Towards the optimization of a gyrokinetic Particle-In-Cell (PIC) code on large-scale hybrid architectures

N Ohana¹, A Jocksch², E Lanti¹, T M Tran¹, S Brunner¹, C Gheller², F Hariri¹, L Villard¹

¹ Swiss Plasma Center, Ecole Polytechnique Fédérale de Lausanne, Switzerland
² Swiss National Supercomputing Centre, Lugano, Switzerland
E-mail: noe.ohana@epfl.ch

Abstract. With the aim of enabling state-of-the-art gyrokinetic PIC codes to benefit from the performance of recent multithreaded devices, we developed an application from a platform called the “PIC-engine” [1, 2, 3] embedding simplified basic features of the PIC method. The application solves the gyrokinetic equations in a sheared plasma slab using B-spline finite elements up to fourth order to represent the self-consistent electrostatic field. Preliminary studies of the so-called Particle-In-Fourier (PIF) approach, which uses Fourier modes as basis functions in the periodic dimensions of the system instead of the real-space grid, show that this method can be faster than PIC for simulations with a small number of Fourier modes. Similarly to the PIC-engine, multiple levels of parallelism have been implemented using MPI+OpenMP [2] and MPI+OpenACC [1], the latter exploiting the computational power of GPUs without requiring complete code rewriting. It is shown that sorting particles [3] can lead to performance improvement by increasing data locality and vectorizing grid memory access. Weak scalability tests have been successfully run on the GPU-equipped Cray XC30 Piz Daint (at CSCS) up to 4,096 nodes. The reduced time-to-solution will enable more realistic and thus more computationally intensive simulations of turbulent transport in magnetic fusion devices.

1. Introduction

In order to use nuclear fusion as a source of energy, magnetic confinement of plasmas needs to be improved. Experimental devices are facing difficulties associated with so-called anomalous transport, which is attributed to microturbulence. Because of the physical complexity and the chaotic behavior of the system, analytical studies are limited and numerical schemes are necessary to better understand instabilities occurring in it.

During the last decades, much effort has been invested into the development of first principle models, among which we find gyrokinetic Particle-In-Cell (PIC) codes. They apply a Monte-Carlo particle method to gyrokinetic theory, which takes advantage of the difference in time scales between the fast cyclotron motion of charged particles in a magnetic field and the slow instabilities. The phase space is thus reduced to five dimensions, and simulations can use larger time steps. The PIC method then consists of discretizing the distribution functions of each species with numerical markers, and representing the fields on a real space grid, e.g. with finite elements.
Despite the use of many numerical methods to reduce the noise inherent to particle methods (e.g. \( \delta f \) scheme [4], noise control [5], ...), we are still too limited by numerical resources to achieve the accuracy required to explain experimental observations. For instance, simulations of a medium-sized tokamak taking into account certain effects related to electron dynamics together with ion dynamics require a grid of at least \( 360 \times 1024 \times 512 \) cells, and 2 billion markers of each species [6]. Such resolution for large-sized devices reaches the limits of today’s computational resources.

We thus have to turn towards High Performance Computing (HPC), a field in constant evolution. Among the arising multi-threaded architectures, a trend in scientific computing is to use General-Purpose computation on Graphic Processing Units (GPGPU). GPUs turn out to be particularly suitable for PIC codes [7], thanks to the high number of particles that can be treated simultaneously by the device. The OpenACC directive-based interface is particularly interesting for portability purposes as one can run the same code on different architectures.

In this paper, we present the so-called “GK-engine”, a code that embeds all the main features of a gyrokinetic PIC code in a modular and portable structure. It can be used as a testbed for various numerical methods or data structures on different platforms, without requiring the refactoring of all the physics modules around the core. As an example, a variant of the classical electrostatic potential, \( \langle \cdot \rangle \)

\[
\begin{equation}
\frac{d\mathbf{R}}{dt} = v_\parallel b + \frac{\mu}{eB^*} \mathbf{b} \times \nabla \mathbf{B} + \frac{1}{B^*} \mathbf{b} \times \nabla \langle \phi \rangle
\end{equation}
\]

