definite. Thus, Eq. (28) can be solved with standard methods of linear algebra.

The radial profiles for the ion density n_0, the ion temperature T_i and the electron temperature T_e are input functions and are computed at initialisation from the derivative with respect to r given by

$$\frac{d}{dr}\ln G = -\frac{\kappa_G}{a}\ \cosh^{-2}\left(\frac{r-r_0}{\Delta r_G\ a}\right) \tag{31}$$

where $G = n_0, T_i, T_e$. In Eq. (31), r_0 is a point inside the cylinder, Δr_G and κ_G are input parameters. Eq. (31) is defined such that it is easy to control the driving term of the micro-instability to define a localised maximum κ_G at $r = r_0$ in the logarithmic derivative of the plasma profiles.

The plasma is initialised at time $t = 0$ by perturbing a Maxwellian distribution function $f_M(r, v_\parallel) = n_0(m_i/2\pi T_i)^{1/2}\exp(-m_i v_\parallel^2/(2T_i))$ with a superposition of modes with random phases and amplitudes

$$f(r,\theta,z,v_\parallel,0) = f_M(r,v_\parallel)\left[1+\epsilon h(v_\parallel)g(r)\sum_{n,m}\alpha_{nm}\cos(2\pi nz/L + m\theta + 2\pi\alpha'_{mn})\right] \tag{32}$$

where $\alpha_{mn}, \alpha'_{mn}$ are random numbers between 0 and 1, ϵ is the perturbation amplitude, $1 \le n \le n_{max}$, $1 \le m \le m_{max}$ (ϵ, n_{max} and m_{max} being input parameters), $h(v_\parallel)$ and $g(r)$ are Gaussian functions centered at $v_\parallel = 0$ and $r = r_0$, respectively. The ion distribution function is assumed periodic in θ and z directions, while in r and $v_\parallel$ directions fixed boundary conditions are considered, i.e. $f = f_M$ at $r = a$ and $v_\parallel = \pm v_{max}$, where v_{max} is the cutoff velocity. The same boundary conditions are considered for the calculation of the trajectories: z and θ are periodic, while r and $v_\parallel$ are such that if $|v_\parallel| > v_{max}$, we set $|v_\parallel| = v_{max}$ and if $r > a$, we set $r = a$.

We require that the electric field is uniquely defined and finite on the axis $r = 0$, which means to impose the following boundary conditions for the electric potential

$$\phi'_{m=0,n}|_{r=0} = 0, \quad \forall n \tag{33}$$
$$\phi_{mn}|_{r=0} = 0, \quad \forall m \neq 0, \forall n \tag{34}$$

Moreover, we choose that there is not tangential electric field on the outer boundary giving

$$\phi_{mn}|_{r=a} = 0, \quad \forall (m,n) \neq (0,0) \tag{35}$$

In addition, we set the zero of the potential at $r = a$, thus $\phi_{m=0,n=0}|_{r=a} = 0$.

3.5 Preservation of the positivity of f

As discussed at the beginning of this section, Hamiltonian systems develop increasingly smaller scales as time evolves. Eventually, these scales become smaller than the grid size and cannot be resolved correctly. As a consequence, regions with negative distribution function can appear. Negative values in f are due to high order interpolation schemes (as cubic splines used in CYGNE), which have the tendency to produce overshoots and undershoots when fine-scale filamentation develops unless some additional averaging (dissipative) procedure is applied [23]. Different methods have been developed in the past to get rid of these problems, examples being the 'piecewise parabolic' [24] or the 'positive flux conservative' method [25], which impose the preservation of monotonicity and/or positivity on f. The reason why we do not use them in CYGNE is that this code is only a first step toward more complex geometry and physics, and a 'physical cutoff', given by the Larmor radius ρ, will appear

in the system when we will move to gyrokinetic equations. Thus, it is expected that scales in r and θ directions smaller than ρ will be physically eliminated without introducing spurious numerical diffusion.

In CYGNE we have implemented another method to ensure the positivity of f, which does not introduce any additional dissipation. This is performed by interpolating the function $(\ln f)$ instead of f itself. This is accurately obtained with cubic spline interpolation, for example in the radial direction:

$$\ln f(r, \theta, z, v_\parallel, t) = \sum_\alpha \tilde{c}_\alpha(\theta, z, v_\parallel, t)\ \Lambda_\alpha(r). \tag{36}$$

We then use $\tilde{c}_\alpha$ to interpolate the distribution function at the actual particle position r^* and to calculate $\ln f(r^*, \theta, z, v_\parallel, t)$. Inverting the logarithm now gives the desired values of $f(r^*, \theta, z, v_\parallel, t)$ which, by construction, are positive. Note that this scheme does not introduce any additional dissipation, being just another way of performing cubic spline interpolation, as shown in Fig. 1a-b, where the relative error in the entropy and the L2-norm are plotted against time for cases without (solid line) and with (dashed line) logarithmic interpolation.

This method has another practical advantage. When the smaller scales approach the grid size and the resolution is not sufficient to correctly resolve them, the constants of motion can continue to be conserved due sometimes to a significant proportion of negative f values. With this new interpolation technique, imposing positive f values, the constants of motion are no longer conserved when the simulation is dominated by features with scales smaller than the grid size. Thus, the quality of the simulations is assured by looking simply at the time evolution of the constants of motion. This is easier and more systematic than by looking, for example, at the evolution of the spectra

of f to understand if unresolved small scales are present in the system.

3.6 Improved spatial accuracy

As mentioned before, it turns out that in our case the appearance of negative f values is related to the loss of sufficient spatial accuracy to resolve small scales features. An improvement has been obtained by using the Bulirsch-Stoer algorithm, however this is not sufficient. In order to be able to increase the spatial accuracy without increasing the required memory size and CPU time per processor, we have generalized cubic splines to non-uniform grids, as detailed in the Appendix. This is performed for the meshes in the non-periodic directions, r and $v_\parallel$. In general, a proper choice of the mesh points distribution allows us to obtain the same physical results as those obtained with a uniform grid with the double of points, which results in a gain of more than a factor of two in both memory size and CPU time, as shown in Table 2. We will show a comparison in the next section.

3.7 Parallelisation strategy

Since most of the computing time is spent on evolving the distribution function, we have distributed the values of f amongst the processors as follows. During the advection in r-θ (see Eq. 12), each processor stores values of f for all r and θ grid points for a different subset of z and $v_\parallel$, which are treated as parameters in this context, and *viceversa* during the advection in z-$v_\parallel$ (see Eq. 13). Communications between processors are needed to move between the two kinds of storage and to calculate the ion density used in the right-hand

side of the quasi-neutrality equation (which is not parallelised). The code is written in fortran 90 and it is parallelized using MPI library.

With relatively large grid, we are able to scale well up to 64 processors on different machines [19]. A typical run with phase space resolution of $N_r = N_z = 64$, $N_\theta = 256$, $N_{v_\parallel} = 32$, non-equidistant meshes and time-step $\Delta t\ \Omega_i = 0.2$ requires 80 h CPU time up to the saturation phase of the instability, using 16 processors on Janus, a cluster of 25 ES45 HP-Compaq nodes with Quadrics interconnections of the Ecole Polytechnique Fédérale de Lausanne. The required memory can be estimated considering that the largest array (that is, the distribution function) has size $N_r N_\theta N_z N_{v_\parallel}/N_{proc}$, where N_{proc} is the number of processors, and that we store four real 3D global arrays of size $N_r N_\theta N_z$ (namely, the electric potential and the three components of the electric field) and two complex 3D arrays during the solution of the quasi-neutrality equation. Thus, the memory size per processor is roughly given by

$$M \simeq N_r N_\theta N_z N_{v_\parallel}/N_{proc} + 8 N_r N_\theta N_z \tag{37}$$

During communications between processors, which require transposition of the 4D array, the memory requirements are nearly double, $\simeq 2M$ when the first term in Eq. (37) is dominant. As an example, Table 2 shows the memory requirements per processor with and without communications in the case of $N_{proc} = 4$. In order to reduce both communications and required memory, a method to avoid the transposition of the distribution function is under study.

