HAL
archives-ouvertes.fr

Some parallel algorithms for the Quasineutrality solver of GYSELA

Guillaume Latu, Virginie Grandgirard, Nicolas Crouseilles, Radoin Belaouar, Eric Sonnendrücker

▶ **To cite this version:**

Guillaume Latu, Virginie Grandgirard, Nicolas Crouseilles, Radoin Belaouar, Eric Sonnendrücker. Some parallel algorithms for the Quasineutrality solver of GYSELA. [Research Report] RR-7591, INRIA. 2011, pp.15. <inria-00583521>

HAL Id: inria-00583521
https://hal.inria.fr/inria-00583521

Submitted on 5 Apr 2011

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Some parallel algorithms for the Quasineutrality solver of GYSELA

G. Latu — V. Grandgirard — N. Crouseilles —

R. Belaouar — E. Sonnendrücker

N° 7591

Avril 2011

Domaine 1

apport de recherche

ISSN 0249-6399 ISRN INRIA/RR--7591--FR+ENG

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

INRIA

centre de recherche
NANCY - GRAND EST

Some parallel algorithms for the Quasineutrality solver of GYSELA

G. Latu * †, V. Grandgirard *, N. Crouseilles † ,
R. Belaouar †, E. Sonnendrücker †

Domaine : Mathématiques appliquées, calcul et simulation
Équipes-Projets CALVI

Rapport de recherche n° 7591 — Avril 2011 — 15 pages

Abstract: The very first parallelizations of the quasineutrality Poisson solver used in the GYSELA code are presented here. We investigate some numerical schemes. For each considered scheme, we propose an algorithmic decomposition in term of parallel computations and inter-processor communications. A set of benchmarks on a parallel machine has permitted to evaluate the performance of the different versions of the Quasineutrality solver.

Key-words: Quasineutrality solver, Gyrokinetics, MPI

* CEA Cadarache, 13108 Saint-Paul-les-Durance Cedex
† INRIA Nancy-Grand Est & Université de Strasbourg, 7 rue Descartes, 67000 Strasbourg

Centre de recherche INRIA Nancy – Grand Est
LORIA, Technopôle de Nancy-Brabois, Campus scientifique,
615, rue du Jardin Botanique, BP 101, 54602 Villers-Lès-Nancy
Téléphone : +33 3 83 59 30 00 — Télécopie : +33 3 83 27 83 19

Quelques algorithmes parallèle pour le solveur Poisson Quasi-neutre de GYSELA

Résumé : Les toutes premières parallelisations du solveur Poisson Quasi Neutre utilisées dans le code GYSELA sont présentées ici. Les schémas numériques qui ont été envisagés pour ce solveur Poisson sont décrits. Pour chaque schéma, nous proposons une décomposition en terme d'étapes de calculs et de communications inter-processeurs. Des benchmarks sur machine parallèle ont permis d'évaluer les performances de ces différentes versions du solveur Poisson Quasi neutre.

Mots-clés : Solveur Quasi neutre, Gyrocinétique, MPI

1 Introduction

Nowadays, the modelization of magnetized plasmas is a key issue for controlled thermonuclear fusion. In practice, the study of such plasmas requires to solve the Maxwell equations coupled to the computation of the plasma response. Different ways are possible to compute this response: the fluid or the kinetic description. Obviously solving the full Vlasov equation (kinetic description) involves the discretization of the six-dimensional phase space (3D space, 3D velocity), which is a challenging problem. On the other side, the fluid approach is not able to capture kinetic effects such as nonlinear wave-particle interaction (see [2, 3]).

In the context of strongly magnetized plasmas, gyrokinetics enable to recast the Vlasov equation into a 5D equation in which the fast gyromotion is averaged. But the particle information is not lost since the finite Larmor radius effects are taken into account through the gyroaverage operator in the gyrokinetic Poisson equation. The physical model is then based on a gyrokinetic equation for the ions, whereas an adiabatic response is currently assumed for the electrons [1, 7, 14]. These equations are supplemented by the quasineutrality equation for the electric potential (*i.e.* the gyrokinetic Poisson equation in which quasineutrality is assumed). This so-obtained gyrokinetic model can be used to numerically study ion temperature gradients (ITG) instabilities for example. To that purpose, two classes of methods are used to approximate such a model; the most popular method is the Particle In Cell (PIC) method (see [4, 8, 10]), but Vlasov type simulations are also conceivable in this context (see [5]). Both methods require the computation of the electric potential on a grid of the physical space. Hence, a fast and efficient way of solving the quasineutrality equation seems to be important.

A parallel gyrokinetic code needs to couple a parallel vlasov solver with a parallel QN solver to be an efficient method. The role of a quasi-neutrality (QN) solver is to give the potential Φ taking as an input the particles density, whereas the Vlasov solver moves the particle density forward in time. In this work, we focus on the study of the QN equation.

Moreover, the numerical resolution of the QN equation requires the solving of a 3 dimensional equation, involving a non local term which penalizes an efficient parallel implementation (see [9]).

One of the possible ways to numerically treat the quasi-neutrality equation consists in the use of Fast Fourier Transform, other choices are multigrid or use of a direct solver for example. Even if the FFT approach is not adapted to general geometries [11] (*i.e.* realistic tokamak geometry), in periodic direction they remains a fast, simple and accurate method. Hence, we propose here a new solver which is based on a decomposition of the electric potential. First, one solves N_φ independent $2D$ Helmholtz type equations (N_φ is the number of points in the φ direction). Second, a simple $1D$ ordinary differential equation has to be solved on the average of the electric potential on the magnetic surface.

An adapted communication scheme is introduced to reduce the communication cost between the Vlasov solver and the QN solver (the two main parts of a gyrokinetic code). As a consequence, a global parallelized gyrokinetic solver is then obtained, the performance of which are presented at the end of the paper.

The rest of the paper is organized as follows: In a first part, we intend to present the QN equation together with the gyroaverage operator. Then, three

different methods are exposed to approximate the QN equation. Finally various numerical tests have been implemented to compare these numerical methods, and their influence on the coupling with a full-f gyrokinetic Vlasov code: GYSELA [5].