\[
\begin{equation}
\frac{dv_\parallel}{dt} = -\frac{e}{m} \mathbf{b} \cdot \nabla \langle \phi \rangle
\end{equation}
\]

\[
\begin{equation}
\frac{d\mu}{dt} = 0
\end{equation}
\]

with \( B^* = B \left( 1 + \frac{mB^*_y B_z}{eB^*_x v_\parallel} \right) \), \( \mathbf{b} \) the unitary magnetic field vector, \( e \) the particle charge, \( \phi \) the electrostatic potential, \( \langle \cdot \rangle \) standing for gyroaveraging and the prime ‘ for \( \partial/\partial x \).

The electrostatic field is solved using the quasineutrality equation with the long wavelength approximation and adiabatic electrons:

\[
\begin{equation}
\frac{en_0}{T_e} (\phi - \phi_0) - \nabla_\perp \cdot \left( \frac{n_0 B \Omega_i}{B^*} \nabla_\perp \phi \right) = \int \frac{B^*}{m} f(\mathbf{x} + \hat{\rho}_L(\mu, \alpha), v_\parallel; \mu) \, d\mu \, d\alpha \, dv_\parallel - n_0
\end{equation}
\]

2. The GK-engine model

For simplicity, we consider a slab-like system with Cartesian coordinates. \( x \) will represent the radial direction, \( y \) the poloidal one, and \( z \) the toroidal one. The magnetic field can thus be written as \( \mathbf{B}(x, y, z) = B_y(x) \mathbf{\hat{e}}_y + B_z \mathbf{\hat{e}}_z \). We have arbitrarily chosen to include the radial dependency in the poloidal component exclusively. The safety factor \( q(x) = \frac{L_y B_z}{L_z B_y} \) is a measure of the magnetic field tilt angle, where \( L_i \) stands for the length of dimension \( i \).

Gyrokinetic theory consists of following the guiding center dynamics in the 5D phase space \((\mathbf{R}, v_\parallel, \mu)\), where \( \mathbf{R} \) stands for the position, \( v_\parallel \) the projection of velocity on the magnetic field direction, and \( \mu = \frac{mv^2}{2m} \) the magnetic moment with \( m \) the particle mass, \( v_\perp \) the perpendicular velocity, and \( B \) the magnetic field strength. Within our geometry, the equations of motion are:

2. The GK-engine model

For simplicity, we consider a slab-like system with Cartesian coordinates. \( x \) will represent the radial direction, \( y \) the poloidal one, and \( z \) the toroidal one. The magnetic field can thus be written as \( \mathbf{B}(x, y, z) = B_y(x) \mathbf{\hat{e}}_y + B_z \mathbf{\hat{e}}_z \). We have arbitrarily chosen to include the radial dependency in the poloidal component exclusively. The safety factor \( q(x) = \frac{L_y B_z}{L_z B_y} \) is a measure of the magnetic field tilt angle, where \( L_i \) stands for the length of dimension \( i \).

Gyrokinetic theory consists of following the guiding center dynamics in the 5D phase space \((\mathbf{R}, v_\parallel, \mu)\), where \( \mathbf{R} \) stands for the position, \( v_\parallel \) the projection of velocity on the magnetic field direction, and \( \mu = \frac{mv^2}{2m} \) the magnetic moment with \( m \) the particle mass, \( v_\perp \) the perpendicular velocity, and \( B \) the magnetic field strength. Within our geometry, the equations of motion are:

2. The GK-engine model

For simplicity, we consider a slab-like system with Cartesian coordinates. \( x \) will represent the radial direction, \( y \) the poloidal one, and \( z \) the toroidal one. The magnetic field can thus be written as \( \mathbf{B}(x, y, z) = B_y(x) \mathbf{\hat{e}}_y + B_z \mathbf{\hat{e}}_z \). We have arbitrarily chosen to include the radial dependency in the poloidal component exclusively. The safety factor \( q(x) = \frac{L_y B_z}{L_z B_y} \) is a measure of the magnetic field tilt angle, where \( L_i \) stands for the length of dimension \( i \).