4 Validation of the code

The code is validated in two different ways. First, we compare CYGNE results with those obtained with electrostatic PIC codes which solve the same equations. Second, we study the convergence of the physical quantities, such as the radial heat flux and the energy, as phase space resolution increases.

4.1 Comparisons with PIC codes

CYGNE has been benchmarked quantitatively against two PIC codes: the linear code LORB5 [26] and the nonlinear code ORB5 [8].

The parameters for the comparison case are listed in Table 3, where $\rho_S = \sqrt{T_e(r_0)/m_i}/\Omega_i$, $T_{i0} = T_i(r_0)$ and $v_{thi0} = \sqrt{T_{i0}/m_i}$. Growthrates and frequencies of the ITG modes obtained with CYGNE (crosses) and with LORB5 (circles) are shown in Fig. 2 (frames (a) and (b), respectively), for different values of the mode numbers (m, n) in θ and z directions. The results are in good agreement.

For the nonlinear regime, Fig. 3 shows the time evolution of the field energy. Fourier components of the potential field $\phi_{m,n=0}(r)$, $\forall m$, are set to zero because the code ORB5 does not treat these modes. It can be seen that the saturation level of E_{field} obtained with CYGNE (solid line) agrees within a factor two with the level obtained with ORB5 (dashed line).

For this second validation of the code, we use parameters as in the comparison case in Table 3 except for the value of the perturbation amplitude, which is set to $\epsilon = 10^{-3}$ in order to shorten the linear regime during which, as we will show below, physical quantities converge much faster than in the nonlinear phase. Moreover, we include Fourier components $\phi_{m,n=0}(r)$, $\forall m$. We use a non-equidistant mesh in r and $v_\parallel$ directions, the Bulirsch-Stoer method described in the previous section (with sequence (14) for the time splitting) and the logarithmic interpolation for f.

We study the convergence of the field energy, the kinetic energy and the radial heat flux, given by

$$Q = \frac{m_i}{2} \int v_\parallel^2 \; \vec{v}_{GC} \cdot \vec{e}_r \; f \; d\vec{R} \; dv_\parallel \tag{38}$$

with the number of points in phase space keeping a fixed time-step, $\Delta t \; \Omega_i = 0.2$.

The time evolution of these three physical quantities is shown in Fig. 4, where the solid line corresponds to $(N_r, N_\theta, N_z, N_{v_\parallel}) = (64, 256, 64, 32)$, the dashed line to $(128, 256, 32, 32)$ and the dotted-dashed line to $(32, 128, 32, 32)$. In Fig. 4, we show for comparisons also the results obtained using the Newton-Raphson algoritm and predictor-corrector with resolution $(32, 128, 32, 32)$ (dotted line) and those obtained using equidistant meshes in r and $v_\parallel$ with the same resolution, $(32, 128, 32, 32)$ (dotted line and '+' markers).

In Fig. 5, we show the relative error in the constants of motion, namely the

number of ions (frame (a)), the total energy [1] (frame (b)), the entropy (frame (c)) and the L_2-norm (frame (d)) for the same simulations considered in Fig. 4.

As can be seen from Figs. 4-5, while convergence is reached during the linear regime and the turnover phase of the radial heat flux, enhanced resolution is required to describe the regime of nonlinear saturation, as appears from the time evolution of the kinetic energy. The constants of motion are conserved with a good accuracy until (unresolved) small scales develop, as discussed in the previous section. The time at which the conservation is no longer satisfied is delayed as resolution increases. From Figs. 4-5, the advantage of using non-equidistant meshes is evident (compare dotted-dashed line with dotted line and '+' markers): with the same number of points we gain in accuracy in the conservation of all constants of motion. Finally, from these two figures the Bulirsh-Stoer method turns out to give slightly more accurate results than those obtained with the Newton-Raphson algoritm (compare dotted-dashed line with dotted line).

In Figs. 6-7, the time evolution of the physical quantities and of the relative error in the constants of motion are shown for the phase space resolution corresponding to the solid line of Figs. 4-5, $(N_r, N_\theta, N_z, N_{v_\parallel}) = (64, 256, 64, 32)$, and different time-steps, $\Delta t\ \Omega_i = 0.2$ (solid line), $\Delta t\ \Omega_i = 0.5$ (dashed line) and $\Delta t\ \Omega_i = 1$ (dotted-dashed line). The time-step in semi-Lagrangian codes is not restricted by the Courant condition, $\Delta t < \Delta x / \max|U|$, but by the deformational Courant number, $\Delta t < (\max(|\partial U/\partial x|))^{-1}$, where U is the velocity field component along the x coordinate, which in general is less restrictive [14,15].

[1] The relative error in the total energy is computed as $\Delta E_{tot}(t)/\Delta E_{avg}(t)$, where $\Delta E = E(t) - E(0)$ and $\Delta E_{avg} = (|\Delta E_{field}| + |\Delta E_{kin}|)/2$.

We have seen that in our problem small time-steps are required during the nonlinear phase, because of the appearance of steep gradients in the velocity field, while during the linear phase larger time-steps can be used.

Finally, we show in Fig. 8 the cross-section of the distribution function in the r-θ plane at time $t\ \Omega_i = 1200$, that is during the nonlinear saturation regime. Filaments appear in the distribution function which become finer and finer as time evolves, and eventually cannot be resolved correctly by the (finite) grid resolution of the simulation. In order to study the saturation phase for a longer time, we need to move to a more complete physical model given by gyroaveraged equations. In this case, it can be expected that scales smaller than the ion Larmor radius in the perpendicular direction to the magnetic field (r, θ directions) will be smoothed out, thus avoiding the problem of the development of infinitesimally small scales as we found in the present drift-kinetic model.

5 Conclusions

CYGNE is a massively parallel code which uses the semi-Lagrangian approach to solve the drift-kinetic equations in a straight cylinder configuration. With this code we are able to follow the development of the ion temperature gradient driven instability until the nonlinear saturation occurs. The quality of the global nonlinear simulations is measured by checking the evolution of the constants of motion which are found to be conserved with a good accuracy when sufficiently high phase space resolution is used. As time evolves, small scales appear in the plasma which eventually become smaller than the grid size and cannot be resolved correctly. We have not used diffusive interpolation

methods to regularize the smallest structures, because CYGNE is only a first step toward a more general gyrokinetic formulation. When finite Larmor radius effects will be included, a physical cutoff appears in the system and it can be expected that scales in r and θ directions smaller than the Larmor radius will be smoothed out, allowing us to study cases deeper into the nonlinear saturation regime of ITG modes.

Acknowledgments

We are grateful to X. Garbet, one of those who initiated this project. We would like to thank A. Bottino for LORB5 and ORB5 data results and T.M. Tran for useful discussions on the performance of the code. Simulations were performed on the parallel servers Eridan (SGI Origin3800) and Janus of the Ecole Polytechnique Fédérale de Lausanne. This work was partly supported by the Swiss National Science Foundation.