2 Gyroaveraging and gyrokinetic field equations

2.1 Quasineutrality equation

In tokamak configurations, the plasma quasineutrality approximation is currently assumed ([5, 7]). This leads to $n_i = n_e$ where n_i (resp. n_e) is the ionic (resp. electronic) density. On the one side, electron inertia is ignored, which means that an adiabatic response of electrons are supposed. On the other side, the ionic density splits into two parts. Using the notation $\nabla_\perp = (\partial_r, \frac{1}{r}\partial_\theta)$, the so-called linearized polarization density n_{pol} writes

$$n_{pol}(r,\theta,\varphi) = -\nabla_\perp \cdot \left[\frac{n_0(r)}{B_0}\nabla_\perp \Phi(r,\theta,\varphi)\right],$$

where n_0 is the equilibrium density and B_0 the magnetic field at the magnetic axis. Second, the guiding-center density n_{Gi} is

$$n_{Gi}(r,\theta,\varphi) = 2\pi \int B(r,\theta)d\mu \int dv_{//} J_0(k_\perp\sqrt{2\mu})\bar{f}(r,\theta,\varphi,v_{//},\mu), \qquad (1)$$

where B is the magnetic field, $\bar{f}$ denotes the ionic distribution function evolving through a Vlasov type equation, $v_{//}$ the parallel velocity, μ the magnetic momentum and J_0 is the Bessel function.

Hence, the QN equation can be written in dimensionless variables

$$-\frac{1}{n_0(r)}\nabla_\perp \cdot \left[\frac{n_0(r)}{B_0}\nabla_\perp \Phi(r,\theta,\varphi)\right] + \frac{1}{T_e(r)}\left[\Phi(r,\theta,\varphi) - \langle\Phi\rangle_{\theta,\varphi}(r)\right] = \tilde{\rho}(r,\theta,\varphi) \qquad (2)$$

with T_e the electronic temperature and where the definition of $\tilde{\rho}$ is given by

$$\tilde{\rho}(r,\theta,\varphi) = \frac{2\pi}{n_0(r)} \int B(r,\theta)d\mu \int dv_{//} J_0(k_\perp\sqrt{2\mu})(\bar{f} - \bar{f}_{eq})(r,\theta,\varphi,v_{//},\mu). \qquad (3)$$

In this last equation, $\bar{f}_{eq}$ denotes an electronic local Maxwellian equilibrium, and $\langle\rangle_{\theta,\varphi}$ the average onto the variables θ, φ.

The QN solver includes two computation parts. First, the function $\tilde{\rho}$ is derived taking as input function $\bar{f}$. Specific methods, that will be described afterwards, are used to evaluate the gyroaverage operator J_0 on $(\bar{f} - \bar{f}_{eq})$ in Eq. (3).
Second, the 3D potential Φ is found in computing discrete fourier transforms of $\tilde{\rho}$, followed by solving of tridiagonal systems and inverse fourier transforms. For this step, two kinds of solving procedure are considered in this paper. Their descriptions and performance evaluation are the aim of this paper.

2.2 Gyroaverage operator

Let us now detail the computation of $\tilde{\rho}$. Starting from the ionic guiding center distribution function $\bar{f} = \bar{f}(r,\theta,\varphi,v_{//},\mu)$, we can obtain the ionic density on

the particle coordinates thanks to a gyroaverage operator. After some computations, this operator makes appear the Bessel operator and leads to (3). In the sequel, we focus on the computation of $g = g(r, \theta, \varphi)$ satisfying

$$\bar{g}(r, \theta, \varphi) = J_0(k_\perp \sqrt{2\mu}) g(r, \theta, \varphi) \tag{4}$$

The numerical resolution of such a problem is based on a Padé approximation which is currently performed for the Bessel function J_0

$$J_0(k_\perp \sqrt{2\mu}) \approx \frac{1}{1 + \frac{(k_\perp \sqrt{2\mu})^2}{4}}. \tag{5}$$

This approximation is correct for small wavenumbers $k_\perp$ and keeps J_0 finite in the opposite limit $|k_\perp| \to +\infty$. Injecting the Padé approximation (5) in (4), a Fourier transform enables to use the equivalence between $(ik_\perp)$ and $\nabla_\perp$. Finally, we can obtain g by solving the following implicit equation

$$\left[1 - \frac{\mu B}{2 {\omega_c}^2 m_i} \nabla_\perp^2\right] \bar{g}(r, \theta, \varphi) = g(r, \theta, \varphi).$$

2.3 Numerical methods for gyroaveraging

Let us consider a function f which is defined on a global domain $[r_0, r_{Nr}] \subset \mathbb{R}$. In the following, we will use the notation $r_i = r_0 + i\,dr$, where dr is the mesh size: $dr = (r_{Nr} - r_0)/(Nr + 1)$.

Let us now restrict the study of $g : r \to g(r)$ on an interval $[r_0, r_{Nr}]$, $Nr \in \mathbb{N}$, where all r_i are known. Our goal is to get the gyroaverage $\bar{g}$ from a known g function.

Let us do a Fourier transform in variable θ. Each k fourier mode of $\bar{g}$ is the solution of the equation of the following equation (after some approximations [5, 6]):

$$\left[1 - \frac{\mu B_0}{2\omega_c^2 m_i} \left(\frac{\partial^2}{\partial r^2} - \frac{k^2}{r^2}\right)\right] \bar{g}^{\,k}(r, \varphi) = g^{\,k}(r, \varphi)$$

We assume that we have Neumann boundary conditions, and that ${g_1}^m = {g_2}^m$ et ${g_{Nr}}^m = {g_{Nr-1}}^m$. So, solving the equation requires, for each φ, the inversion of the following tridiagonal system:

$$\begin{pmatrix} a+b_2 & a & 0 & & \\ a & b_3 & a & 0 & \\ & \ddots & \ddots & \ddots & \\ & 0 & a & b_{Nr-2} & a \\ & & 0 & a & a+b_{Nr-1} \end{pmatrix} \begin{pmatrix} \bar{g}_2^k(\varphi) \\ \bar{g}_3^k(\varphi) \\ \vdots \\ \bar{g}_{Nr-2}^k(\varphi) \\ \bar{g}_{Nr-1}^k(\varphi) \end{pmatrix} = \begin{pmatrix} g_2^k(\varphi) \\ g_3^k(\varphi) \\ \vdots \\ g_{Nr-2}^k(\varphi) \\ g_{Nr-1}^k(\varphi) \end{pmatrix} \tag{6}$$

with

$$\begin{cases} a & = -\frac{B_0}{2\omega_c^2 m_i} \mu \frac{1}{\Delta r^2} \\ b_j & = 1 + \frac{B_0}{2\omega_c^2 m_i} \mu \left(\frac{2}{\Delta r^2} + \frac{k^2}{r_j^2}\right) \end{cases} \tag{7}$$

