Interactive out-of-core visualisation of 4-D+t plasma data

Christophe Mion, Florence Zara, Jean-Michel Dischler

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Interactive out-of-core visualisation of 4-D+t plasma data

Christophe Mion — Florence Zara — Jean-Michel Dischler

N° 5444

Décembre 2004

Thème NUM D

ISSN 0249-6399 ISRN INRIA/RR--5444--FR+ENG

Interactive out-of-core visualisation of 4-D+t plasma data

Christophe Mion* , Florence Zara * , Jean-Michel Dischler *

Thème NUM D — Modélisation, simulation et analyse numérique
Projet CALVI

Rapport de recherche n° 5444 — Décembre 2004 — 28 pages

Abstract: This research report presents a new interactive visualisation technique for exploring plasma behaviors resulting from 4-D+t numerical simulations on regular grids. Therefore, a new out-of-core 4-D+t scalar field visualisation technique, based on a "focus and context" approach is presented. The latter uses a hybrid data compression method. The originality of this work precisely consists in coupling the 3-D visualization with the progressive load and decompression of data in such a way that it still guarantees real-time framerates even on low-end PCs while maintaining a high degree of numerical precision.

Key-words: Out-of-core Visualisation, 4-D+t Visualisation, Plasma Simulation, Data Compression.

* LSIIT-IGG, UMR CNRS-ULP 7005

Unité de recherche INRIA Lorraine
LORIA, Technopôle de Nancy-Brabois, Campus scientifique,

Exploration interactive de données 4-D+t issues de simulations de plasma

Résumé : Ce travail présente une nouvelle approche interactive d'exploration du comportement des plasmas issus de simulations numériques 4-D+t. Pour ce faire, nous proposons une méthode de visualisation de champs scalaires 4-D+t de type "focus and context" couplée à l'utilisation d'une approche hybride de compression des données. Comme la masse de données demeure après compression telle qu'il n'est toujours pas possible de la charger complètement en mémoire, une partie des données doit rester sur disque. Toute l'originalité de ce travail réside alors dans sa capacité à maintenir une performance de visualisation 3-D temps réel sur PC, malgrès un chargement et une décompression réalisés en permanence à la volée, tout un préservant un degré de précision très élevé.

Mots-clés : Visualisation out-of-core, visualisation 4-D+t, simulation de plasma, compression de données.

1 Introduction

1.1 Contexte général

Dans la quête actuelle de nouvelles ressources énergétiques, la fusion thermonucléaire constitue un des enjeux majeurs de ce siècle. Mais la réalisation de réacteurs nécessite la détermination de champs magnétiques permettant de confiner suffisamment longtemps un plasma très chaud. La complexité du phénomène rend indispensable le recours à des simulations numériques avant de procéder à des tests en vraie grandeur. Le modèle généralement utilisé pour étudier le comportement des particules du plasma se base sur l'**équation de Vlasov** couplée avec les **équations de Maxwell** ou **Poisson** permettant de décrire les champs électriques et magnétiques [FS03b, FS03a, SFF+03, GPS03] :

$$\frac{\partial \vec{f_\alpha}}{\partial t} + \vec{v}.\frac{\partial f_\alpha}{\partial \vec{x}} + \frac{q_\alpha}{m_\alpha}\left(\vec{E} + \frac{\vec{v} \wedge \vec{B}}{c}\right).\frac{\partial f_\alpha}{\partial \vec{v}} = 0. \quad (1)$$

L'équation de Vlasov caractérise l'évolution dans le temps et l'espace de la distribution des particules d'un plasma non collisionnel. Le but de la simulation est de calculer la fonction de distribution $f_{alpha}(\vec{x}, \vec{v}, t)$ caractérisant, pour chaque espèce de particules, leur densité à un instant t donné, pour un vecteur position $\vec{x}$ et un vecteur vitesse $\vec{v}$ dans l'**espace des phases**.

Il existe deux principales approches pour la résolution numérique de l'équation de Vlasov. La première approche se base sur un calcul par particules (dit Particle-In-Cell ou PIC), tandis que la seconde se base sur une discrétisation de l'espace des phases. Cette dernière permet de pallier la faible précision des codes de type PIC. Une méthode de discrétisation de l'équation de Vlasov a été proposée dans [FSB01]. Elle utilise des volumes finis et consiste à approximer la fonction de distribution $f(\vec{v}, \vec{x}, t)$ par un ensemble de volumes discrets de l'espace des phases. Notons qu'une simulation dans un espace physique de dimension n (avec $n = 1, 2, 3$) entraîne des volume discrets dans l'espace des phases de dimension $2n$.

C'est à partir des données fournies par le code `Vador2D` [VF02] de simulations de plasmas 2-D ($n = 2$) que se base le présent travail. La résolution des équations de Vlasov par ce code fournit en sortie un champ scalaire à 4 dimensions, qui sont les dimensions de l'espace Euclidien $\vec{X} = (x, y)$ et de l'espace des vitesses $\vec{V} = (v_x, v_y)$. Par la suite, nous appellerons **diagnostique**, $D_{u,v}(i, j)$ (avec $(u, v, i, j) \in [1, N]$, le résultat de la simulation numérique pour un instant t donné sur une grille discrète de résolution N, où (u, v) représente la position et (i, j) la vitesse.

1.2 Motivations

Une simulation sur une grille de résolution $N = 64$ entraîne une masse de données égale à $64 \times 64 \times 64 \times 64$ valeurs double pour chaque pas de temps t, soit $12,5$ Go pour seulement 100 pas de temps ! (la précision double est nécessaire à la réduction du bruit numérique et est imposée par le simulateur). Cette très grande quantité d'information induit deux

problèmes qui sont l'exploitation rationnelle et assistée par ordinateur du flot de données, ainsi que la gestion des unités de stockage afin de pouvoir exploiter et parcourir ce volume de données de manière interactive. En effet, il semble inutile de calculer plusieurs Go de données par simulations, si nous ne pouvons ultérieurement les analyser, et de la même manière il est inutile de posséder une méthode d'exploration si cette dernière n'est pas interactive et précise. En effet, l'**interactivité** est un élément primordial dans l'exploration de champs de données car elle facilite la reconstruction mentale d'un phénomène et donc son analyse. La précision quant à elle, est nécessaire à la validation de l'analyse par les utilisateurs du système.

Pour répondre à l'ensemble des contraintes engendrées par une simulation en 4-D+t, il est alors nécessaire de combiner deux domaines informatiques : la **visualisation** qui fournit la méthode d'exploration des données, et la **compression de données** qui permet la diminution des accès vers les interfaces de stockages. Ces deux thématiques sont donc abordées dans ce travail de recherche, le but étant de proposer un couplage des deux qui réponde bien aux conditions précédemment évoquées (interaction temps réel et très haute précision). Nous cherchons donc à réaliser dans ce rapport de recherche un outil interactif d'exploration et de visualisation de données scalaires 4-D(+t).

1.3 Principes de la méthode

A ce jour, les physiciens visualisent essentiellement les données 4-D en réalisant une intégration selon deux des quatre variables. L'affichage se réduit donc à une image 2-D (ou une vidéo en considérant le temps). Le problème est que toutes les méthodes de visualisation basées sur des intégrations, projections ou coupes pour réduire la dimension à deux ou trois ne permettent évidemment pas d'analyser les subtilités d'un phénomène complexe dans son intégralité.