A Cubic splines for non-equidistant meshes

In order to generalize cubic splines to the case of non-equidistant meshes, we start from the recurrence relation for constructing spline functions $B_{i,k}$ [27]:

$$B_{i,k}(x) = \frac{x - x_{i-2}}{x_{i+k-3} - x_{i-2}} B_{i,k-1}(x) + \frac{x_{i+k-2} - x}{x_{i+k-2} - x_{i-1}} B_{i+1,k-1}(x) \qquad \text{(A.1)}$$

where $B_{i,1}(x) = 1$ if $x_{i-2} \leq x \leq x_{i-1}$ and 0 otherwise. If we call $h_i = x_i - x_{i-1}$, we obtain for the linear spline basis the following expression

$$B_{i,2}(x) = \begin{cases} (x - x_{i-2})/h_{i-1} & \text{if } x_{i-2} \leq x \leq x_{i-1} \\ (x_i - x)/h_i & \text{if } x_{i-1} \leq x \leq x_i \\ 0 & \text{otherwise} \end{cases}$$

for the quadratic spline basis:

$$B_{i,3}(x) = \begin{cases} (x - x_{i-2})^2/(h_{i-1}h_{12}) & \text{if } x_{i-2} \leq x \leq x_{i-1} \\ (x_i - x)(x - x_{i-2})/(h_i h_{12}) + & \\ +(x_{i+1} - x)(x - x_{i-1})/(h_i h_{23}) & \text{if } x_{i-1} \leq x \leq x_i \\ (x_{i+1} - x)^2/(h_{i+1}h_{23}) & \text{if } x_i \leq x \leq x_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

where $h_{12} = h_{i-1} + h_i$ and $h_{23} = h_i + h_{i+1}$, and for the cubic spline basis:

$$B_{i,4}(x) = \begin{cases} (x - x_{i-2})^3/(h_{i-1}h_{12}h_{123}) & \text{if } x_{i-2} \leq x \leq x_{i-1} \\ (x_i - x)(x - x_{i-2})^2/(h_i h_{12}h_{123}) + & \\ +(x_{i+1} - x)(x - x_{i-2})(x - x_{i-1})/(h_i h_{23}h_{123}) + & \\ +(x_{i+2} - x)(x - x_{i-1})^2/(h_i h_{23}h_{234}) & \text{if } x_{i-1} \leq x \leq x_i \\ (x_{i+1} - x)^2(x - x_{i-2})/(h_{i+1}h_{23}h_{123}) + & \\ +(x_{i+2} - x)(x_{i+1} - x)(x - x_{i-1})/(h_{i+1}h_{23}h_{234}) + & \\ +(x_{i+2} - x)^2(x - x_i)/(h_{i+1}h_{34}h_{234}) & \text{if } x_i \leq x \leq x_{i+1} \\ (x_{i+2} - x)^3/(h_{i+2}h_{34}h_{234}) & \text{if } x_{i+1} \leq x \leq x_{i+2} \\ 0 & \text{otherwise} \end{cases}$$

where $h_{34} = h_{i+1} + h_{i+2}$, $h_{123} = h_{i-1} + h_i + h_{i+1}$ and $h_{234} = h_i + h_{i+1} + h_{i+2}$.

A 1D function $g(x)$ can be written in terms of the cubic spline basis $B_{i,4}(x)$ for all $x \in [x_0, x_N]$ as:

$$g(x) = \sum_{i=-1}^{N+1} c_i B_{i,4}(x) \tag{A.2}$$

The coefficients c_i are computed solving the following system of equations:

$$g(x_j) = \sum_{i=-1}^{N+1} c_i B_{i,4}(x_j) \qquad j = 0, \ldots, N \tag{A.3}$$

The system (A.3) can be written in the following matricial form:

$$\Big[B\Big]_{(N+3)\times(N+1)} \begin{pmatrix} c_{-1} \\ \vdots \\ c_{N+1} \end{pmatrix} = \begin{pmatrix} g(x_0) \\ \vdots \\ g(x_N) \end{pmatrix}$$

Thus, there are $(N+1)$ equations and $(N+3)$ unknowns. Imposing periodic or non-periodic boundary conditions gives rise to other two equations.

A.1 Non-periodic boundary conditions

We suppose that the first derivate is known at the boundaries. Derivatives of spline functions satisfy the following relation [27]

$$\frac{d}{dx}\left[\sum_{i=-1}^{N+1} c_i B_{i,k}(x)\right] = \sum_{i=-1}^{N+2} (k-1)\frac{c_i - c_{i-1}}{x_{i+k-3} - x_{i-2}} B_{i,k-1}(x) \tag{A.4}$$

where $c_{-2} = c_{N+2} = 0$. Eq. (A.4) can be written as:

$$\frac{d}{dx}\left[\sum_{i=-1}^{N+1} c_i B_{i,k}(x)\right] = \sum_{i=-1}^{N+1} (k-1)c_i \left(\frac{B_{i,k-1}(x)}{x_{i+k-3} - x_{i-2}} - \frac{B_{i+1,k-1}(x)}{x_{i+k-2} - x_{i-1}}\right)$$
$$= \sum_{i=-1}^{N+1} c_i \tilde{B}_{i,k-1}(x) \tag{A.5}$$

where

$$\tilde{B}_{i,3}(x) = 3 \begin{cases} (x - x_{i-2})^2/(h_{i-1}h_{12}h_{123}) & \text{if } x_{i-2} \le x \le x_{i-1} \\ (x_i - x)(x - x_{i-2})/(h_i h_{12} h_{123}) + & \\ +(x_{i+1} - x)(x - x_{i-1})/(h_i h_{23} h_{123}) + & \\ -(x - x_{i-1})^2/(h_i h_{23} h_{234}) & \text{if } x_{i-1} \le x \le x_i \\ (x_{i+1} - x)^2/(h_{i+1}h_{23}h_{123}) + & \\ -(x_{i+1} - x)(x - x_{i-1})/(h_{i+1}h_{23}h_{234}) + & \\ -(x_{i+2} - x)(x - x_i)/(h_{i+1}h_{34}h_{234}) & \text{if } x_i \le x \le x_{i+1} \\ -(x_{i+2} - x)^2/(h_{i+2}h_{34}h_{234}) & \text{if } x_{i+1} \le x \le x_{i+2} \\ 0 & \text{otherwise} \end{cases}$$

Using Eq. (A.4) (or, equivalently, Eq. (A.5)), we find that the following relations hold at the boundaries:

$$
\begin{aligned}
g'(x_0) &= \frac{3}{h_0+h_1}\left(\frac{c_0-c_{-1}}{x_1-x_{-2}}h_1+\frac{c_1-c_0}{x_2-x_{-1}}h_0\right)\\
&= \frac{3}{h_0+h_1}\left[c_0\left(\frac{h_1}{x_1-x_{-2}}-\frac{h_0}{x_2-x_{-1}}\right)+\frac{c_1h_0}{x_2-x_{-1}}-\frac{c_{-1}h_1}{x_1-x_{-2}}\right]\\
&= c_0F_0+c_1F_1+c_{-1}F_{-1} \qquad \text{(A.6)}\\
g'(x_N) &= \frac{3}{h_N+h_{N+1}}\left(\frac{c_N-c_{N-1}}{x_{N+1}-x_{N-2}}h_{N+1}+\frac{c_{N+1}-c_N}{x_{N+2}-x_{N-1}}h_N\right)\\
&= \frac{3}{h_N+h_{N+1}}[c_N\left(\frac{h_{N+1}}{x_{N+1}-x_{N-2}}-\frac{h_N}{x_{N+2}-x_{N-1}}\right)\\
&\quad +\frac{c_{N+1}h_N}{x_{N+2}-x_{N-1}}-\frac{c_{N-1}h_{N+1}}{x_{N+1}-x_{N-2}}]\\
&= c_NF_N+c_{N+1}F_{N+1}+c_{N-1}F_{N-1} \qquad \text{(A.7)}
\end{aligned}
$$

which define the constants F_{-1}, F_0, F_1, F_{N-1}, F_N and F_{N+1}.