In the code, the hypothesis $B_0 = 2\omega_c^2 m_i$ is done in a set of simplified geometries where B is considered constant in the poloïdal plane. It gives the following system to solve:

$$\begin{pmatrix} \alpha+\beta_2 & \alpha & 0 & & \\ \alpha & \beta_3 & \alpha & 0 & \\ & \ddots & \ddots & \ddots & \\ & 0 & \alpha & \beta_{Nr-2} & \alpha \\ & & 0 & \alpha & \alpha+\beta_{Nr-1} \end{pmatrix} \begin{pmatrix} \bar{g}_2^k(\varphi) \\ \bar{g}_3^k(\varphi) \\ \vdots \\ \bar{g}_{Nr-2}^k(\varphi) \\ \bar{g}_{Nr-1}^k(\varphi) \end{pmatrix} = \begin{pmatrix} g_2^k(\varphi) \\ g_3^k(\varphi) \\ \vdots \\ g_{Nr-2}^k(\varphi) \\ g_{Nr-1}^k(\varphi) \end{pmatrix} \tag{8}$$

with

$$\begin{cases} \zeta & = \frac{1}{\Delta r^2} \\ \alpha & = -\frac{\mu}{2}\,\zeta \\ \beta_j & = 1 + \frac{\mu}{2}\,(2\,\zeta + \frac{k^2}{r_j^2}) \end{cases} \tag{9}$$

The solving of system (8) allows one to apply the gyroaverage operator on function g. A LU decomposition is done once for the tridiagonal matrix of system (8). The matrices L and U are used every time a gyroaveraging is needed with a computational complexity of $\Theta(N_r)$ for the solving procedure and $\Theta(N_r\,log(N_\theta))$ for the fourier transforms.

2.4 2D Fourier transforms method

The QN equation (2) could be solved in the Fourier space along the two periodic directions (θ and φ) taking as input the values of $\tilde{\rho}$. This method is presented in [5]. Let Φ and $\tilde{\rho}$ be defined in terms of Fourier expansion as:

$$\begin{array}{ll} \Phi(r,\theta,\varphi) = & \sum_u \sum_w \hat{\Phi}^{u,w}(r) e^{i\,u\,\theta} e^{i\,w\,\varphi} \\ \tilde{\rho}(r,\theta,\varphi) = & \sum_u \sum_w \hat{\rho}^{u,w}(r) e^{i\,u\,\theta} e^{i\,w\,\varphi} \end{array} \tag{10}$$

In this wave number representation, the QN equation could be written as:

$$-\frac{\partial^2 \hat{\Phi}^{u,w}(r)}{\partial r^2} - [\frac{1}{r} + \frac{1}{n_0(r)}\frac{\partial n_0(r)}{\partial r}]\frac{\partial \hat{\Phi}^{u,w}(r)}{\partial r} + \left(\frac{u^2}{r^2} + \frac{(1-\delta_{u=0,w=0})}{Z_i\,T_e(r)}\right)\hat{\Phi}^{u,w}(r) = \hat{\rho}^{u,w}(r)$$

Let N_r be the number of radial points. We want to compute each $\hat{\Phi}^{u,w}$ such as:

$$\begin{pmatrix} m_2 & o_2 & 0 & & \\ p_3 & m_3 & o_3 & 0 & \\ & \ddots & \ddots & \ddots & \\ & 0 & p_{Nr-2} & m_{Nr-2} & o_{Nr-2} \\ & & 0 & p_{Nr-1} & m_{Nr-1} \end{pmatrix} \begin{pmatrix} \hat{\Phi}_2^{u,w}(r) \\ \hat{\Phi}_3^{u,w}(r) \\ \vdots \\ \hat{\Phi}_{Nr-2}^{u,w}(r) \\ \hat{\Phi}_{Nr-1}^{u,w}(r) \end{pmatrix} = \begin{pmatrix} \hat{\rho}_2^{u,w}(r) \\ \hat{\rho}_3^{u,w}(r) \\ \vdots \\ \hat{\rho}_{Nr-2}^{u,w}(r) \\ \hat{\rho}_{Nr-1}^{u,w}(r) \end{pmatrix} \tag{11}$$

with

$$\begin{cases} p_i & = -(\frac{1}{\Delta r^2} - \frac{\alpha(r_i)}{2\Delta r}) \quad \text{where} \quad \alpha(r_i) = \frac{1}{r} + \frac{1}{n_0(r_i)}\frac{d\,n_0(r_i)}{dr} \\ m_i & = \frac{2}{\Delta r^2} + \frac{m^2}{r_i^2} + (1-\delta_{u=0,w=0})\frac{1}{Z_i\,T_e(r_i)} \\ o_i & = -(\frac{1}{\Delta r^2} + \frac{\alpha(r_i)}{2\Delta r}) \\ \hat{\rho}_i^{u,w} & = \hat{\rho}^{u,w}(r_i) \end{cases} \tag{12}$$

We assume here vanishing Dirichlet boundary conditions in r direction[1]: $\hat{\Phi}_1^{u,w}(\varphi) = \hat{\Phi}_{Nr}^{u,w}(\varphi) = 0$. The system (11) is solved using a LU decomposition of the matrix (the decomposition is computed only once).

[1] in the GYSELA code, neumann boundary conditions are also available.

2.5 1D Fourier transforms method

The previous formulation of the QN solver using 2D fourier transforms has a main drawback from a parallelization point of view. In order to solve the equation (11) for a given couple (u, w), one must compute 2D FFTs that require to know all values of $\tilde{\rho}$ in dimensions θ and φ. Then, systems are solved in dimension r. There is no simple domain decomposition of $\tilde{\rho}$ that leads to the design of a QN solver with a good load balance and that induces few communication with such 2D FFTs. The only possible algorithms perform global transpositions of $\hat{\rho}$ and $\hat{\Phi}$ before or after FFT transforms (as will be shown in algorithm 1). These transpositions constitute an overhead that does not scale well on many processors. So, we are looking for another method that does not need to consider all values in one of the three dimensions, and then uncouples the dimensions (r, θ, φ).

The main advantages of the method that follows (from a work distribution point of view) is to consider only 1D FFTs in θ dimension and to uncouple hardly all computations in φ direction.