Gyrokinetic theory consists of following the guiding center dynamics in the 5D phase space \((\mathbf{R}, v_\parallel, \mu)\), where \( \mathbf{R} \) stands for the position, \( v_\parallel \) the projection of velocity on the magnetic field direction, and \( \mu = \frac{mv^2}{2m} \) the magnetic moment with \( m \) the particle mass, \( v_\perp \) the perpendicular velocity, and \( B \) the magnetic field strength. Within our geometry, the equations of motion are:

2. The GK-engine model

For simplicity, we consider a slab-like system with Cartesian coordinates. \( x \) will represent the radial direction, \( y \) the poloidal one, and \( z \) the toroidal one. The magnetic field can thus be written as \( \mathbf{B}(x, y, z) = B_y(x) \mathbf{\hat{e}}_y + B_z \mathbf{\hat{e}}_z \). We have arbitrarily chosen to include the radial dependency in the poloidal component exclusively. The safety factor \( q(x) = \frac{L_y B_z}{L_z B_y} \) is a measure of the magnetic field tilt angle, where \( L_i \) stands for the length of dimension \( i \).

Gyrokinetic theory consists of following the guiding center dynamics in the 5D phase space \((\mathbf{R}, v_\parallel, \mu)\), where \( \mathbf{R} \) stands for the position, \( v_\parallel \) the projection of velocity on the magnetic field direction, and \( \mu = \frac{mv^2}{2m} \) the magnetic moment with \( m \) the particle mass, \( v_\perp \) the perpendicular velocity, and \( B \) the magnetic field strength. Within our geometry, the equations of motion are:

\[
\begin{equation}
\frac{d\mathbf{R}}{dt} = v_\parallel b + \frac{\mu}{eB^*} \mathbf{b} \times \nabla \mathbf{B} + \frac{1}{B^*} \mathbf{b} \times \nabla \langle \phi \rangle
\end{equation}
\]

\[
\begin{equation}
\frac{dv_\parallel}{dt} = -\frac{e}{m} \mathbf{b} \cdot \nabla \langle \phi \rangle
\end{equation}
\]

\[
\begin{equation}
\frac{d\mu}{dt} = 0
\end{equation}
\]

with \( B^* = B \left( 1 + \frac{mB^*_y B_z}{eB^*_x v_\parallel} \right) \), \( \mathbf{b} \) the unitary magnetic field vector, \( e \) the particle charge, \( \phi \) the electrostatic potential, \( \langle \cdot \rangle \) standing for gyroaveraging and the prime ‘ for \( \partial/\partial x \).

The electrostatic field is solved using the quasineutrality equation with the long wavelength approximation and adiabatic electrons:

\[
\begin{equation}
\frac{en_0}{T_e} (\phi - \phi_0) - \nabla_\perp \cdot \left( \frac{n_0 B \Omega_i}{B^*} \nabla_\perp \phi \right) = \int \frac{B^*}{m} f(\mathbf{x} + \hat{\rho}_L(\mu, \alpha), v_\parallel; \mu) \, d\mu \, d\alpha \, dv_\parallel - n_0
\end{equation}
\]
where $T_e$ is the electron temperature, $\bar{\cdot}$ stands for the flux-surface averaging (i.e. in toroidal and poloidal directions), $\Omega_i$ is the ion cyclotron frequency, $\rho_L$ is the Larmor radius, and $f$ is the ion distribution function. The right-hand-side term is the gyro-averaged density.

$f$ is decomposed into an equilibrium background part $f_0$ that does not depend on time, and a perturbed part $\delta f$ that we discretize with numerical markers:

$$\delta f(\vec{R}, v_\parallel, t; \mu) = \sum_{p=1}^{N} \frac{1}{2\pi B^*} w_p(t) \delta(\vec{R} - \vec{R}_p(t)) \delta(v_\parallel - v_{\parallel p}(t)) \delta(\mu - \mu_p)$$

(5)

where $w_p$ are the marker weights. Markers are initialized to populate uniformly the phase space (velocity space is delimited by $\kappa_v v_{th}$, with $\kappa_v$ typically chosen around 5), and their weights are initialized with:

$$w_p(t = 0) = \frac{L_x L_y L_z v_{\perp p} \pi \kappa_v v_{th}(x_p)^2 B^*(x_p, v_{\parallel p})}{\Omega_p} \delta f_p(t = 0)$$

(6)

where $\Omega_p$ is the phase space volume occupied by marker $p$.