Using Eqs. (A.6)-(A.7) and the values of the cubic spline basis listed in Table 4, the $(N+3, N+3)$ matricial system to be solved becomes [2]:

$$
\begin{pmatrix}
F_{-1} & F_0 & F_1 & & & \\
D_{-1}^{(1)} & D_0^{(0)} & D_1^{(-1)} & & 0 & \\
 & D_0^{(1)} & D_1^{(0)} & D_2^{(-1)} & & \\
 & \ddots & \ddots & \ddots & & \\
0 & & D_{N-1}^{(1)} & D_N^{(0)} & D_{N+1}^{(-1)} & \\
 & & F_{N-1} & F_N & F_{N+1} &
\end{pmatrix}
\times
\begin{pmatrix} c_{-1} \\ c_0 \\ \vdots \\ \vdots \\ c_N \\ c_{N+1} \end{pmatrix}
=
\begin{pmatrix} g'(x_0) \\ g(x_0) \\ \vdots \\ \vdots \\ g(x_N) \\ g'(x_N) \end{pmatrix}
\qquad \text{(A.9)}
$$

[2] The coefficients c_i, for $i = 0, \ldots, N$, are computed solving the following system of equations

$$
g(x_j) = \sum_{i=-1}^{N+1} c_i B_{i,4}(x_j) = c_{j-1}D_{j-1}^{(1)} + c_jD_j^{(0)} + c_{j+1}D_{j+1}^{(-1)} \qquad \text{(A.8)}
$$

where factors $D_j^{(-1,0,1)}$ are defined in Table 4.

If we permute the matrix to keep the boundary conditions in the last two rows, Eq. (A.9) reads:

$$\tilde{A}\begin{pmatrix} u' \\ \\ v' \end{pmatrix} = \begin{pmatrix} u \\ \\ v \end{pmatrix} \qquad \text{where: } \begin{cases} u' = (c_0, \cdots, c_N)^t \\ v' = (c_{N+1}, c_{-1})^t \\ u = (g(x_0), \cdots, g(x_N))^t \\ v = (g'(x_N), g'(x_0))^t \end{cases} \tag{A.10}$$

and

$$\tilde{A} = \left(\begin{array}{c|cc} A & \gamma & \\ \hline & & \\ \lambda & \xi_1 & \xi_2 \\ & & \\ & \xi_3 & \xi_4 \end{array}\right) \tag{A.11}$$

where:

$$\begin{cases} \cdot\, A \text{ is the } (N+1)\times(N+1) \text{ tridiagonal matrix : } \begin{pmatrix} D_0^{(0)} & D_1^{(-1)} & & & \\ D_0^{(1)} & D_1^{(0)} & D_2^{(-1)} & & \\ & \ddots & \ddots & \ddots & \\ & & D_{N-2}^{(1)} & D_{N-1}^{(0)} & D_N^{(-1)} \\ & & & D_{N-1}^{(1)} & D_N^{(0)} \end{pmatrix} \\ \cdot\, \lambda \text{ is the } 2\times(N+1) \text{ matrix : } \begin{pmatrix} 0 & \cdots & 0 & F_{N-1} & F_N \\ F_0 & F_1 & 0 & \cdots & 0 \end{pmatrix} \\ \cdot\, \gamma \text{ is the } (N+1)\times 2 \text{ matrix : } \begin{pmatrix} D_{-1}^{(1)} & 0 & \cdots & 0 \\ 0 & \cdots & 0 & D_{N+1}^{(-1)} \end{pmatrix}^t \\ \cdot\, \delta = \begin{pmatrix} \xi_1 & \xi_2 \\ \xi_3 & \xi_4 \end{pmatrix} = \begin{pmatrix} F_{N+1} & 0 \\ 0 & F_{-1} \end{pmatrix} \end{cases}$$

The system (A.10) can be solved with the same method described in Ref. [15] using optimised library subroutines.

In order to calculate integrals of cubic splines in the case of non-equidistant meshes, the following relations are needed

$$\int_{x_{i-2}}^{x_{i-1}} B_{i,4}(x)dx = h_{i-1}^3/(4h_{12}h_{123}) \tag{A.12}$$

$$\int_{x_{i+1}}^{x_{i+2}} B_{i,4}(x)dx = h_{i+2}^3/(4h_{34}h_{234}) \tag{A.13}$$

For $\int_{x_{i-1}}^{x_i} B_{i,4}(x)dx$ and $\int_{x_i}^{x_{i+1}} B_{i,4}(x)dx$, Gaussian quadrature can be used. Indeed, for a polynomial of degree three, results obtained with a 2-point Gaussian quadrature are exact [3]. By the linear transformation

$$t = [2x - (x_a + x_b)]/(x_b - x_a), \qquad x = [(x_b - x_a)t + (x_a + x_b)]/2 \tag{A.14}$$

the interval $[x_a, x_b]$ can be transformed to $[-1, 1]$ and one obtains:

$$\begin{aligned}\int_{x_a}^{x_b} f(x)dx &= \frac{x_b - x_a}{2}\int_{-1}^{1} f(t)dt \\ &= x_c[f(x_m - x_c t_G) + f(x_m + x_c t_G)]\end{aligned} \tag{A.15}$$

where $x_m = (x_a + x_b)/2$, $x_c = (x_b - x_a)/2$ and t_G is the Gaussian point $t_G = 0.577350269189626$ (for the 2-point formula, the Gaussian weigth is equal to 1).

[3] For the integration of the product of two Bs, 4-points Gaussian quadrature is needed, and the same rule can be generalized to integrate polynomials of higher order.

Dirichlet boundary conditions are imposed to solve the quasi-neutrality equation (4) as specified in Eqs. (34)-(35). Only the three splines nearest to the boundary need to be changed. The transformation is such that the sum of the basis functions at any given point remains equal to one [4]. In the case of non-equidistant meshes, the transformation on the inside boundary is

$$\begin{pmatrix} \hat{B}_{-1,4} \\ \hat{B}_{0,4} \\ \hat{B}_{1,4} \end{pmatrix} = \begin{pmatrix} E_0 & 0 & 0 \\ C_0 & 1 & 0 \\ A_0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} B_{-1,4} \\ B_{0,4} \\ B_{1,4} \end{pmatrix} \tag{A.16}$$

and on the outside boundary is:

$$\begin{pmatrix} \hat{B}_{N-1,4} \\ \hat{B}_{N,4} \\ \hat{B}_{N+1,4} \end{pmatrix} = \begin{pmatrix} 1 & 0 & A_N \\ 0 & 1 & C_N \\ 0 & 0 & E_N \end{pmatrix} \times \begin{pmatrix} B_{N-1,4} \\ B_{N,4} \\ B_{N+1,4} \end{pmatrix} \tag{A.17}$$

where:

[4] One can easily check that this condition is satisfied by $B_{i,k}$. For $k = 4$, we have indeed:

$$B_{i,4}(x_i) + B_{i-1,4}(x_i) + B_{i+1,4}(x_i) = \frac{1}{h_{23}} \left(\frac{h_{i+1}h_{12}}{h_{123}} + \frac{h_i h_{34}}{h_{234}} \right)$$
$$+ \frac{h_i^2}{h_{23}h_{234}} + \frac{h_{i+1}^2}{h_{23}h_{123}} = \frac{1}{h_{23}}(h_i + h_{i+1}) = 1$$

$$A_0 = -\frac{h_0^2(h_{-1}+h_0+h_1)}{h_1^2(h_0+h_1+h_2)} \tag{A.18}$$

$$C_0 = -[(h_{-1}+h_0)/h_1 + \frac{h_0(h_1+h_2)(h_{-1}+h_0+h_1)}{h_1^2(h_0+h_1+h_2)}] \tag{A.19}$$

$$E_0 = 1-(A_0+C_0) \tag{A.20}$$

$$A_N = -\frac{h_{N+1}^2(h_N+h_{N+1}+h_{N+2}}{h_N^2(h_{N-1}+h_N+h_{N+1})} \tag{A.21}$$

$$C_N = -[(h_{N+1}+h_{N+2})/h_N + \frac{h_{N+1}(h_N+h_{N-1})(h_N+h_{N+1}+h_{N+2})}{h_N^2(h_{N-1}+h_N+h_{N+1})}] \tag{A.22}$$

$$E_N = 1-(A_N+C_N) \tag{A.23}$$

References

[1] T.S. Hahm, Phys. Fluids 31 (1988) 2670.

[2] A. Brizard, J. Plasma Physics 41 (1989) 541.

[3] W.W. Lee, J. Comput. Phys. 72 (1987) 243.

[4] R. Sydora, V. Becyk and J. Bawson, Pl. Phys. Control. Fusion 38 (1996) A281.

[5] A. Dimits, Phys. Rev. Lett. 77 (1996) 71.

[6] Z. Lin, T.S. Hahm, W.W. Lee, W.M. Tang and R.B. White, Science 281 (1998) 1835.

[7] S. Parker, C. Kim and Y. Chen, Phy. of Plasmas 6 (1999) 1709.

[8] T.M. Tran, K. Appert, M. Fivaz, G. Jost, J. Vaclavik and L. Villard, in: Theory of Fusion Plasmas, Int. Workshop (Ed. Compositori, Società italiana di Fisica, Bologna, 1999), p. 45.

[9] Y. Idomura, S. Tokuda and Y. Kishimoto, Nuclear Fus. 43 (2003) 234.

[10] S.J. Allfrey and R. Hatzky, Comp. Phys. Comm. 154 (2003) 98.

[11] W. Dorland, F. Jenko, M. Kotschenreuther and B.N. Rogers, Phys. Rev. Lett. 85 (2000) 5579.

[12] F. Jenko, Comp. Phys. Comm. 125 (2000) 196.

[13] J. Candy and R.E. Waltz, J. Comput. Phys. 186 (2003) 545.

[14] A. Staniforth and J. Côté, Mon. Wea. Rev., 119 (1991) 2206.

[15] E. Sonnendrücker, J. Roche, P. Bertrand and A. Ghizzo, J. Comput. Phys. 149 (1999) 201.

[16] V. Grandgirard *et al.*, in: Proceedings of the 29th EPS Conference on Controlled Fusion and Plasma Physics, Montreaux, 2002, ECA Vol. 26B, P4.095.

[17] M. Brunetti *et al.*, in: Proceedings of the 29th EPS Conference on Controlled Fusion and Plasma Physics, Montreaux, 2002, ECA Vol. 26B, P4.102.

[18] W. H. Press, S. A. Teukolsky, W. T. Vetterling and B. P. Flannery, Numerical Recipes in Fortran, 2nd edition (Cambridge University Press, 1992), p. 716.

[19] V. Grandgirard *et al.*, to be published.

[20] The Nature of Chaos, Ed. by T. Mullin (Oxford University Press, 1993) p. 262.

[21] G. Strang, SIAM J. Numer. Anal. 5 (1968) 506.

[22] C.Z. Cheng and G. Knorr, J. Comput. Phys. 22 (1976) 330.

[23] T.D. Arber and R.G.L. Vann, J. Comput. Phys. 180 (2002) 339.

[24] P. Colella and P.R. Woodward, J. Comput. Phys. 54 (1984) 174.

[25] F. Filbet, E. Sonnendrücker and P. Bertrand, J. Comput. Phys. 172 (2001) 166.

[26] A. Bottino, T.M. Tran, O. Sauter, J. Vaclavik and L. Villard, in: Theory of Fusion Plasmas, Int. Workshop (Ed. Compositori, Società italiana di Fisica, Bologna, 2001), p. 327; A. Bottino, A. G. Peeters, O. Sauter, J. Vaclavik, L. Villard and ASDEX Upgrade Team, Phys. Plasmas 11 (2004) 198.

[27] C. deBoor, A practical guide to splines, Applied Mathematical Sciences, 27 (Springer-Verlag, New York, 2001).

Figure captions

(1) *Time evolution of the relative error in the entropy (a) and the L2-norm (b) without (solid line) and with (dashed line) logarithmic interpolation for the case* $(N_r, N_\theta, N_z, N_{v_\parallel}) = (64, 64, 64, 64)$ *and* $\Delta t\ \Omega_i = 0.2$.

(2) *ITG growthrates (a) and frequencies (b): values obtained with* CYGNE *(*$\times$*) and* LORB5 *(o).*

(3) *Time evolution of the field energy obtained with* CYGNE *(solid line) and* ORB5 *(dashed line).*

(4) *Time evolution of the field energy, the kinetic energy and the radial heat flux for different phase space resolutions:* $(N_r, N_\theta, N_z, N_{v_\parallel}) = (64, 256, 64, 32)$ *in solid line,* $(128, 256, 32, 32)$ *in dashed line,* $(32, 128, 32, 32)$ *in dotted-dashed line,* $(32, 128, 32, 32)$ *with Newton-Raphson algorithm in dotted line, and* $(32, 128, 32, 32)$ *using equidistant meshes in dotted line and '+' markers.*

(5) *Time evolution of the relative error in the number of ions (frame (a)), the total energy (b), the entropy (c) and the* L_2*-norm (d) for different phase space resolutions:* $(N_r, N_\theta, N_z, N_{v_\parallel}) = (64, 256, 64, 32)$ *in solid line,* $(128, 256, 32, 32)$ *in dashed line,* $(32, 128, 32, 32)$ *in dotted-dashed line,* $(32, 128, 32, 32)$ *with Newton-Raphson algorithm in dotted line, and* $(32, 128, 32, 32)$ *using equidistant meshes in dotted line and '+' markers.*

(6) *Time evolution of the field energy, the kinetic energy and the radial heat flux for the same phase space resolution,* $(N_r, N_\theta, N_z, N_{v_\parallel}) = (64, 256, 64, 32)$*, and different time-steps:* $\Delta t\ \Omega_i = 0.2$ *(solid line),* $\Delta t\ \Omega_i = 0.5$ *(dashed line) and* $\Delta t\ \Omega_i = 1$ *(dotted-dashed line).*

(7) *Time evolution of the relative error in the number of ions (frame (a)), the total energy (b), the entropy (c) and the* L_2*-norm (d) for the same*

resolution, $(N_r, N_\theta, N_z, N_{v_\parallel}) = (64, 256, 64, 32)$*, and different time-steps:* $\Delta t\ \Omega_i = 0.2$ *(solid line),* $\Delta t\ \Omega_i = 0.5$ *(dashed line) and* $\Delta t\ \Omega_i = 1$ *(dotted-dashed line).*

(8) *Cross-section of the distribution function in* r*-*θ *plane at time* $t\ \Omega_i = 1200$*.*

Table 1

CPU time (normalised to the time required in the case of Bulirsch-Stoer (BS) and splitting sequence S_2 given by Eq. (15)) for a grid $(N_r, N_\theta, N_z, N_{v_\parallel}) = (64, 256, 64, 32)$. NR means 'Newton-Raphson' and the splitting sequence S_1 is given by Eq. (14).