The equation (2) averaged on (θ, φ) gives :

$$-\frac{\partial^2 \langle\Phi\rangle_{\theta,\varphi}(r)}{\partial r^2} - [\frac{1}{r} + \frac{1}{n_0(r)}\frac{\partial n_0(r)}{\partial r}]\frac{\partial \langle\Phi\rangle_{\theta,\varphi}(r)}{\partial r} = \langle\tilde{\rho}\rangle_{\theta,\varphi}(r) \qquad (13)$$

A fourier transform in θ direction gives:

$$\begin{array}{ll} \Phi(r,\theta,\varphi) = & \sum_u \hat{\Phi}^u(r,\varphi)e^{i\,u\,\theta} \\ \tilde{\rho}(r,\theta,\varphi) = & \sum_u \hat{\rho}^u(r,\varphi)e^{i\,u\,\theta} \end{array} \qquad (14)$$

The equation (2) could be rewritten as:

for $u > 0$:

$$-\frac{\partial^2\hat{\Phi}^u(r,\varphi)}{\partial r^2} - [\frac{1}{r} + \frac{1}{n_0(r)}\frac{\partial n_0(r)}{\partial r}]\frac{\partial\hat{\Phi}^u(r,\varphi)}{\partial r} + \frac{u^2}{r^2}\hat{\Phi}^u(r,\varphi) + \frac{\hat{\Phi}^u(r,\varphi)}{Z_i\,T_e(r)} = \hat{\rho}^u(r,\varphi) \qquad (15)$$

for $u = 0$:

$$\frac{\partial^2 \langle\Phi\rangle_\theta(r,\varphi)}{\partial r^2} - [\frac{1}{r} + \frac{1}{n_0(r)}\frac{\partial n_0(r)}{\partial r}]\frac{\partial \langle\Phi\rangle_\theta(r,\varphi)}{\partial r} + \frac{\langle\Phi\rangle_\theta(r,\varphi) - \langle\Phi\rangle_{\theta,\varphi}(r)}{Z_i\,T_e(r)} = \langle\tilde{\rho}\rangle_\theta(r,\varphi) \qquad (16)$$

The equation (18) allows one to directly find out the value of $\langle\Phi\rangle_{\theta,\varphi}(r)$ from the data $\langle\tilde{\rho}\rangle_{\theta,\varphi}(r)$. Let us define the function $\Upsilon(r,\theta,\varphi)$ as $\Phi(r,\theta,\varphi) - \langle\Phi\rangle_{\theta,\varphi}(r)$. Substracting equation (18) to equation (21) leads to

$$-\frac{\partial^2 \langle\Upsilon\rangle_\theta(r,\varphi)}{\partial r^2} - [\frac{1}{r} + \frac{1}{n_0(r)}\frac{\partial n_0(r)}{\partial r}]\frac{\partial \langle\Upsilon\rangle_\theta(r,\varphi)}{\partial r} + \frac{\langle\Upsilon\rangle_\theta(r,\varphi)}{Z_i\,T_e(r)} = \langle\tilde{\rho}\rangle_\theta(r,\varphi) - \langle\rho\rangle_{\theta,\varphi}(r) \qquad (17)$$

Let us notice that $\hat{\Phi}^0{}_{(r,\varphi)} = \langle\Upsilon\rangle_\theta{}_{(r,\varphi)} + \langle\Phi\rangle_{\theta,\varphi}{}_{(r)}$. So, the solving of equations (18) and (22) allows one to compute $\langle\Phi\rangle_{\theta,\varphi}{}_{(r)}$, $\langle\Upsilon\rangle_\theta{}_{(r,\varphi)}$ and $\hat{\Phi}^0{}_{(r,\varphi)}$ from the quantities $\langle\tilde{\rho}\rangle_\theta{}_{(r,\varphi)}$ and $\langle\tilde{\rho}\rangle_{\theta,\varphi}{}_{(r)}$.

Then, the equation (20) is sufficient to compute $\hat{\Phi}^{u>0}(r,\varphi)$ from $\tilde{\rho}$. Let us notice that one has the equality tnotein the GYSELA code, neumann boundary conditions are also available.: $\hat{\Phi}_1^{u,w}(\varphi) = \hat{\Phi}_{Nr}^{u,w}(\varphi) = 0$. The system (11) is solved using a LU decomposition of the matrix computed only once.

2.6 1D Fourier transforms method

The previous formulation of the QN solver using 2D fourier transforms has a main drawback from a parallelization point of view. In order to solve the equation (11) for a given couple (u, w), one must compute 2D FFTs that require to know all values of $\tilde{\rho}$ in dimensions θ and φ. Then, systems are solved in dimension r. There is no simple domain decomposition of $\tilde{\rho}$ that leads to the design of a QN solver with a good load balance and that induces few communication with such 2D FFTs. The only possible algorithms perform global transpositions of $\hat{\rho}$ and $\hat{\Phi}$ before or after FFT transforms (as will be shown in algorithm 1). These transpositions constitute an overhead that do not scale well on many processors. So, we are looking for another method that does not need to consider all values in one of the three dimensions, and then uncouples the dimensions (r, θ, φ).

The main advantages of the method that follows (from a work distribution point of view) is to consider only 1D FFTs in θ dimension and to uncouple hardly all computations in φ direction.

The equation (2) averaged on (θ, φ) gives :

$$-\frac{\partial^2 \langle\Phi\rangle_{\theta,\varphi}(r)}{\partial r^2} - [\frac{1}{r} + \frac{1}{n_0(r)}\frac{\partial n_0(r)}{\partial r}]\frac{\partial \langle\Phi\rangle_{\theta,\varphi}(r)}{\partial r} = \langle\tilde{\rho}\rangle_{\theta,\varphi}(r) \qquad (18)$$

A fourier transform in θ direction gives:

$$\begin{array}{ll} \Phi(r,\theta,\varphi) = & \sum_u \hat{\Phi}^u(r,\varphi)e^{i\,u\,\theta} \\ \tilde{\rho}(r,\theta,\varphi) = & \sum_u \hat{\rho}^u(r,\varphi)e^{i\,u\,\theta} \end{array} \qquad (19)$$