The time evolution of the weights is given by the Vlasov equation:

$$\frac{dw_p}{dt} = \Omega_p \frac{df_p}{dt} = -\Omega_p \left( \frac{d\vec{R}}{dt} \cdot \nabla f_0 + \frac{dv_\parallel}{dt} \frac{\partial f_0}{\partial v_\parallel} \right)_p$$

(7)

We use a Runge-Kutta scheme of order 4 for this evolution. Each Runge-Kutta substep is composed of the successive stages shown in Figure 1.

![Figure 1. Stages of a time step.](image-url)

The first stage is the construction of Larmor rings from guiding centers, using an adaptive number of points for each guiding center proportional to its Larmor radius $\rho_L$.

Then, one needs to build the gyro-averaged density in order to solve equation (4). This is done by depositing the charge on a finite element basis, in our case B-splines of order $p$ [8]:

$$\rho(x, y, z) = \sum_{i=1}^{N_x+p} \sum_{j=1}^{N_y+p} \sum_{k=1}^{N_z+p} \hat{\rho}_{ijk} \Lambda_i(x) \Lambda_j(y) \Lambda_k(z)$$

(8)
The magnetic field on axis is $B(r, z)$ where $r = R/a$. In order to reduce the number of particles per node, one can use MPI communication stages (called parallel move) at the end of every subdomain, allowing us to filter out numerical noise [9]. Knowing that the instability wavelengths are much longer in the parallel direction than the perpendicular one [10], one can set to 0 the modes with a poloidal wavenumber $m$ such that $|m + nq(x)| > \Delta m$, where $n$ is the toroidal wavenumber. $\Delta m$ is a parameter typically set to 5.

Alternatively, the PIF method directly builds the field Fourier transform along $y$ and $z$:

$$\rho(x, y, z) = \sum_{i=1}^{N_i+p} \sum_{n=n_{min}}^{n_{max}} \sum_{m=-nq_i-\Delta m}^{-nq_i+\Delta m} \hat{\rho}_{i,m,n}(x)e^{2\pi i (ny/L_y + nz/L_z)}$$

where $q_i$ is the safety factor evaluated at the middle of the support of spline $\Lambda_i$.

One can then solve the quasineutrality equation for the electrostatic potential, and evaluate the PIF method directly builds the field Fourier transform along $y$ and $z$.

To perform Discrete Fourier Transforms (DFTs) of the field coefficients $\{\hat{\rho}_{i,m,n}\}$ along $y$ and $z$, one needs an MPI communication stage (called parallel move) to send markers leaving a subdomain to the new one. An extra dimension of parallelism consists of cloning the subdomains among MPI tasks. The markers of each processor have to be in the right subdomain, therefore one can set to 0 the modes with a poloidal wavenumber $m$ such that $|m + nq(x)| > \Delta m$, where $n$ is the toroidal wavenumber. $\Delta m$ is a parameter typically set to 5.

Alternatively, the PIF method directly builds the field Fourier transform along $y$ and $z$:

$$\rho(x, y, z) = \sum_{i=1}^{N_i+p} \sum_{n=n_{min}}^{n_{max}} \sum_{m=-nq_i-\Delta m}^{-nq_i+\Delta m} \hat{\rho}_{i,m,n}(x)e^{2\pi i (ny/L_y + nz/L_z)}$$

where $q_i$ is the safety factor evaluated at the middle of the support of spline $\Lambda_i$.

One can then solve the quasineutrality equation for the electrostatic potential, and evaluate this field at each marker position to make them evolve according to equations [10,12] and [7].