Method	Splitting sequence	CPU time
BS	S_2	1
BS	S_1	1.1
NR	S_2	1.3

Table 2

CPU time (normalised to the time required in the case with non-equidistant meshes) and memory requirements per processor (the last column refers to the memory size during communications) with and without equidistant meshes (EM) in r-$v_\parallel$ directions, using $N_{proc} = 4$.

			Memory size	
$(N_r, N_\theta, N_z, N_{v_\parallel})$	mesh	CPU time	w/o	with comm.
$(64, 256, 64, 32)$	non-EM	1	128 MB	225 MB
$(128, 256, 64, 64)$	EM	2.9	396 MB	768 MB

Table 3

Parameter set for the comparison case.

a/ρ_S	L/ρ_S	v_{max}/v_{thi0}	r_0	T_{i0}	n_{max}	m_{max}
14.5	1508	4.15	0.5 a	T_e	4	12

k_{T_e}	k_{T_i}	k_{n_0}	Δr_{T_i}	Δr_{n_0}	Z_i	ϵ
0	4	0.8	0.1	0.2	1	10^{-4}

Table 4

Values of the spline basis $B_{i,4}$, $B_{i,3}$ and $\tilde{B}_{i,3}$ at given points.

x	x_{i-1}	x_i	x_{i+1}
$B_{i,4}(x)$	$D_i^{(-1)} = \frac{h_{i-1}^2}{h_{12}h_{123}}$	$D_i^{(0)} = \frac{1}{h_{23}}\left(\frac{h_{i+1}h_{12}}{h_{123}} + \frac{h_i h_{34}}{h_{234}}\right)$	$D_i^{(1)} = \frac{h_{i+2}^2}{h_{34}h_{234}}$
$B_{i,3}(x)$	h_{i-1}/h_{12}	h_{i+1}/h_{23}	0
$\tilde{B}_{i,3}(x)$	$3h_{i-1}/(h_{12}h_{123})$	$3(h_{i+1}/h_{123} - h_i/h_{234})/h_{23}$	$-3h_{i+2}/(h_{34}h_{234})$

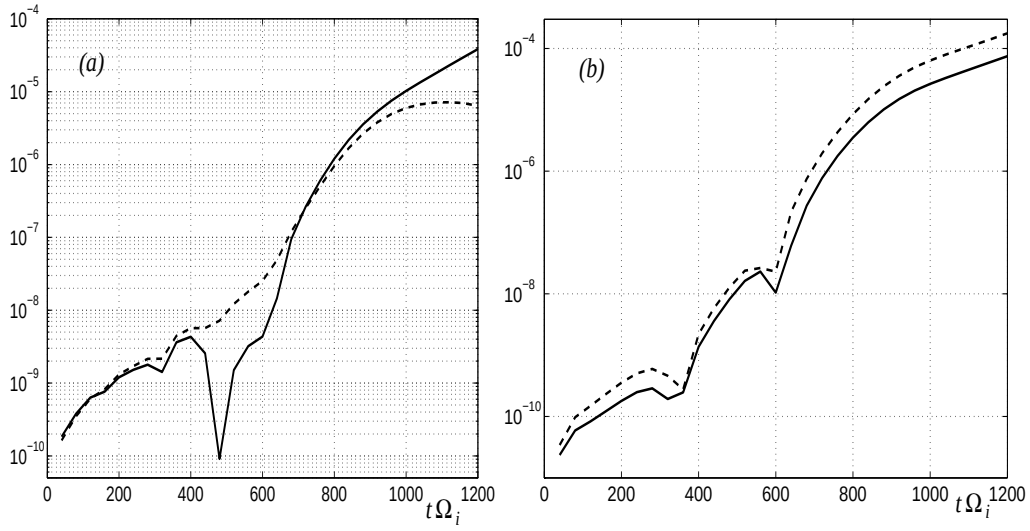

Fig. 1.

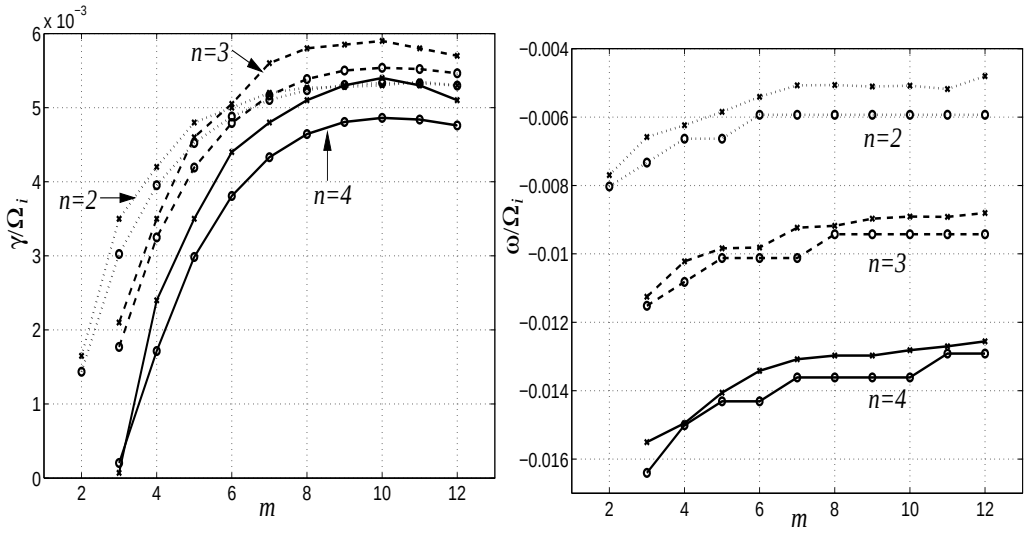

Fig. 2.

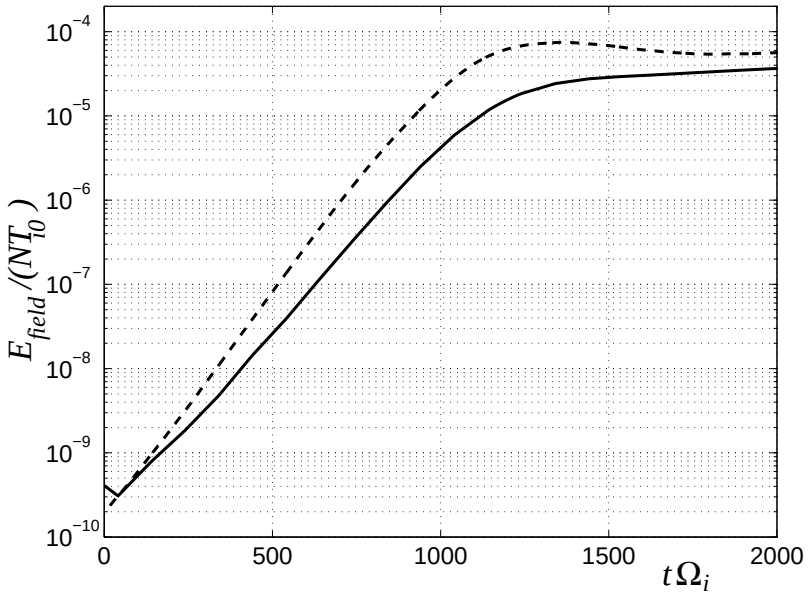

Fig. 3.