Des techniques de visualisation spécifiques aux plasmas ont été proposées par le passé [MSW+02, WMQR02]. Mais ces techniques se basent sur des simulations par approche PIC. Dans un cadre PIC, les efforts se concentrent surtout sur l'affichage des particules en combinant une visualisation par point avec une visualisation volumique. Ces méthodes sont donc intrinsèquement inadaptées à notre problématique, la nature de l'information n'étant pas la même.

A notre connaissance, il s'agit dans ce rapport d'un premier travail consacré à la visualisation de données plasma issues d'une simulation 4-D+t par discrétisation de l'espace des phases, laquelle permet de visualiser localement l'ensemble du résultat d'une simulation contrairement aux projections utilisées jusqu'alors.

La solution que nous proposons se base sur une combinaison entre le "focus + context" (en utilisant une loupe virtuelle) et la technique "world within worlds" (voir section état de l'art pour une description plus détaillée de ces méthodes). Le principe général en est le suivant : en dehors de la loupe, la donnée est affichée par simple intégration 2-D comme le font les physiciens actuellement. Mais au sein de la loupe, chaque case de projection 2-D contient l'ensemble du sous espace 2-D lui correspondant. Nous affichons donc un monde

2-D dans un monde 2-D, ce qui permet d'afficher l'ensemble de la fonction 4-D dans la zone délimitée par la loupe. L'affichage se fait sous la forme d'une carte de hauteur, donc en 3-D. Selon la taille de la loupe, le nombre de triangles générés peut devenir excessif. Aussi nous proposons, pour limiter les primitives graphiques à afficher, une approche multi-résolution de visualisation de cartes de hauteurs directement liée à la décompression. Ceci nous permet de ne visualiser qu'une partie des données en haute-résolution tandis que les autres sont affichées à un niveau de détail inférieur, sans toutefois nuire à la visibilité du phénomène. Nous montrons également qu'une représentation judicieuse de l'information permet de maintenir une performance temps réel malgrès un chargement permanent à partir d'un disque. C'est d'ailleurs là, la principale contribution de cette approche.

1.4 Plan du rapport

La suite de ce rapport se découpe en cinq parties. Dans un premier temps, un bref état de l'art des deux domaines informatiques nécessaires à la réalisation d'un outil interactif d'exploration de champs scalaires 4-D+t sera proposé, à savoir les techniques de visualisation et de compression de données. La seconde partie est ensuite consacrée à la description de notre nouvelle méthode de visualisation, puis la troisième à la description de notre méthode de compression directement intégrée et adaptée à la visualisation. Nous récapitulons ensuite l'ensemble de la méthode, c'est-à-dire l'algorithme qui a été mis en oeuvre. Enfin, avant de conclure, la dernière partie du rapport se propose de présenter et d'analyser les résultats obtenus.

2 Etat de l'art

2.1 Visualisation multidimensionnelle

En visualisation scientifique, le problème de données 4-D+t semble à priori bien connu depuis longtemps. Il existe en effet de nombreuses méthodes, y compris des outils commerciaux, permettant une visualisation de données dites 5-D. Le champs visualisé consiste généralement en un quintuplet de la forme (x, y, z, u, t), soient 3 coordonnées d'espace, une dimension physique (paramètre comme la température, la pression, etc... dans le cas de la météorologie) et la dimension temporelle. Ce type de données se visualise facilement en 3-D, en utilisant la position pour (x, y, z) et la couleur (et/ou la taille) pour encoder le paramètre physique u. Cela revient ainsi à visualiser une fonction allant de l'espace $\mathbb{R}^3 + t$ dans l'espace $\mathbb{R} + t$.

Cependant, dans le cas des plasmas 4-D+t, la donnée à représenter est un "vrai" champ scalaire correspondant à une fonction à cinq dimensions de la forme $d = f(x, y, vx, vy, t)$. Cette fonction représente la densité de particules à un instant donné en un point défini par 2 coordonnées d'espace et 2 coordonnées de vitesse. Il s'agit alors de visualiser une fonction allant de l'espace $\mathbb{R}^4 + t$ dans l'espace $\mathbb{R} + t$. De manière informatique, il s'agit d'un tableau 4-D+1-D de réels.

La visualisation de données en "vrai" 4-D+t n'est, contrairement au quintuplet précédent, pas intuitive. En effet, il est difficile de se représenter une donnée de dimension supérieure à 3, la couleur ou la taille ne suffisant pas à encoder une véritable quatrième dimension. Or ce type de données peut être relativement courant, c'est pourquoi de nombreuses techniques de visualisation dites multidimensionnelles ont déjà été développées par le passé. Ce domaine ne fait généralement pas partie de la visualisation scientifique, mais fait partie de ce que la communauté appelle la "visualisation d'information".

Les techniques de visualisation multidimensionnelle sont essentiellement développées pour un nombre de dimensions très grand (bien supérieur à 5) :

- **Les techniques à panneaux** comme "scatterplot matrix" [CCKT83, Cle93, BMMS91], "Brushing" [BC87], "hyperslice" [vv93] et "hyperbox" [AC91] par exemple, transforment un espace nD en un ensemble d'espaces 2-D appelés panneaux. Elles permettent principalement de mettre en évidence des corrélations entre paires de variables. Ces techniques semblent donc peu adaptées à l'étude des plasmas 4-D.

- **Les techniques iconographiques** ont pour but d'afficher une icône ou "glyph" [PG88, GPW89, SGB91, Lev91] dont les paramètres de représentations correspondent aux valeurs des variables de la fonction à analyser. Les valeurs de la fonction sont alors associées à des attributs graphiques comme la forme, la taille, la couleur ou la position. Mais la difficulté avec ce type d'approche consiste à trouver une icône capable de caractériser la fonction à étudier.

- **Les techniques à vue hiérarchique** ont pour but de hiérarchiser la visualisation de données multidimensionnelles en fonction de différentes variables. Différentes méthodes ont été élaborées sur ce principe : "axes hiérarchiques" [MGTS90, LWW90, MTS91a, MTS91b], "dimension stacking" [LWW90, War94] et "world within worlds" [FB90b, FB90a]. Mais ces techniques de visualisation se limitent généralement aux données dont la discrétisation est assez grossière.

- **Les techniques avec axes non cartésiens** redéfinissent des axes non cartésiens afin d'obtenir un espace nD affichable en 2-D. Elles sont représentées par les techniques à "coordonnées parallèles" [IRC87, ID87, ID90] permettant une analyse de la géométrie des objets multidimensionnels, et les techniques à "coordonnées en étoile" [Kan00] consistant à réarranger les axes représentant les dimensions de chaque variable sur un cercle dans un espace 2-D.