In the PIC implementation, the parallelization scheme of the GK-engine allows for domain decomposition along the $z$ direction, i.e. splitting the grid data into subdomains distributed among MPI tasks. The markers of each processor have to be in the right subdomain, therefore one needs an MPI communication stage (called parallel move) to send markers leaving a subdomain to the new one. An extra dimension of parallelism consists of cloning the subdomains on several MPI tasks in order to reduce the number of particles per node.

3. ITG simulations

As a validation step, we choose to simulate ITG instabilities arising in a tokamak of major radius $R_0 = 77$ cm and minor radius $a = 21$ cm. Our slab model uses $L_x = a$, $L_y = \pi a$ and $L_z = 2\pi R_0$. The magnetic field on axis is $B_0 = 1$ T, and the safety factor is given by:

$$q(s) = 1.25 + 3s^2$$

where $s = x/a$ is the normalized radial coordinate.

The temperature and density profiles are given by:

$$T_i(s) = T_{i0}e^{-\kappa_{T_i}\Delta T_i\tanh\left(\frac{x-a}{\Delta x}\right)}$$

$$T_e(s) = T_{e0}e^{-\kappa_{T_e}\Delta T_e\tanh\left(\frac{x-a}{\Delta x}\right)}$$

$$n_0(s) = n_{00}e^{-\kappa_n\Delta n\tanh\left(\frac{x-a}{\Delta x}\right)}$$

with the parameters $T_{i0} = T_{e0} = 1$ keV, $\kappa_{T_i} = 2$, $\Delta T_i = 0.2$, $\kappa_{T_e} = 1$, $\Delta T_e = 0.1$, $\kappa_n = 0.5$, and $\Delta n = 0.3$.

Local linear analytical studies can show that a condition for instability is $\eta(s) := \frac{d\ln(T_e(s))}{d\ln(n(s))} > 2$ [15]. A second condition is that $k//\rho_s \ll 1$, where $\rho_s = \sqrt{mT_e/\epsilon_B}$ is the ion sound Larmor radius.

We first run linear simulations using the PIF method with 5M particles and 128 radial cubic splines in order to find the most unstable mode of the system (see Figure 2).

We find that the growth rate $\gamma$ is positive only for $n \lesssim 13$, and that the most unstable modes satisfy $m = -nq(s = 0.5)$, i.e. $k///(s = 0.5) = 0$. 

4
Figure 2. Linear growth rate of quasi field-aligned Fourier modes. Simulations with PIF method, 5M particles, and 128 radial cubic splines.

Let us now initialize the overall most unstable mode $(m, n) = (-16, 8)$, and simulate until nonlinear phase begins. We use the PIC method with $\Delta t = 10/\Omega_0$, 200M particles and $128 \times 512 \times 128$ cubic splines (see Figure 3).

Figure 3. Diagnostics of a nonlinear simulation with PIC method, $\Delta t = 10/\Omega_0$, 200M particles, $128 \times 512 \times 128$ cubic splines. Fourier filter only keeps $0 \leq n \leq 25$ and $|m + nq| \leq 5$. 
We see that starting at \( t \approx 10^4 / \Omega_0 \), an outward radial heat flux flattens the temperature profile. The kinetic energy is converted into potential energy, and the balance is satisfied.

4. Convergences studies
Having verified that our simulations are physically meaningful, we want to investigate the numerical convergence of the results.

**Figure 4.** Coefficient of variation in time of the linear growth rate as a function of the number of particles. Simulations with PIF, \( \Delta t = 200 / \Omega_0 \), and 128 radial cubic splines.

**Figure 5.** Mass non-conservation at \( t_{end} = 4 \cdot 10^4 \Omega_0 \) of a nonlinear simulation as a function of the time step. Simulations with PIF, 100M particles, and 128 radial quadratic splines.

In Figure 4 we look at the linear growth rate of a single mode as a function of the number of particles. As we find that the instantaneous growth rate temporally varies around an average value, we plot the coefficient of variation of this oscillation, i.e. the ratio of the standard deviation to the average. This quantity is found to be inversely proportional to the square root of the number of particles, reflecting a statistical noise.

Figure 5 reports the relative variation of the total mass \( \int f d^3x d^3v \) at the end of the nonlinear simulation. It converges with \( \Delta t^4 \), consistent with the time integrator we used (fourth order Runge-Kutta).