The equation (2) could be rewritten as:

for $u > 0$:

$$-\frac{\partial^2\hat{\Phi}^u(r,\varphi)}{\partial r^2} - [\frac{1}{r} + \frac{1}{n_0(r)}\frac{\partial n_0(r)}{\partial r}]\frac{\partial\hat{\Phi}^u(r,\varphi)}{\partial r} + \frac{u^2}{r^2}\hat{\Phi}^u(r,\varphi) + \frac{\hat{\Phi}^u(r,\varphi)}{Z_i\,T_e(r)} = \hat{\rho}^u(r,\varphi) \qquad (20)$$

for $u = 0$:

$$\frac{\partial^2 \langle\Phi\rangle_\theta(r,\varphi)}{\partial r^2} - [\frac{1}{r} + \frac{1}{n_0(r)}\frac{\partial n_0(r)}{\partial r}]\frac{\partial \langle\Phi\rangle_\theta(r,\varphi)}{\partial r} + \frac{\langle\Phi\rangle_\theta(r,\varphi) - \langle\Phi\rangle_{\theta,\varphi}(r)}{Z_i\,T_e(r)} = \langle\tilde{\rho}\rangle_\theta(r,\varphi) \qquad (21)$$

The equation (18) allows one to directly find out the value of $\langle\Phi\rangle_{\theta,\varphi}(r)$ from the data $\langle\tilde{\rho}\rangle_{\theta,\varphi}(r)$. Let us define the function $\Upsilon(r,\theta,\varphi)$ as $\Phi(r,\theta,\varphi) - \langle\Phi\rangle_{\theta,\varphi}(r)$. Substracting equation (18) to equation (21) leads to

$$-\frac{\partial^2 \langle\Upsilon\rangle_\theta(r,\varphi)}{\partial r^2} - [\frac{1}{r} + \frac{1}{n_0(r)}\frac{\partial n_0(r)}{\partial r}]\frac{\partial \langle\Upsilon\rangle_\theta(r,\varphi)}{\partial r} + \frac{\langle\Upsilon\rangle_\theta(r,\varphi)}{Z_i\,T_e(r)} = \langle\tilde{\rho}\rangle_\theta(r,\varphi) - \langle\rho\rangle_{\theta,\varphi}(r) \qquad (22)$$

Let us notice that $\hat{\Phi}^0(r,\varphi) = \langle\Upsilon\rangle_\theta(r,\varphi) + \langle\Phi\rangle_{\theta,\varphi}(r)$. So, the solving of equations (18) and (22) allows one to compute $\langle\Phi\rangle_{\theta,\varphi}(r)$, $\langle\Upsilon\rangle_\theta(r,\varphi)$ and $\hat{\Phi}^0(r,\varphi)$ from the quantities $\langle\tilde{\rho}\rangle_\theta(r,\varphi)$ and $\langle\tilde{\rho}\rangle_{\theta,\varphi}(r)$.

Then, the equation (20) is sufficient to compute $\hat{\Phi}^{u>0}(r,\varphi)$ from $\tilde{\rho}$. Let us notice that one has the equality $\hat{\Phi}^{u>0}(r,\varphi) = \hat{\Upsilon}^{u>0}(r,\varphi)$. The different equations are solved using a LU decomposition in the same way that we do in the previous subsection with system (11). Moreover, variable φ acts as a parameter in equation (20), allowing computations to be parallelized.

3 Algorithms for the QN solver

Hereafter, three algorithms are proposed to solve the QN equation. These algorithms wil be denoted by `2d_fft`, `1d_fft`, `p1d_fft`.
To distribute computation of part 1, all aglorithms use a single simple formulation to obtain $\tilde{\rho}$ data structure from integrals on $\bar{f}$.
However, several solutions based on techniques described in the previous subsections (2.4, 2.6) are used to compute the part 2 that derives Φ. Each solution will be presented in a peculiar subsection. We assume two main hypothesis concerning the data distribution: 1) initially each processor knows the value of a block $\bar{f}(r = block, \theta = block, \varphi = *, v_{//} = *, \mu = value)$, 2) at the end the data Φ must be known on all processors. In forthcoming works, we should try to remove the latter hypothesis to only produce small blocks such as $\Phi(r = block, \theta = block, \varphi = *)$ on each processor. But depandancies in the GYSELA code (which encapsulates our QN solver) prevent us to do so at the present day.

3.1 Partial parallel algorithm with 2D FFT

The algorithm that follows (Algorithm 1), focuses singly on the parallelization of part 1. The part 2, that performs the Φ computation, uses an unparallelized approach with 2D FFTs.

Let N_r, N_θ, N_φ, $N_{v_\parallel}$, N_μ be respectively the number of points in each dimension r, θ, φ, $v_\parallel$, μ. Le P be the number of processors. The 5D data f has size $(N_r\, N_\theta\, N_\varphi\, N_{v_\parallel}\, N_\mu)$, whereas the 3D data $\tilde{\rho}$ has size $(N_r\, N_\theta\, N_\varphi)$. Because of 2D FFTs, the computation complexity of part 2 is in $\Theta(N_\theta\, N_\varphi\, N_r\, ln(N_\theta)\, ln(N_\varphi))$. The data structures $\tilde{\rho}$ and Φ have size $\Theta(N_\theta\, N_\varphi N_r)$. Here, we have chosen to perform the part 2 redondantly on each processor. The parallelization of part 2 would imply adding commmunication to redistribute data after line 21 and before line 24. The parallelization with these extra communication can not be scalable on many processors because of communication over complexity costs ratio. The Algorithm (1) will be taken as a reference to evaluate the other algorithms presented afterwards.

The communications in part 1 consist in a global transposition of data $\bar{f}$ (line 7) and a broadcast of distributed data $\tilde{\rho}$ (line 16). The respective costs in term of global volume exchanged are $N_\theta\, N_\varphi\, N_r\, N_\mu$ double precision floating point values for the first one and $N_\theta\, N_\varphi\, N_r\, (P-1)$ for the second one.
The method that evaluates the gyroaverage operator introduced in section 2.1 requires that we know, for a given couple (φ, μ), all values $\tilde{\rho}_1(r{=}*, \theta{=}*, \varphi, \mu)$. Furthermore, we have to integrate over $d\mu$ (line 13) to get $\tilde{\rho}$; to avoid extra communication, it is safe to put all $\tilde{\rho}_1(r{=}*, \theta{=}*, \varphi, \mu{=}*)$ on the same processor in order to perform the integrals locally. So, with our gyroaveraging method (that couples r and θ dimensions), the parallel algorithm we have introduced induces small communication cost and good load balance. It mainly uses a parallel loop in φ at line 9.