Plus récemment la visualisation d'information s'est vue se rapprocher de la visualisation scientifique 3-D. **Les méthodes "focus + context"** [RC94, JN02, DGH03] par exemple se basent sur l'utilisation d'un point focal permettant à l'utilisateur de naviguer dans l'espace de données et de réaliser un zoom sur une portion désignée des données. Une approche très récente [RE04] utilise la visualisation volumique pour afficher des informations multidimensionnelles en ré-exprimant une partie des données en 3-D. La méthode "table lens" [RC94, RT97] par ailleurs consiste à visualiser l'information dans une grille de taille

variable, alors que la méthode "radial focus + context" [MRC91, Fur86, JN02] organise les n dimensions dans un mode radial.

Notre approche se propose de combiner le principe du "focus and context" avec celui du "worlds within worlds". La principale difficulté consiste alors à gérer la masse de données de façon à garantir une performance d'affichage temps réel tout en préservant une précision numérique élevée. Il s'agit actuellement d'une thématique importante et récente en visualisation scientifique, la masse de données ne cessant de croître rapidement avec les capacités de calculs et de simulation.

2.2 Compression de données multidimensionnelles

La compression de données est un domaine de recherche très vaste si bien qu'une étude exhaustive dépasserait très largement le cadre de ce rapport. Nous proposons ici de survoler brièvement les techniques qui sont le plus en relation avec notre travail. Globalement, les algorithmes de compression de données peuvent se diviser en deux principales catégories qui sont la **compression sans perte** et la **compression avec perte**. La compression sans perte permet la reconstruction à l'identique des données compressées, tandis que la compression avec perte implique une détérioration des données. Dans ce cas, le but est de minimiser ou de contrôler l'erreur introduite. Nous ne présentons ici que les techniques de compression avec perte, celles sans perte n'étant pas adaptées à des champs scalaires de grande taille. Les trois principales techniques sont les suivantes :

- **Les méthodes par quantification** [LBG80, CS94b, CS94a] qui transforment des données représentées par de grands nombres par des plus petits se codant sur un espace plus faible. Malheureusement ces méthodes peuvent engendrer des pertes d'informations importantes et le contrôle de l'erreur n'est pas aisé.
- **Les méthodes de compression par ondelettes** [Dau88, DJL92, Mey92, Lew95, RB99] qui consistent à choisir une ondelette ou fonction d'onde, appelée **ondelette mère**. Cette ondelette mère sert de base à l'espace vectoriel des fonctions, qui sont exprimées sous la forme de combinaisons linéaires de l'ondelette mère à une échelle et translation près.

Notre méthode hybride de compression est basée dans un premier temps sur une transformée en ondelette de Haar de tous les sous-espaces 2-D du diagnostique. Un second niveau de compression est ensuite basé sur une quantification par seuil d'erreurs pour tenir compte des corrélations entre les sous-espaces 2-D d'un même diagnostique. Pour des raisons de performance, la cohérence temporelle n'est pas prise en compte et aucune compression n'est effectuée selon cette dimension.

3 Visualisation d'un champ scalaire 4-D

Nous détaillons ici la méthode de visualisation que nous avons développée pour permettre une analyse précise en temps réel des données scalaires 4-D+t issues de la simulation de plasmas. Pour cela nous allons tout d'abord décrire le pipeline graphique, puis nous décrirons la méthode "focus and context" que nous avons proposée.

3.1 Pipeline graphique

La simulation de plasmas fournit pour chaque pas de temps t des données scalaires 4-D calculées à partir de grilles uniformes d'un volume total de taille N^4. La **structure de stockage** retenue pour ces données est un tableau 2-D de dimension $N \times N$ noté D^t pour diagnostique au temps t. Chaque élément ou case de D^t est une image 2-D, elle-même de dimension $N \times N$, notée $D^t_{u,v}$ avec $(u,v) \in [1,N]$. Nous appellerons cette image **sous-espace** du diagnostique D^t. Un élément $D^t_{u,v}[i][j]$ de ce sous-espace, avec $(i,j) \in [1,N]$, correspond à un scalaire représentant la **densité de particules** pour une position (u,v), une vitesse (i,j) et un temps t donné. Pour des raisons de lisibilité nous ignorerons le paramètre temporel dans ce qui suit et nous noterons le diagnostique D au lieu de D^t. Par ailleurs, la densité moyenne $\overline{D_{u,v}}$ d'un sous-espace $D_{u,v}$ est donnée par :

$$\overline{D_{u,v}} = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} D_{u,v}[i][j]$$

La figure 1 illustre cette structure de données.

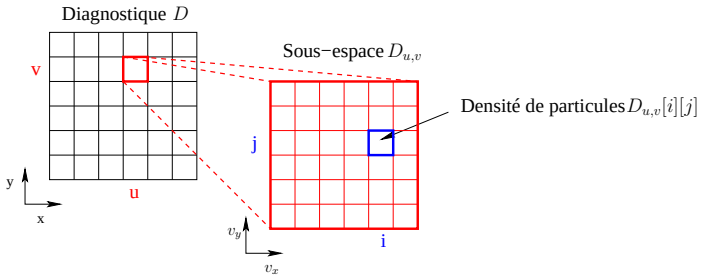

Figure 1: Structure D des données de l'application

Pour l'affichage du champ scalaire D à quatre dimensions, nous utilisons une approche "focus and context" combinée avec une méthode "worlds within worlds". Le "focus" est matérialisé par une **loupe** qui peut être déplacée interactivement par l'utilisateur. Cette loupe notée L_{u_L,v_L,S_L} correspond à un ensemble rectangulaire de sous-espaces de D défini par :

$$L_{u_L,v_L,S_L} = \{D_{u_L+k,v_L+l}\} \text{ avec } (k,l) \in [1,S_L] \text{ et } (u_L,v_L) \in [1,N-S_L+1].$$

S_L définit ainsi la taille de la loupe et (u_L, v_L) la position de son coin supérieur gauche au sein du diagnostique D (partie gauche de la figure 2).

Pour un pas de temps t donné, le diagnostique D est visualisé en 3-D de la manière suivante :

- Tout sous-espace $D_{u,v}$ extérieur à la loupe, c'est-à-dire tel que $D_{u,v} \notin L_{u_L,v_L,S_L}$, est affiché sous la forme d'un "quad" carré dont la couleur $c_{u,v}$ est fonction de la densité moyenne $\overline{D_{u,v}}$. Nous avons alors $c_{u,v} = F\left(\overline{D_{u,v}}\right)$, où F désigne une **fonction de transfert** qui associe à tout scalaire un triplet de couleur (R, V, B).
- Au sein de la loupe, c'est-à-dire pour $D_{u,v} \in L_{u_L,v_L,S_L}$, un affichage complet du sous-espace $D_{u,v}$ est effectué sous la forme d'une carte de hauteur colorée en utilisant la même fonction de transfert F.

La figure 2 illustre l'affichage que nous venons de décrire.