5. Numerical optimization
We now run a benchmark of the different implementations of the code, on CPU and on GPU, for a simulation of a unique Fourier mode (see Figure 6). On CPU, we can either use 8 MPI tasks, or a single task with 8 OpenMP threads. Unsorted and sorted versions are compared: loops on particles are better vectorized in the latter version because sorting increases data locality and avoids random memory access. Moreover, for the multithreaded implementations, sorting enables the multithreaded outer loop to be on the grid cells.

One can see that the most important steps are particle-to-grid and grid-to-particle operations. On CPU implementations, the sorting is more expensive than the speed-up it brings, but on GPU it gives an overall gain of a factor 2 because it avoids conflicts between multiples threads accessing the same grid point in memory (race condition). The sorting routine could still be optimized using another algorithm [3], or taking advantage of the fact that Larmor rings lie on the same poloidal plane.

For comparison, we also show the best implementation of the PIF method, namely on GPU with sorting. It is twice faster than the best PIC implementation.
Figure 6. Timings comparison of full MPI, OpenMP, and OpenACC implementations, with or without sorting, on 1 node of Piz Daint. Parameters are 2M particles and $64 \times 64 \times 32$ cubic splines for PIC, and 1 mode for PIF.

Then, we run the same benchmarking on a bigger case with $26 \times 11$ Fourier modes (Figure 7).

Figure 7. Same comparison as Figure 6 with 32 subdomains (32 nodes). Parameters are 32 subdomains, 100M particles, $128 \times 512 \times 128$ cubic splines for PIC, and 32 clones, $26 \times 11$ modes for PIF (shown on a different scale).
This time, sorting particles is always worth it, even on CPU implementations. We also see the parallel move timing, slower on the GPU version because it requires a lot of host-device communications. The overall best timing is achieved by the GPU implementation with sorting.

The PIF method is now one order of magnitude slower than PIC, because all the Fourier modes need to be evaluated at each marker position. Although optimization is still under development, it seems that this method will only be advantageous for a restricted number of Fourier modes.

6. Parallel scalability
Lastly, we run a weak scaling to get an idea of the behavior of the timings on a large scale cluster. The length of the domain and the time step are fixed, but the toroidal resolution and the number of particles are increased proportionally to the number of nodes.

Figure 8. Weak scaling of sorted OpenACC version with 16 clones and 1 to 256 subdomains. Simulation parameters are $10^6$ guiding centers per node, $128 \times 512 \times (4 \text{ grid cells per subdomain})$, and cubic interpolation. Simulations were run on Piz Daint.

The result, shown on Figure 8, clearly demonstrates that most of the routines scale perfectly well (over 98% efficiency).

The exceptions are the field solver and the parallel move. Indeed, when increasing the number of subdomains for a fixed-size domain, the number of escaping particles increases as well, resulting in complete non-scalability. For a large number of subdomains, a large fraction of the particles needs to be sent to other subdomains. We are currently considering adding another direction of domain decomposition to counterbalance this problem.

As for the field solver, the non-scalability comes from the parallel data transposes required to perform Discrete Fourier Transforms (DFTs). In the future, we will investigate an idea to compute partial local DFTs and communicate compacted fields instead of the full real grid.

7. Conclusions
We have demonstrated the proper functioning of the GK-engine both physically, through linear and nonlinear ITG simulations, and numerically through convergence studies. We then compared performance of different implementations (MPI cloning versus OpenMP multithreading, sorting or not) and different architectures (CPU versus GPU) in two reference cases (small-sized on a single node and medium-sized on several nodes). We found that on CPU, sorting can be more
costly than the gain it brings, in particular for a small grid, but that it is always profitable on GPU because it avoids race conditions.

We also found that the PIF performance depends greatly on the number of Fourier modes we want to represent. Even though it is not yet fully optimized, we know that its application will be limited to systems where the number of Fourier modes is much smaller than the real-space grid size. It could for instance be interesting for toroidal Alfvén eigenmodes simulations, as one typically needs only one toroidal mode and five or seven poloidal modes, eventually adding the $n = 0$ modes to study the coupling to zonal flows and geodesic acoustic modes.