We now assume that $\tilde{\rho}_1$, which is the output of part 1 computations, is distributed along with φ over the parallel machine because of previous arguments. As we expect that each processor finally knows the data Φ, then a global communication will transfer parts of locally computed Φ structure between processors.

Algorithm 1: Partial parallelization of QN solver (`2d_fft`)

Input : local block $\bar{f}(r = block, \theta = block, \varphi = *, v_{//} = *, \mu)$

(part 1 *)*

Computation : $\tilde{\rho}_1$ by integration in $dv_{//}$ of $\bar{f}$
(parallelization in μ, r, θ)

Send local data $\tilde{\rho}_1(r = block, \theta = block, \varphi = *, \mu)$

Redistribute $\tilde{\rho}_1$ **/ Synchronization**

Receive block $\tilde{\rho}_1(r = *, \theta = *, \varphi = block, \mu = *)$

for *local* φ *values* **do in parallel**

  (parallelization in φ)

  Computation : $\tilde{\rho}_2$ for a given φ by application of operator J_0

  Fourier transform in θ, Solving of LU systems in r

  Computation : $\tilde{\rho}$ for a given φ by integration in $d\mu$ of $\tilde{\rho}_2$

end

Send local data $\tilde{\rho}(r = *, \theta = *, \varphi = block)$

Broadcast of $\tilde{\rho}$ **/ Synchronization**

Receive global data $\tilde{\rho}(r = *, \theta = *, \varphi = *)$

(part 2, not parallelized *)*

Computation : 2D FFTs of $\tilde{\rho}$ on dimensions (θ,φ)
Computation of $\hat{\rho}$

Computation : Solving of LU systems on dim. r ,
Computation of $\hat{\Phi}$

Computation : 2D inverse FFTs 2D on dim. (θ,φ)

Outputs : $\Phi(r = *, \theta = *, \varphi = *)$

With these hypothesis, the communication cost of $\mathrm{N}_\theta \, \mathrm{N}_\varphi \, \mathrm{N}_r \, (\mathrm{P} - 1)$ is best reduced. So, from our actual knowledge, the proposed algorithm for part 1, does the minimum possible volume of communication together with a reasonable computation distribution. The parallel overhead is very low for this solution and it is difficult to further reduce it.

3.2 Partial parallel algorithm with 1D FFT

The part 2 of the algorithm uses the approach with 1D FFTs.

Algorithm 2: Partial parallelization of QN solver (`1d_fft`)

Input : local block $\bar{f}(r = block, \theta = block, \varphi = *, v_{//} = *, \mu)$

(part 1 *)*

Computation : $\tilde{\rho}_1$ by integration in $dv_{//}$ of $\bar{f}$
(parallelization in μ, r, θ)

Send local data $\tilde{\rho}_1(r = block, \theta = block, \varphi = *, \mu)$

Redistribute $\tilde{\rho}_1$ **/ Synchronization**

Receive block $\tilde{\rho}_1(r = *, \theta = *, \varphi = block, \mu = *)$

for *local* φ *values* **do in parallel**

  (parallelization in φ)

  Computation : $\tilde{\rho}_2$ for a given φ by application of operator J_0
  Fourier transform in θ, Solving of LU systems in r

  Computation : $\tilde{\rho}$ for a given φ by integration in $d\mu$ of $\tilde{\rho}_2$

end

Send local data $\tilde{\rho}(r = *, \theta = *, \varphi = block)$

Broadcast of $\tilde{\rho}$ **/ Synchronization**

Receive global data $\tilde{\rho}(r = *, \theta = *, \varphi = *)$

(part 2, not parallelized *)*

for $\varphi \leftarrow 1$ **to** N_φ **do**

  Computation : 1D FFTs of $\tilde{\rho}$ on dimension (θ)

  Computation : Solving of LU systems for $\hat{\Phi}$ modes $(u > 0)$, eq. (20)

  Computation : inverse 1D FFTs on dim. (θ)

  Computation : accumulation of $\tilde{\rho}$ values to compute $\langle\tilde{\rho}\rangle_\theta\,(r = *, \varphi)$

end

Computation : Solving of LU system to find $\langle\Phi\rangle_{\theta,\varphi}$ from $\langle\tilde{\rho}\rangle_\theta$, eq. (18)

for $\varphi \leftarrow 1$ **to** N_φ **do**

  Computation : Solving of LU system for $\langle\Upsilon\rangle_\theta(r=*,\varphi)$, eq. (22)

  Computation : Adding $\langle\Phi\rangle_{\theta,\varphi}$ to $\langle\Upsilon\rangle_\theta(r=*,\varphi)$ gives $\hat{\Phi}^0(r = *, \varphi)$

  Computation : Compute $\Phi(r = *, \theta = *, \varphi)$ in adding $\hat{\Phi}^0(r = *, \varphi)$

end

Outputs : $\Phi(r = *, \theta = *, \varphi = *)$

The computation complexity of part 2 is in $\Theta(\mathrm{N}_\theta\,\mathrm{N}_\varphi\,\mathrm{N}_r\,ln(\mathrm{N}_\theta))$. The FFT transforms dominate the time cost of this part. Concerning the asymptotic complexity, the computation cost is divided by $ln(\mathrm{N}_\varphi)$ compared to the algorithm 1 (`2d_fft`). Thus, one may expect to improve performances for large values of N_φ. We will be concern with this fact when analysing the benchmarks. In the next subsection, we will try to do concurrently some computations of part 2.

3.3 Full parallel algorithm with 1D FFT

The last algorithm version `f1d_fft`, presented here, describes a full parallel algorithm (excluding a very small redondant computation line 21). This algorithm improves the previous one in parallelizing the final computations at the expense of a few more communications.

We will use the intermediate function $\Upsilon(r,\theta,\varphi) = \Phi(r,\theta,\varphi) - \langle\Phi\rangle_\theta(r,\varphi)$. The main idea is to compute $\langle\Phi\rangle_{\theta,\varphi}$ as soon as possible. Thus, it allows us to perform the computations $\langle\Upsilon\rangle_{\theta(r=*,\varphi)}$ in parallel, and we use these quantities to obtain $\hat{\Phi}^0(r=*,\varphi)$ and then $\Phi(r=*,\theta=*,\varphi)$.