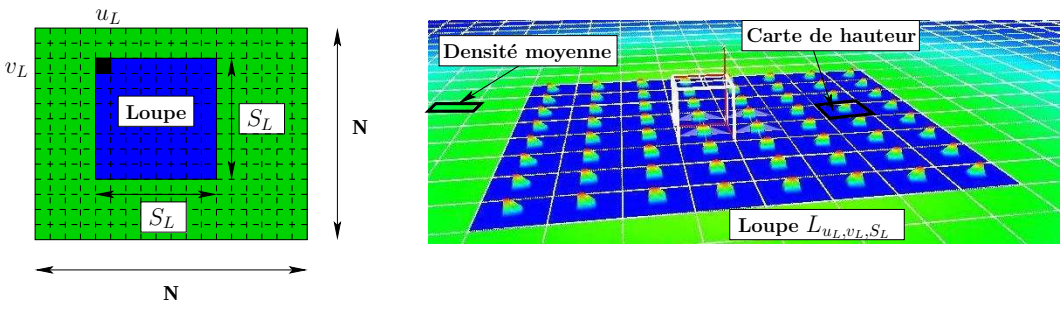

Figure 2: Visualisation 3-D du diagnostique D en utilisant une loupe rectangulaire L_{u_L,v_L,S_L} de taille $S_L \times S_L$ positionnée au sein de D en (u_L, v_L)

L'ensemble du pipeline graphique de ce système de visualisation 4-D est décrit par la figure 3. L'utilisateur sélectionne le pas de temps t, et le diagnostique D correspondant doit être affiché sous la forme décrite par la figure 2. Pour cela, une première partie de l'information compressée est chargée depuis le disque dur. Cette information concerne l'ensemble du diagnostique D. Cette information est décompressée en mémoire vive et le restera tant que l'utilisateur ne change pas de pas de temps. Puis, selon la position (u_L, v_L) de la loupe L, une autre partie de l'information concernant uniquement cette loupe est chargée. Cette partie restera compressée en mémoire. Elle n'est décompressée que pour faire l'affichage de la carte de hauteur correspondante.

A priori le nombre N_T de triangles générés pour afficher la carte de hauteur peut être très important avec $N_T = 2N^2S_L^2$. C'est pourquoi nous proposons d'adapter la résolution des sous-espaces $D_{u,v}$ à leur position au sein de la loupe ainsi qu'à la position de l'observateur. Le mécanisme de décompression que nous utilisons (voir paragraphe suivant) permet précisément de contrôler cette résolution.

Par ailleurs, lorsque l'utilisateur déplace la loupe sans changer de pas de temps, les informations concernant les nouveaux $D_{u,v}$ sélectionnés sont chargées puis décompressées en fonction de la résolution de l'affichage de la carte de hauteur et ainsi de suite.

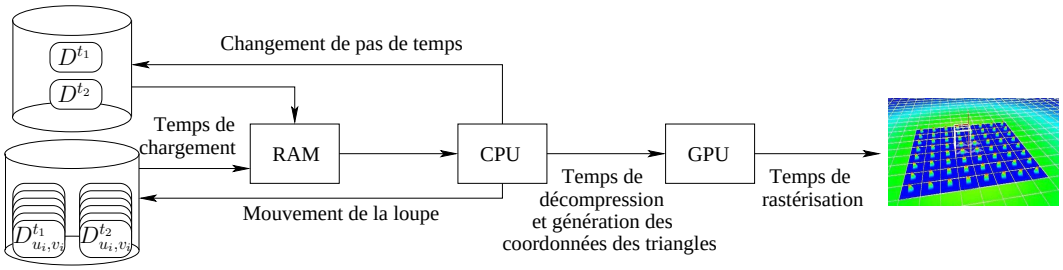

Figure 3: Pipeline graphique de notre système de visualisation 4-D+t "out-of-core"

La figure 3 met ainsi en avant les trois principaux goulets d'étranglement du système : le temps mis pour le chargement dépendant du disque et de la taille des fichiers fonction du taux de compression, le temps de décompression dépendant du processeur et de la technique de compression, et le temps mis par la carte graphique dépendant du nombre de triangles envoyés. Les trois temps cumulés doivent répondre au critère de temps réel. Notons que la performance est la plus dégradée lorsque l'utilisateur modifie le pas de temps. En effet, c'est là que le plus de données sont à charger et à décompresser. Nous voyons dans le paragraphe suivant la technique utilisée pour afficher la carte de hauteur.

3.2 Génération des triangles de la carte de hauteur

La loupe L délimite le domaine du diagnostique D dont l'affichage est réalisé de manière détaillée, c'est-à-dire sous la forme d'une carte de hauteur en 3-D (voir figure 2).

De très nombreuses méthodes de visualisation de cartes de hauteur ont été développées par le passé (il s'agit d'un domaine appelé visualisation de terrains). Parmi les méthodes les plus connues nous pouvons citer par exemple le ROAM [DWS+97] (Real-time Optimally Adapting Meshes) qui consiste à adapter le nombre et la taille des triangles à la position de l'observateur. Cependant toutes ces méthodes font des calculs (essentiellement des mesures d'erreur) qui peuvent être coûteux. Or la décompression que nous sommes contraints d'appliquer monopolise déjà l'essentiel des ressources CPU, si bien qu'il n'est matériellement pas possible d'appliquer ce type d'algorithmes.

De plus, nous proposons de nous servir de la compression pour contrôler la résolution des cartes de hauteur à afficher, l'affichage se limitant alors à construire trivialement un nombre de triangles égal à deux fois le nombre de pixels de la carte (deux triangles par pixel). Cette compression, basée sur les ondelettes de Haar (voir paragraphe suivant) exprime les sous-espaces $D_{u,v}$ sous une forme hiérarchique (pyramidale). Soit $H^m[i][j]$ la carte de hauteur de l'élément $D_{u,v}[i][j]$ pour un niveau m de résolution souhaité. $H^m[i][j]$ est définie en fonction

d'un sous-espace $D_{u,v}$ par :

$$H^0[i][j] = D_{u,v}[i][j], \qquad H^m[i][j] = \frac{1}{4}\sum_{k=0}^{1}\sum_{l=0}^{1} H^{m-1}[i+k][j+l].$$

Pour la visualisation, le niveau de résolution m est fixé en fonction du point de vue de l'observateur et en fonction de la position au sein de la loupe du sous-espace $D_{u,v}$ considéré. Ainsi plus le sous-espace est proche du centre de la loupe, plus la résolution est élevée, mais plus l'observateur est éloigné plus la résolution est faible.

L'utilisation de la loupe à des résolution différentes permet ainsi de mettre en valeur les détails du phénomène de manière localisée en concentrant les ressources CPU et GPU uniquement sur les portions de la simulation jugées importantes par l'utilisateur, tout en conservant une vision globale du phénomène au voisinage de la partie ciblée.

L'essentiel du travail de visualisation se situe donc au niveau de la décompression des sous-espaces du diagnostique. C'est ce que nous détaillons dans le paragraphe suivant.

4 Compression des données 4-D+t

Nous décrivons dans ce paragraphe la technique de compression que nous avons mise en oeuvre. Cette technique garantit un taux de compression important tout en préservant une grande précision. Elle permet un chargement rapide de l'information, et est adaptée à la méthode hiérarchique de visualisation que nous venons de décrire. Cette compression s'applique à chaque diagnostique D pour chaque pas de temps de façon indépendante. En effet, pour des raisons de performance nous ne tenons pas compte de la cohérence temporelle.