The knowledge we acquired with the GK-engine will help in the refactoring of the larger gyrokinetic PIC code ORB5 [11][12][13].

Acknowledgements

This work is partly funded by the Swiss National Science Foundation and the Platform for Advanced Scientific Computing. It has been carried out within the framework of the EUROfusion Consortium and has received funding from the Euratom research and training programme 2014-2018 under grant agreement No 633053. The views and opinions expressed herein do not necessarily reflect those of the European Commission.

References

[1] Hariri F, Tran T M, Jocksch A, Lanti E, Progsch J, Messmer P et al. 2016 A portable platform for accelerated PIC codes and its application to GPUs using OpenACC Comput. Phys. Commun. (in press, available online on May 21 2016)
[2] Lanti E et al. 2016 Towards porting a gyrokinetic PIC code on many-core platforms Comput. Phys. Commun. (submitted)
[3] Jocksch A, Hariri F, Tran T M, Brunner S, Gheller C and Villard L 2015 A bucket sort algorithm for the particle-in-cell method on manycore architectures International Conference on Parallel Processing and Applied Mathematics 43-52
[4] Brunner S, Valeo E and Krommes J A 1999 Collisonal delta-f scheme with evolving background for transport time scale simulations Phys. Plasmas 6(12) 4504
[5] McMillan B F, Jolliet S, Tran T M, Villard L, Bottino A and Angelino P 2008 Long global gyrokinetic simulations: Source terms and particle noise control Phys. Plasmas 15(5) 052308
[6] Dominski J 2016 Development and application of Eulerian and Particle-In-Cell gyrokinetic codes for studying the effect of non-adiabatic passing electron dynamics on microturbulence PhD thesis EPFL
[7] Burau H, Widera R, Honig W, Jackelend G, Debux A, Kluge T et al. 2010 PIConGPU: A fully relativistic particle-in-cell code for a GPU cluster IEEE Transactions on Plasma Science 38(10) 2831-9
[8] De Boor C 1978 A practical guide to splines New York: Springer-Verlag 27 325
[9] McMillan B F, Jolliet S, Bottino A, Angelino P, Tran T M and Villard L 2010 Rapid Fourier space solution of linear partial integro-differential equations in toroidal magnetic confinement geometries Comput. Phys. Commun. 181(4) 715-19
[10] Jolliet S, McMillan B F, Villard L, Vernay T, Angelino P, Tran T M et al. 2012 Parallel filtering in global gyrokinetic simulations J. Comput. Phys. 231(3) 745-58
[11] Tran T M, Appert K, Fivaz M, Jost G, Vaclavik J and Villard L 1999 Global gyrokinetic simulation of ion-temperature-gradient-driven instabilities using particles Theory of Fusion Plasmas, Int. Workshop (Editrice Compositori, Bologna, Societa Italiana di Fisica) 45
[12] Jolliet S, Bottino A, Angelino P, Hatzky R, Tran T M, McMillan B F et al. 2007 A global collisionless PIC code in magnetic coordinates Comput. Phys. Commun. 177(5) 409-25
[13] Biancalani A, Bottino A, Lauber P and Zarzoso D 2014 Numerical validation of the electromagnetic gyrokinetic code NEMORB on global axisymmetric modes Nucl. Fusion 54(10) 104004
[14] Abel I G, Plunk G G, Wang E, Barnes M, Cowley S C, Dorland W and Schekochihin A A 2013 Multiscale gyrokinetics for rotating tokamak plasmas: fluctuations, transport and energy flows Reports on Progress in Physics 76(11) 116201
[15] Grandgirard V, Brunetti M, Bertrand P, Besse N, Garbet X, Ghendrih P et al. 2006 A drift-kinetic semi-Lagrangian 4D code for ion turbulence simulation J. Comput. Phys. 217(2) 395–423
[16] Idomura Y, Tokuda S and Wakatani M 1999 Gyrokinetic theory of slab ion temperature gradient mode in negative shear tokamaks Phys. Plasmas 6(12) 4658–4671