Algorithm 3: Full Parallelization of QN solver (`f1d_fft`)

1 *Input :* local block $\bar{f}(r=block,\theta=block,\varphi=*,v_{//}=*,\mu)$
2
3 *(* part 1*)*
4 *Computation :* $\tilde{\rho}_1$ by integration in $dv_{//}$ of $\bar{f}$
5 (parallelization in μ,r,θ)
6 **Send** local data $\tilde{\rho}_1(r=block,\theta=block,\varphi=*,\mu)$
7 **Redistribute** $\tilde{\rho}_1$ / **Synchronization**
8 **Receive** block $\tilde{\rho}_1(r=*,\theta=*,\varphi=block,\mu=*)$
9 **for** *local* φ *values* **do in parallel**
10 (parallelization in φ)
11 *Computation :* $\tilde{\rho}_2$ for a given φ by application of operator J_0
12 Fourier transform in θ, Solving of LU systems in r
13 *Computation :* $\tilde{\rho}$ for a given φ by integration in $d\mu$ of $\tilde{\rho}_2$
14 *Computation :* accumulation of $\tilde{\rho}$ values to get $\langle\tilde{\rho}\rangle_\theta(r=*,\varphi)$
15 **end**
16
17 *(* part 2*)*
18 **Send** local data $\langle\tilde{\rho}\rangle_{\theta(r=*,\varphi=block)}$
19 **Broadcast of** $\langle\tilde{\rho}\rangle_\theta$ / **Synchronization**
20 **Receive** $\langle\tilde{\rho}\rangle_{\theta(r=*,\varphi=*)}$
21 *Computation :* Solving of LU system to find $\langle\Phi\rangle_{\theta,\varphi}$ from $\langle\tilde{\rho}\rangle_\theta$, eq. (18)
22 **for** *local* φ *values* **do in parallel**
23 (parallelization in φ)
24 *Computation :* 1D FFTs of $\tilde{\rho}$ on dimension (θ)
25 *Computation :* Solving of LU systems for $\hat{\Phi}$ modes $(u>0)$, eq. (20)
26 *Computation :* Solving of LU system for $\langle\Upsilon\rangle_{\theta(r=*,\varphi)}$, eq. (22)
27 *Computation :* Adding $\langle\Phi\rangle_{\theta,\varphi}$ to $\langle\Upsilon\rangle_{\theta(r=*,\varphi)}$ gives $\hat{\Phi}^0{}_{(r=*,\varphi)}$
28 *Computation :* inverse 1D FFTs on $\hat{\Phi}^0$ and $\hat{\Phi}^{u>0}$ to get $\Phi_{(r=*,\theta=*,\varphi)}$
29 **end**
30 **Send** local data $\Phi(r=*,\theta=*,\varphi=block)$
31 **Broadcast of values** / **Synchronization**
32 **Receive** global data $\Phi(r=*,\theta=*,\varphi=*)$
33 *Outputs :* $\Phi(r=*,\theta=*,\varphi=*)$

The amount of communication added in comparison to the previous algorithms is $O(\mathrm{N}_r\,\mathrm{N}_\varphi\,\mathrm{P})$. This is negligible when one considers other communication costs. Nevertheless, a synchronization of processors is also induced.

4 Performance Analysis

Numerical experiments were performed on a cluster of 932 nodes owned by CCRT/CEA, France. Each node hosts eight Itanium2 1.6Ghz cores and offers 24GB of shared memory. Performances of the different versions of the QN solver for one 5D test case are presented in table 1 ($\mathrm{N}_r = 256, \mathrm{N}_\theta = 64, \mathrm{N}_\varphi = 256, \mathrm{N}_{v_\parallel} = 16, \mathrm{N}_\mu = 32$). Note that the Vlasov solver of the GYSELA code uses a parallelization based on a domain decomposition in dimension μ and r. So, the number of processors P is given by the product of p_μ the number of μ values with p_r the number of block in dimension r. The number of μ values in the presented test case is $\mathrm{p}_\mu = 32$, then our parallel program uses a minimum of 32 processors. Then, the relative speedups shown in table 1 considers as a reference the execution times on 32 cores of four computation nodes.

In order to give fine performance results, we will subdivise the algorithms into small recognizable parts. The computation of $\tilde{\rho}$ (part 1) consists in a communication part and parallelized integral calculations. For the different versions, the solver giving Φ depending on $\tilde{\rho}$ (part 2) could be decomposed into communication steps plus two types of computation: the redondant ones and the parallel ones. Finally, one can have a look to time costs of the QN solver considering three mesures: 1) the time spent in communication (the maximum among all processors is given in table 1), 2) the time spent in sequential computations (each processors has exactly the same work to do), 3) the time spent in parallel tasks (the maximum among all processors is taken).

The execution time measurements of algorithms `2d_fft`, `1d_fft` and `f1d_fft` are given in table 1. For the reference algorithm `2d_fft`, there is only one computation part that is parallelized: the $\tilde{\rho}$ computation (part 1). The timing results shows this part is scalable. A major problem is that the `solve_seq` becomes the dominant calculation as soon as P is above a threshold. The Amdhal's law limits the overall performance of this algorithm with usual parameter sets. As a consequence, the relative speedup on 256 processors (1.5 compared with 32 processors as a reference) is extremely low. The communication part is partly responsible for that; it has an overall cost in $\Theta(\mathrm{N}_\theta\,\mathrm{N}_\varphi\,\mathrm{N}_r\,\mathrm{N}_\mu + \mathrm{N}_\theta\,\mathrm{N}_\varphi\,\mathrm{N}_r\,(\mathrm{P}-1))$ that increases with P.

In the `1d_fft` version, the cost associated with `solve_seq` (corresponding to part 2 work) is around 5 times smaller than in the `2d_fft` version. This observed ratio is near to the asymptotic cost ratio which is $log(\mathrm{N}_\varphi) = 8$. The performance of the QN solver `1d_fft` is far from perfect, but the relative speedup of computation (`solve_par+_seq`) is nevertheless improved.

The `f1d_fft` algorithm improves the previous `1d_fft` algorithm with a better work distribution. For the `f1d_fft` algorithm, almost all computations are parallelized. And the speedup of the computation part (`solve_par+_seq`) is impressive: 7.7 instead of 8 in the ideal case. The remaining parallel overhead come from the small sequential computation of `solve_seq` and above all communication `comm`. This QN solver is efficient and reduce on 256 processors the time of 5.7 s with `2d_fft` to 1.2 s with `f1d_fft`.