Pour chaque diagnostique D, la compression se fait à deux niveaux :

- A un **niveau local**, c'est-à-dire pour chaque sous-espace $D_{u,v}$. Chacun de ces sous-espaces correspond à une image 2-D qui peut être compressée par une technique 2-D classique. Pour notre part, nous utilisons une compression par ondelettes de Haar pour effectuer la compression de chacun de ces sous-espaces.
- A un **niveau global**, pour tenir compte de la cohérence entre les $N \times N$ sous-espaces d'un même diagnostique D.

Dans ce qui suit, nous décrivons ces deux procédures.

4.1 Compression des sous-espaces par ondelettes de Haar

Le principe de cette compression est simple et connu [Mul97], de ce fait nous ne détaillons pas ici l'ensemble de ce processus mais seulement les grandes étapes. Le sous-espace $D_{u,v}$, qui est une image 2-D, est tout d'abord ré-écrit sous la forme de moyennes et de différences

en procédant ligne par ligne puis colonne par colonne. Le résultat de cette transformée en ondelettes de Haar est une nouvelle image 2-D de même taille $N \times N$, dont la valeur du coin supérieur gauche correspond à la valeur moyenne sur toute l'image originale. Nous notons $\mathcal{H}(D_{u,v})$ la transformée en ondelettes de Haar du sous-espace $D_{u,v}$.

Le sous-espace $D_{u,v}$ peut être reconstruit à partir de $\mathcal{H}(D_{u,v})$ en appliquant la transformation inverse. Notons que, comme la représentation de Haar est hiérarchique (ensemble de moyennes), il est possible de ne reconstruire le sous-espace d'origine que jusqu'à un niveau m donné qui peut être plus ou moins élevé. Le niveau le plus élevé de la pyramide correspond à une résolution de 1 et la valeur alors calculée représente la moyenne sur tout le sous espace. Il est ainsi aisé de se rendre compte que le temps de calcul de cette reconstruction dépend directement de la résolution souhaitée : plus la résolution de la reconstruction est faible, c'est-à-dire plus le niveau dans la pyramide est élevé et moins les temps de calcul sont importants.

D'autre part, comme l'ensemble des valeurs issues de la transformée $\mathcal{H}(D_{u,v})$ correspondent à des différences, celles-ci sont proches de zéro. La compression consiste alors à fixer un seuil noté ϵ_0, et de mettre à zéro l'ensemble des valeurs de $\mathcal{H}(D_{u,v})$ inférieures à ϵ_0. Nous appelons $\mathcal{H}'(D_{u,v})$ le résultat de cette mise à zéro.

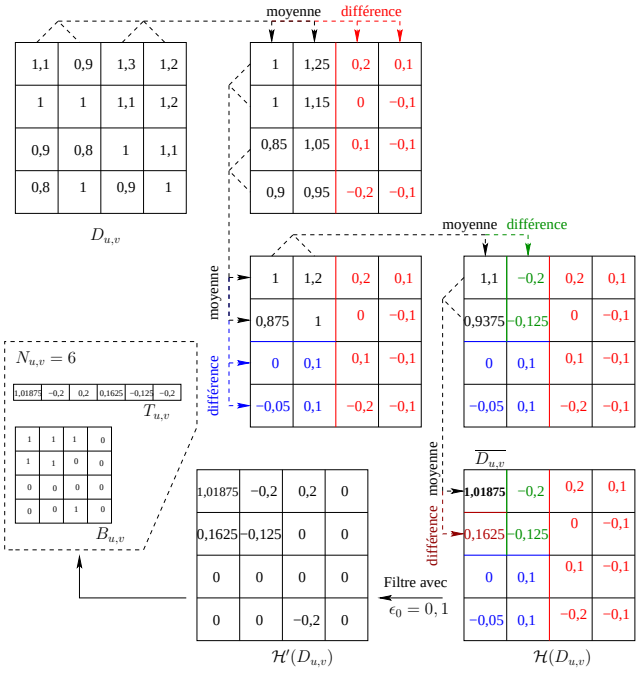

Figure 4: Exemple de compression dans le cas d'un sous-espace de taille $N = 4$.

La transformée $\mathcal{H}'(D_{u,v})$ est ensuite recodée sous la forme d'un couple $(B_{u,v}[N][N], T_{u,v}[N_{u,v}])$. $B_{u,v}$ est une matrice de bits qui indique les valeurs nulles ou non de $\mathcal{H}'(D_{u,v})$, tandis que $T_{u,v}$ est un tableau linéaire contenant uniquement les $N_{u,v}$ valeurs non nulles de $\mathcal{H}'(D_{u,v})$. Quelque soit le sous-espace $D_{u,v}$ considéré, le plan de bits occupe ainsi un espace mémoire de taille $N^2/8$ octets, tandis le nombre d'octets nécessaires au codage du tableau $T_{u,v}$ dépend lui directement de $N_{u,v}$ dépendant lui-même du seuil ϵ_0. Par ailleurs, l'association entre $T_{u,v}$ et $B_{u,v}$ se fait en parcourant la matrice $B_{u,v}$ de gauche à droite et de haut en bas. Ainsi le premier bit de $B_{u,v}$ non nul correspond à la première entrée dans $T_{u,v}$, et ainsi de suite. La figure 4 résume cette procédure dans le cas d'un sous-espace de taille 4×4.

4.2 Compression globale au niveau du diagnostique

Après la phase de compression par ondelettes de Haar, tous les sous espaces $D_{u,v}$ sont représentés par des structures booléennes $B_{u,v}$ associées à des tableaux $T_{u,v}$ contenant les valeurs non nulles de $\mathcal{H}'(D_{u,v})$.

Ce premier niveau de compression ne tient pas compte des corrélations qu'il peut y avoir entre les différents sous-espaces d'un même diagnostique D. C'est pourquoi, une seconde phase de compression est réalisée en deux étapes :

1. **Création d'un dictionnaire commun** en fusionnant toutes les tables de valeurs non nulles $T_{u,v}$ à ϵ_1 près, afin de former une unique table T_D commune à l'ensemble des sous-espaces du diagnostique.

2. **Compression du dictionnaire commun** T_D par un algorithme de partitionnement d'erreur ϵ_2.

Pour effectuer la **fusion des tables** $T_{u,v}$, ces tables sont tout d'abord triées par ordre décroissant, puis elles sont progressivement fusionnées par inter-classement en retirant les doublons à ϵ_1 près. Ce processus permet d'obtenir la table T_D de taille N_{T_D}, dont les éléments sont triés par ordre décroissant, soit $\forall k,\ T_D[k] > T_D[k+1]$. Par ailleurs, la fusion des valeurs réelles étant réalisée à ϵ_1 près, nous avons $\forall k,\ T_D[k] \geq T_D[k+1] + \epsilon_1$.