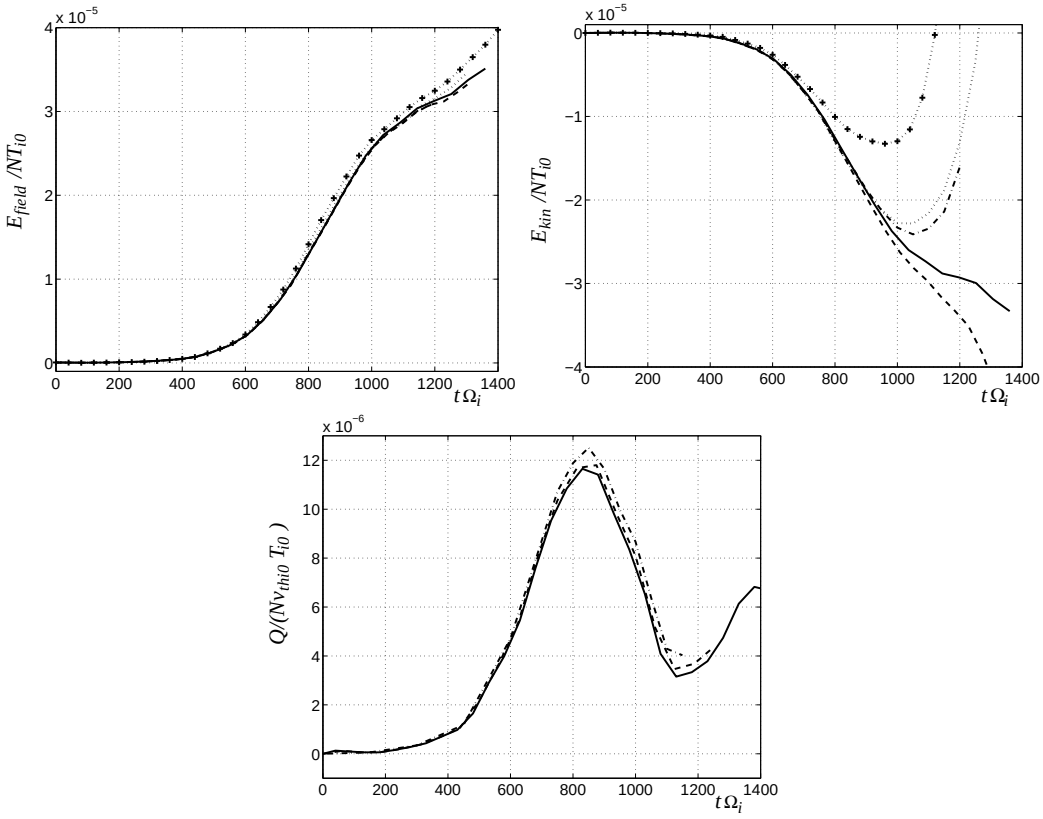

Fig. 4.

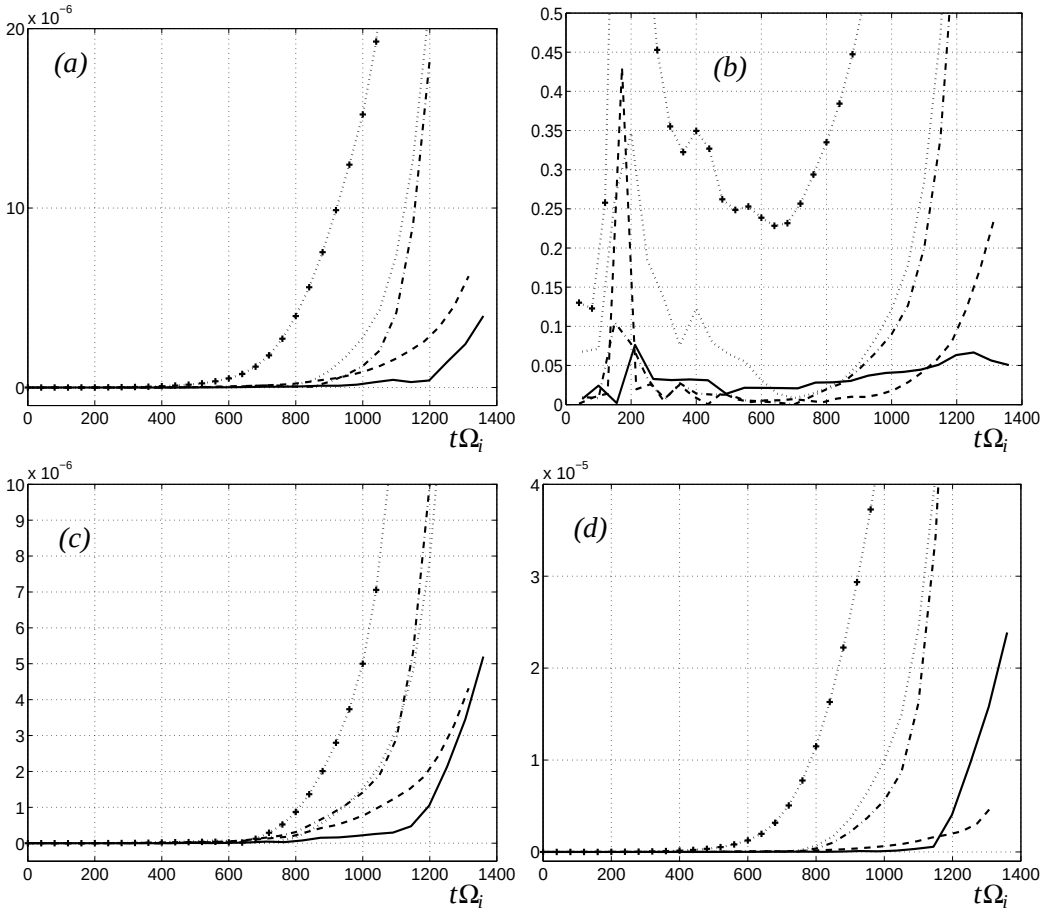
(a)
x 10^-6
20
10
0
0 200 400 600 800 1000 1200 1400
tΩi
(b)
0.5
0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
0 200 400 600 800 1000 1200 1400
tΩi
(c)
x 10^-6
10
9
8
7
6
5
4
3
2
1
0
0 200 400 600 800 1000 1200 1400
tΩi
(d)
x 10^-5
4
3
2
1
0
0 200 400 600 800 1000 1200 1400
tΩi

Fig. 5.