One could think about using other methods such as multigrid or a direct solver for the part 2 of the work. But it won't diminish the cost of communications required for the calculation of $\tilde{\rho}$, and for the final broadcast. So, we could not expect a large enhancement of parallel performance in such a way.

Nb. procs. (P)	32	128	256
p_r	1	4	8
Algorithm `2d_fft`			
comm	0.350 s	0.687 s	0.767 s
solve_seq	4.208 s	4.613 s	4.535 s
solve_par	3.908 s	1.076 s	0.555 s
Rel. speedup solve_par+_seq	1.0	1.4	1.6
Total `2d_fft`	8.444	6.299	5.771
Rel. speedup `2d_fft`	1.0	1.3	1.5
Algorithm `1d_fft`			
comm	0.395 s	0.598 s	0.726 s
solve_seq	0.960 s	0.972 s	0.998 s
solve_par	3.925 s	0.996 s	0.504 s
Rel. speedup solve_par+_seq	1.0	2.5	3.3
Total `1d_fft`	5.211 s	2.529 s	2.174 s
Rel. speedup `1d_fft`	1.0	2.1	2.4
Algorithm `f1d_fft`			
comm	0.377 s	0.593 s	0.668 s
solve_seq	0.003 s	0.006 s	0.018 s
solve_par	4.078 s	1.039 s	0.528 s
Rel. speedup solve_par+_seq	1.0	3.9	7.7
Total `f1d_fft`	4.375 s	1.603 s	1.178 s
Rel. speedup `f1d_fft`	1.0	2.7	3.7

Table 1: Comparison of the three algorithms introduced. Time measurements for one call to the QN solver in seconds and relative speedup are given (compared to performance of 32 processors with $p_r = 1$ and $p_\mu = 32$).

5 Conclusion

We describe the parallelization of a quasineutral Poisson solver used into a full-f gyrokinetic 5D simulator[2]. The parallel performance of different numerical solving methods is demonstrated. The last method achieves good scalability up to 256 processors. Nevertheless, the communication induced by the coupling of the quasineutral solver and the Vlasov code remains high and deteriorate the efficiency of the method. More work has to be done in order to reduce these communication costs to further improve this solver. Avoiding the broadcast of Φ 3D data structure would be a solution. OpenMP paradigm will help to further reduce the computation costs whenever they are large enough, *e.g.* for large $N_{v_\parallel}$ values.

[2]Acknowledgments: This work was partially suported by EURATOM/CEA, contract V.3529.001.

References

[1] A.J. Brizard, T.S. Hahm, *Foundations of nonlineargyrokinetic theory*, PPPL report 4153, 2006.

[2] A.M. Dimits et al., *Comparisons and physics basis of tokamak transport models and turbulence simulations*, Phys. Plasmas **7**, pp. 969-983, (2000).

[3] G.W. Hammet, F.W. Perkins, *Fluid models for Landau damping with application to the ion-temperature-gradient instability*, Phys. Rev. Lett. **64**, pp. 3019-3022, (1990).

[4] S. Jolliet, A. Bottino, P. Angelino, R. Hatzky, T.M. Tran, B.F. Mcmillan, O. Sauter, K. Appert, Y. Idomura, L. Villard, *A global collisionless PIC code in magnetic coordinates*, Comp. Phys. Comm., **177**, pp. 409-425, (2007).

[5] V. Grandgirard, M. Brunetti, P. Bertrand, N. Besse, X. Garbet, Ph. Gendrih, G. Manfredi, Y. Sarazin, O. Sauter, E. Sonnendrucker, J. Vaclavik, L. Villard, *A drift-kinetic Semi-Lagrangian 4D code for ion turbulence simulation*, J. Comput. Phys., **217**(2), pp. 395-423, (2006).

[6] V. Grandgirard, Y. Sarazin, X. Garbet, G. Dif-Pradalier, Ph. Ghendrih, N. Crouseilles, G. Latu, E. Sonnendrücker, N. Besse, P. Bertrand, *Computing ITG turbulence with a full-f semi-Lagrangian code*, Vlasovia 2006, Communications in Nonlinear Science and Numerical Simulation, **13**(1), pp. 81-87, (2008).

[7] T.S. Hahm, *Nonlinear gyrokinetic equations for tokamak microturbulence*, Phys. Fluids, **31**, p. 2670, 1988.

[8] R. Hatzky, T.M. Tran, A. Koenis, R. Kleiber, S.J. Allfrey *Energy conservation in a nonlinear gyrokinetic particle-in-cell code for ion-temperature-gradient-driven modes in θ-pinch geometry* Physics of Plasma, Vol. 9, 2002.

[9] G. Latu, N. Crouseilles, V. Grandgirard, E. Sonnendrücker, *Gyrokinetic semi-Lagarngian parallel simulation using a hybrid OpenMP/MPI programming*, Recent Advances in PVM an MPI, Springer, LNCS, pp. 356-364, Vol. 4757, (2007).

[10] W.W. Lee, *Gyrokinetic approach in particle simulation*, Phys. Fluids **26**, p. 556, (1983).

[11] Z. Lin, W.W. Lee, *Method for solving the gyrokinetic Poisson equation in general geometry*, Phys. Rev. E **52**, p. 5646-5652, (1995).

[12] R.G. Littlejohn, J. Math. Phys. **23**, p. 742, (1982).

[13] Y. Nishimura, Z. Lin, J.L.V. Lewandowski, *A finite element Poisson solver for gyrokinetic particle simulations in a global field aligned mesh*, J. Comput. Phys, **214**, pp. 657-671, (2006).

[14] H. Qin, *A short introduction to general gyrokinetic theory*, PPPL report 4052, 2005.

Centre de recherche INRIA Nancy – Grand Est
LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)

Centre de recherche INRIA Bordeaux – Sud Ouest : Domaine Universitaire - 351, cours de la Libération - 33405 Talence Cedex
Centre de recherche INRIA Grenoble – Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier
Centre de recherche INRIA Lille – Nord Europe : Parc Scientifique de la Haute Borne - 40, avenue Halley - 59650 Villeneuve d'Ascq
Centre de recherche INRIA Paris – Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex
Centre de recherche INRIA Rennes – Bretagne Atlantique : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex
Centre de recherche INRIA Saclay – Île-de-France : Parc Orsay Université - ZAC des Vignes : 4, rue Jacques Monod - 91893 Orsay Cedex
Centre de recherche INRIA Sophia Antipolis – Méditerranée : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399