En triant les tables $T_{u,v}$, nous avons perdu la relation qu'il y avait entre les booléens de la matrice $B_{u,v}$ et les éléments de $T_{u,v}$. Il est donc nécessaire d'introduire une table intermédiaire d'index $I_{u,v}$ permettant de restaurer cette relation. Ceci revient notamment à effectuer un adressage indirect. $I_{u,v}$ est donc définie de telle sorte que $T_D[I_{u,v}[k]]$ soit égale à $T_{u,v}[k]$ à ϵ_1 près. Ainsi par construction, la taille des index $I_{u,v}$ varie en fonction de $N_{u,v}$. En pratique, des entiers codés sur 24 bits sont suffisants, puisqu'ils permettent l'obtention de $2^{24} = 16\ 777\ 216$ entrées possibles dans T_D. Pour résumer, à ce stade du processus de compression, un sous-espace $D_{u,v}$ est à présent défini sous la forme d'un couple $(B_{u,v}, I_{u,v})$ associé au dictionnaire T_D commun à tous les sous-espaces du même diagnostique D.

Nous effectuons ensuite la **compression du dictionnaire commun** T_D. Pour cela, nous utilisons un partitionnement basé sur l'histogramme de distribution des valeurs de T_D

en considérant notamment le fait que les éléments de T_D sont triés par ordre décroissant et que la valeur $T_D[0]$ représente ainsi le maximum des valeurs de T_D.

La table T_D est fractionnée en K partitions dont les bornes v_l avec $l \in [0, K-1]$, sont définies par $v_l = T_D[0] - l \times \epsilon_2 \times 2^8$. De plus, si $T_D[k]$ se trouve au sein de la partition l, c'est-à-dire si $v_l \geq T_D[k] \geq v_{l+1}$, alors il existe deux entiers positifs a_m et a_{m+1} avec $a_m < a_{m+1}$ tels que :

$$v_l - a_m \times \epsilon_2 \geq T_D[k] \geq v_l - a_{m+1} \times \epsilon_2.$$

L'idée consiste alors à stocker $T_D[0]$ et l'ensemble des entiers a_m, plutôt que les valeurs réelles explicites $T_D[k]$. Au final, la table T_D compressée devient une liste de partitions où chaque partition est une structure contenant une valeur maximale initiale et un tableau d'entiers. Il est à noter que le nombre de subdivisions au sein d'une partition est restreint à 256 limitant ainsi le nombre de bits nécessaires au stockage des a_m. La table ainsi compressée est notée T'_D. La figure 5 illustre un exemple en prenant $\epsilon_2 = 0.01$ pour une table T_D de taille $N_{T_D} = 15$ ayant comme valeurs :

$$T_D = \{9.9\ ;9.8\ ;9.07\ ;9.03\ ;9\ ;8.7\ ;7.4\ ;7.1\ ;7.09\ ;7.08\ ;7.05\ ;7\ ;4.6\ ;4\ ;3.9\}.$$

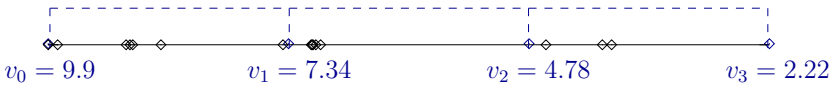

Figure 5: Exemple de partitionnement pour un dictionnaire de taille $N_{T_D} = 15$.

Nous avons ainsi trois partitions $\{9.9\ ;9.8\ ;9.07\ ;9.03\ ;9\ ;8.7\ ;7.4\}$, $\{7.1\ ;7.09\ ;7.08\ ;7.05\ ;7\}$ et $\{4.6\ ;4\ ;3.9\}$ délimitées par les différentes bornes $v_0 = 9.9$, $v_1 = 7.34$, $v_2 = 4.78$ et $v_3 = 2.22$. Nous pouvons alors exprimer la première partition sous la forme :

$$\{v_0 = 9.9\ ;v_0 - 10 \times \epsilon_2\ ;v_0 - 83 \times \epsilon_2\ ;v_0 - 87 \times \epsilon_2\ ;v_0 - 90 \times \epsilon_2\ ;v_0 - 93 \times \epsilon_2\ ;v_0 - 107 \times \epsilon_2\}$$

ou encore $\{9.9\ ;-10\ ;-83\ ;-87\ ;-90\ ;-93\ ;-107\}$. En faisant cette manipulation pour l'ensemble des partitions, nous obtenons la table compressée T'_D suivante :

$$T'_D = \{9.9\ ;-10\ ;-83\ ;-87\ ;-90\ ;-93\ ;-107\ ;7.1\ ;-1\ ;-2\ ;-5\ \ -10\ ;4.6\ ;-60\ ;-70\}.$$

Notons que cette technique de compression ne fonctionne que si l'écart entre les valeurs de T_D se distribuent de manière non-uniforme avec des agrégats importants de valeurs très proches.

5 Récapitulatif de la méthode proposée

La figure 6 résume les différentes étapes de l'algorithme qui a été mis en oeuvre et le tableau 1 l'ensemble des symboles employés.