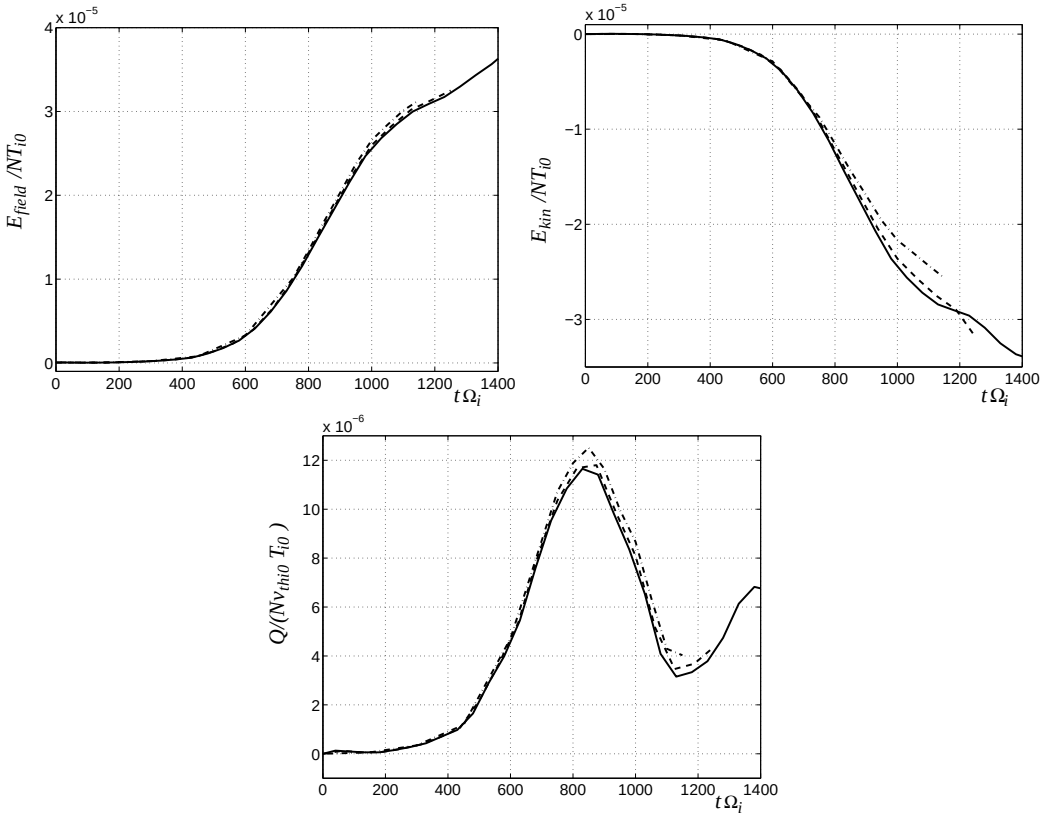

Fig. 6.

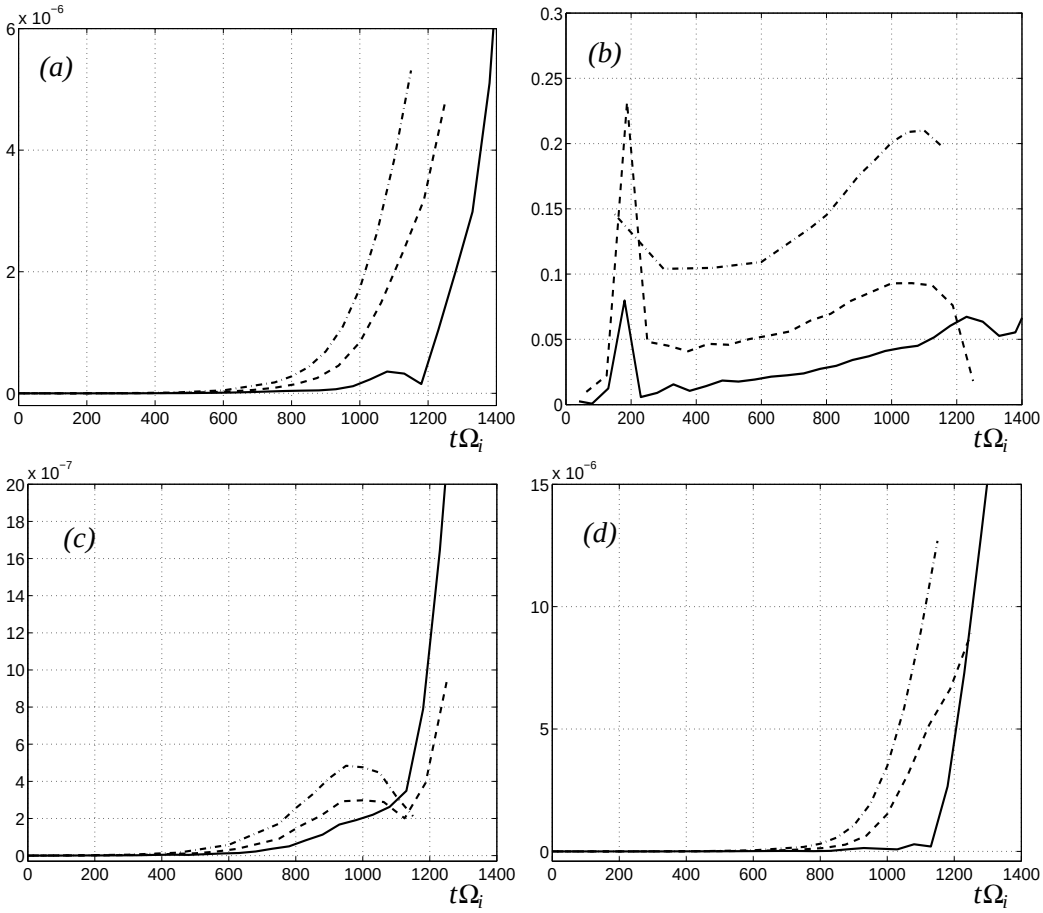
x 10^−6
(a)
6
4
2
0
0 200 400 600 800 1000 1200 1400
tΩ_i
(b)
0.3
0.25
0.2
0.15
0.1
0.05
0
0 200 400 600 800 1000 1200 1400
tΩ_i
x 10^−7
(c)
20
18
16
14
12
10
8
6
4
2
0
0 200 400 600 800 1000 1200 1400
tΩ_i
x 10^−6
(d)
15
10
5
0
0 200 400 600 800 1000 1200 1400
tΩ_i

Fig. 7.

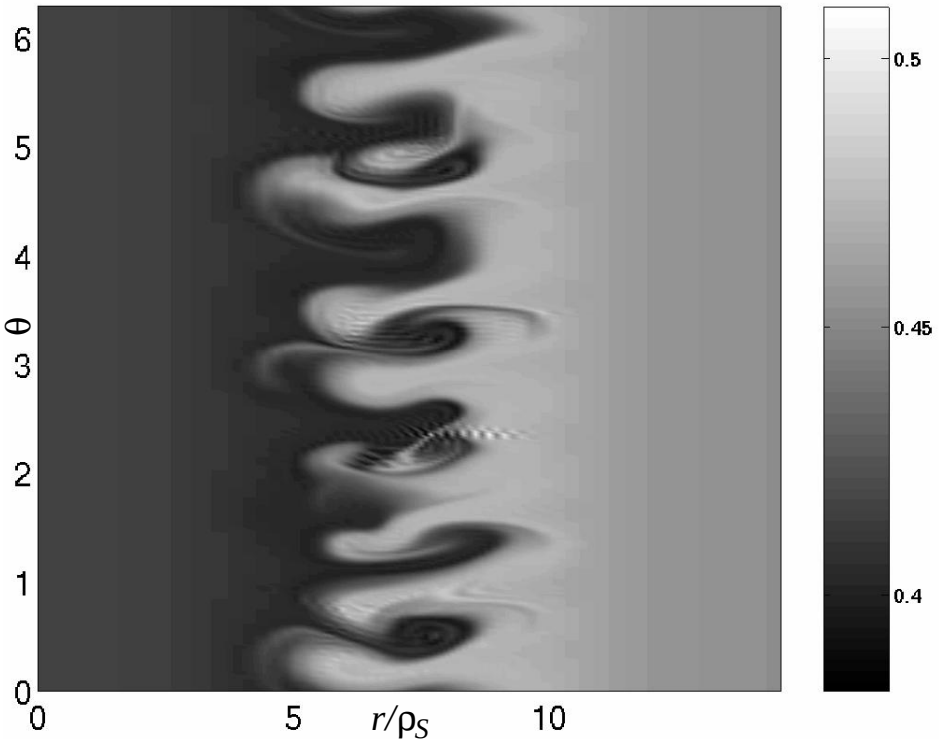

Fig. 8.