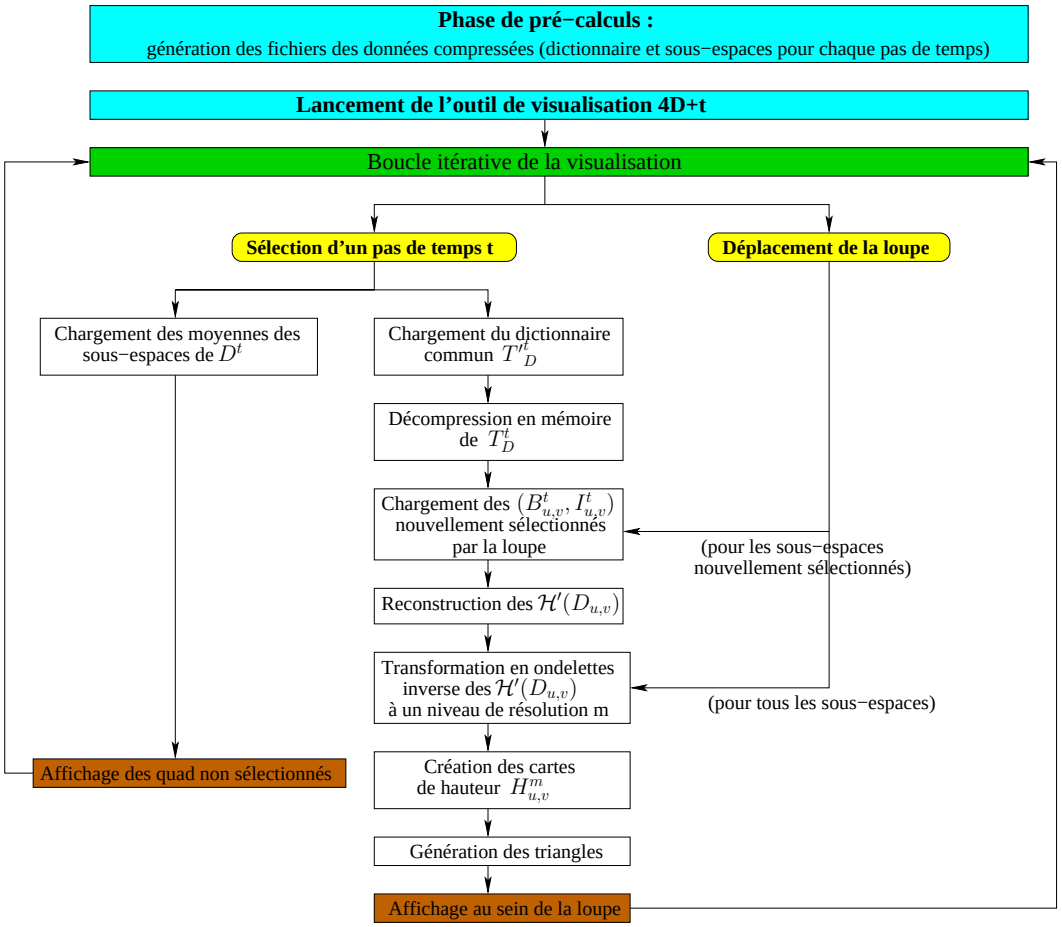

Figure 6: Schéma algorithmique de l'ensemble de la procédure mise en place

Symboles	**Significations**
n	Dimension de l'espace (dans notre cas n=2)
N	Résolution de la discrétisation
t	Temps
$\vec{x}$	Vecteur de position
$\vec{v}$	Vecteur de vitesse
$f(\vec{v},\vec{x},t)$	Fonction de distribution de particules chargées
(u,v)	Coordonnées dans la grille de l'espace Euclidien (x,y)
(i,j)	Coordonnées dans la grille de l'espace des vitesses (v_x,v_y)
D^t ou D	Diagnostique au temps t
$D_{u,v}$	Sous-espace du diagnostique D
$D_{u,v}[i][j]$	Elément du sous-espace $D_{u,v}$ ou densité de particules
$\overline{D_{u,v}}$	Densité moyenne du sous-espace $D_{u,v}$
L_{u_L,v_L,S_L}	Loupe
S_L	Taille de la loupe
(u_L,v_L)	Position de la loupe dans D
$c_{u,v}$	Couleur du "quad" carré (u,v)
F	Fonction de transfert
N_T	Nombre de triangles générés
m	Niveau de résolution
$H^m[i][j]$	Carte de hauteur de $D_{u,v}[i][j]$
$\mathcal{H}(D_{u,v})$	Transformée en ondelettes de Haar de $D_{u,v}$
ϵ_0	Seuil du filtre de mise à zéro
$\mathcal{H}'(D_{u,v})$	Résultat de la mise à zéro de $\mathcal{H}(D_{u,v})$
$N_{u,v}$	Nombre de valeurs non nulles de $\mathcal{H}'(D_{u,v})$
$T_{u,v}$	Tableau des valeurs non nulles de $\mathcal{H}'(D_{u,v})$
$B_{u,v}$	Matrice de bits de $\mathcal{H}'(D_{u,v})$
ϵ_1	Erreur associée à la fusion des tables
T_D	Dictionnaire commun de D
N_{T_D}	Taille du dictionnaire commun T_D
$I_{u,v}$	Table d'index vers le dictionnaire commun T_D
ϵ_2	Erreur associée à la compression de T_D
K	Nombre de partitions du dictionnaire commun T_D
v_k	Bornes des partitions du dictionnaire commun T_D
a_k	Subdivisions des partitions du dictionnaire commun T_D
T'_D	Dictionnaire commun compressé de D
ϵ	Erreur globale du schéma de compression

Tableau 1: Récapitulatif des symboles employés

Le processus débute par une phase de pré-calculs qui n'est donc réalisée qu'une seule fois pour une simulation donnée. Cette phase initiale consiste à effectuer la compression de l'ensemble des données 4-D+t, pour ensuite les stocker dans des fichiers. Il est à noter que les diagnostiques D^t sont traités indépendamment les uns des autres. A ce stade de l'initialisation, nous avons ainsi pour chaque diagnostique D^t :

- un dictionnaire commun T_D^t stocké sous une forme compressée dans un fichier,
- et un ensemble de couples $(B_{u,v}^t, I_{u,v}^t)$ élaborés pour chaque sous-espace $D_{u,v}$ du diagnostique D. Ces couples sont également stockés dans des fichiers.

Pour éviter, dans le cas où N est grand, la création d'un nombre trop important de fichiers qui pénaliserait les accès disque, les sous-espaces $(B_{u,v}^t, I_{u,v}^t)$ sont regroupés dans un même fichier par lots de 4×4 sous-espaces.

Une fois cette première phase réalisée, l'application peut être exécutée. L'utilisateur commence tout d'abord par sélectionner un pas de temps et donc un diagnostique à visualiser. Il est à noter qu'initialement, aucune information n'est en mémoire. Ce n'est donc qu'à cet instant que le système charge le dictionnaire compressé T'_D concerné et le décompresse en temps linéaire pour obtenir le dictionnaire commun T_D. T_D restera ensuite en mémoire tant que l'utilisateur ne change pas de pas de temps. Par ailleurs, une image des moyennes des sous-espaces se trouvant dans le même fichier, est également chargée permettant par la suite la gestion de l'affichage en dehors de la loupe.

Ensuite, dans une seconde phase, les lots des sous-espaces recouverts par la loupe sont chargés. Il est à noter que selon le choix des lots, il peut y avoir plus de sous-espaces chargés qu'il n'y en a effectivement au sein de la loupe. Pour tous ces sous-espaces, le couple $(B_{u,v}^t, I_{u,v}^t)$ est utilisé conjointement avec la dictionnaire T_D pour reconstruire, non pas le sous-espace $D_{u,v}$ mais sa transformée en ondelette de Haar $\mathcal{H}'(D_{u,v})$. Ces transformées correspondent à des images 2-D de taille $N \times N$ de nombres réels en précision double. Cette information reste en mémoire tant que l'utilisateur ne déplace pas la loupe.

La troisième phase concerne à présent l'affichage. D'abord les "quad" non concernés par la loupe sont affichés en utilisant les valeurs moyennes préalablement chargées et stockées pour l'ensemble du diagnostique. Ensuite seuls les sous-espaces concernés par la loupe sont décompressés en appliquant une transformation en ondelettes de Haar inverse, mais uniquement à un niveau hiérarchique m donné, qui est fonction de la position de l'observateur et fonction de la position du sous-espace au sein de la loupe. Ceci permet de créer les cartes de hauteur $H_{u,v}^m$ à des résolutions m variables. Enfin ces cartes de hauteur sont utilisées pour générer les triangles envoyés finalement à la carte graphique.

Lorsque l'utilisateur déplace la loupe de nouveaux lots sont éventuellement chargés et décompressés. Un système de cache permet de minimiser les accès sachant que la loupe ne se déplace toujours que d'un sous-espace à la fois vers le haut, le bas, la gauche ou la droite.

Lorsque l'utilisateur change de pas de temps, c'est l'ensemble de la procédure qui est réitéré, constituant le type d'interaction qui coûte le plus cher en terme de performance.

Toutefois, un choix judicieux pour le taux de compression et la taille de loupe permet de maintenir une performance temps réel même lors d'un changement de diagnostique.

Notons par ailleurs que nous ne sommes pas limités par le nombre de pas de temps, la seule limite physique étant celle de la capacité du disque dur. La performance de l'application est ainsi paramétrée par la taille N de la simulation, le taux de compression obtenu pour T_D et la dimension de la loupe et des lots de sous-espaces. Evidemment, les vitesses des disques, du processeur et de la carte graphique jouent également un rôle. Mais c'est en fonction de ces données physiques qu'il faut fixer les paramètres que nous venons d'énumérer.

Dans le paragraphe qui suit nous effectuons une analyse détaillée des résultats obtenus par notre approche.

6 Résultats

Nous venons de voir en détails les différents mécanismes de l'outil de visualisation que nous avons implanté afin d'analyser des données 4-D+t issues d'une simulation de plasmas. Nous allons désormais présenter les différentes vues offertes par cet outil, ainsi que les taux de compression que nous avons obtenus permettant une application temps réel.

6.1 Technique de visualisation

Par soucis de compréhension, nous avons jusqu'à présent considéré uniquement le cas où le diagnostique D^t est exprimé dans l'espace Euclidien (x, y) avec ses sous-espaces $D^t_{u,v}$ exprimés dans l'espace des vitesses (v_x, v_y). Mais l'ensemble du mécanisme est en fait réalisé pour quatre espaces d'intégration différents (x, y), (v_x, v_y), (x, v_x) et (y, v_y) associés respectivement aux quatre sous-espaces (v_x, v_y), (x, y), (y, v_y) et (x, v_x).

La figure 7 présente la visualisation d'un diagnostique selon quatre différents espaces d'intégration.

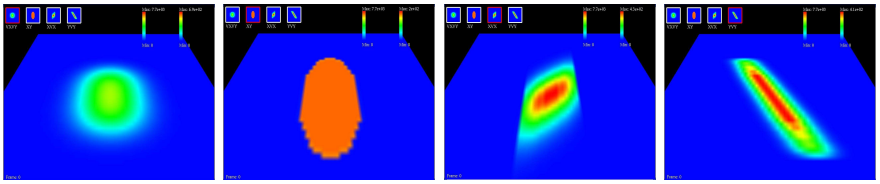

Figure 7: Visualisation d'un diagnostique selon différents espaces d'intégration, respectivement (v_x, v_y), (x, y), (x, v_x) et (y, v_y)

Il est ainsi possible, en cours d'exécution, de modifier le choix de l'espace d'intégration. Ce choix est clairement identifiable sur l'application par un encadré rouge (étiquette (1) de la figure 8), ainsi que les axes choisis pour l'espace de représentation et les sous espaces (étiquettes de (4) à (7)). Par ailleurs, les étiquettes (2) et (3) indiquent les palettes de

couleurs ainsi que les valeurs minimum et maximum de l'espace d'intégration courant et de ses sous-espaces. Il faut également noter que les figures 7 et 8 présentent l'interface de l'application lorsque la loupe n'est pas activée. L'application permet alors une visualisation dans le mode habituellement utilisé par les physiciens.

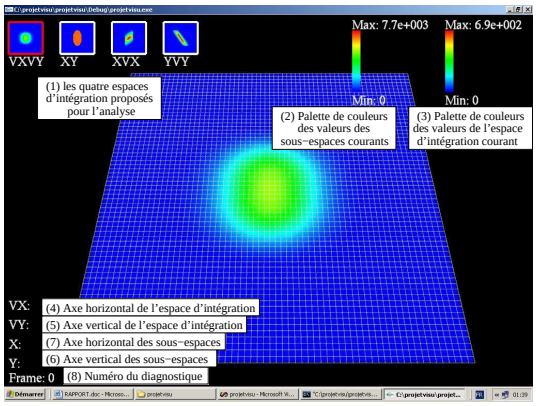

Figure 8: Capture d'écran de l'application avec désactivation de la loupe

La figure 9 montre ensuite l'interface de l'application lorsque la loupe est activée. Rappelons qu'au sein de la loupe, les sous-espaces visualisés correspondent aux données 4-D dont deux variables ont été fixées.

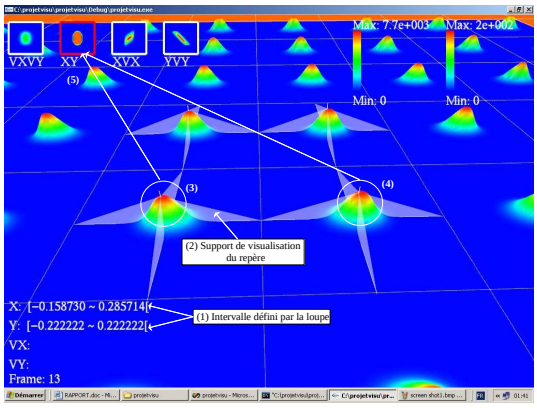

Figure 9: Capture d'écran de l'application avec activation de la loupe

L'étiquette (1) de cette figure présente les plages de valeurs qui sont couvertes par la loupe dans l'espace d'intégration de départ. Par ailleurs, un support transparent est employé au centre de la loupe pour indiquer les axes du sous espace considéré (étiquette (2)).

Enfin, nous pouvons observer sur cette figure une variation de la densité de particules entre le sous espace présenté par l'étiquette (3) et celui de l'étiquette (4). Ainsi en (3) la densité est centrée en $(0, 0)$ alors qu'en (4) une diminution de la densité peut être observée dans la portion négative des v_x, ainsi qu'une augmentation positive dans leur partie positive. L'outil implanté permet ainsi de suivre l'évolution des particules au cours de la simulation. Il est à noter que la représentation habituellement employée ne le permet pas puisqu'en (5) aucune variation de couleurs au centre de la grille n'est visible.

Enfin, la figure 10 montre l'interface de l'application lorsque la loupe et l'outil de pointage sont activés.

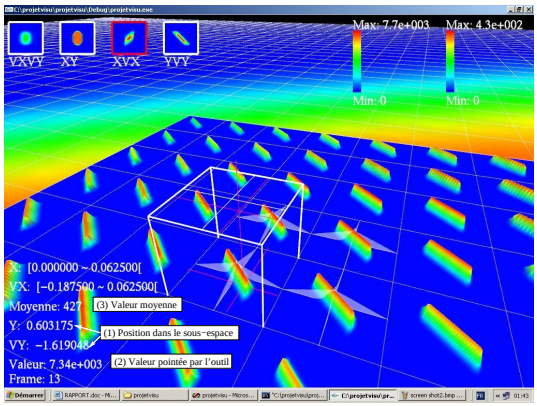

Figure 10: Capture d'écran de l'application montrant la loupe et l'outil de pointage de valeurs

L'outil de pointage permet de connaître avec précision la valeur de la densité (étiquette (2)) pour un point donné du sous-espace considéré (étiquette (1)). De plus, la valeur moyenne de la densité au sein du sous-espace défini par le cube en blanc, est également affichée (étiquette (3)).

6.2 Compression des données

Les résultats présentés ici ont été obtenus sur un PC standard doté d'un processeur AMD Athlon XP 3000+ avec 1Go 300Mhz Dual Channel DDR de mémoire, et ayant un disque dur SATA de 160Go avec 8Mo de cache et une ATI 9700Pro AGP 8x comme carte graphique.

Les mesures de performances ont été réalisées pour quatre simulations physiques différentes, et pour chacune d'entre-elles pour quatre espaces d'intégration différents. Mais les résultats obtenus étant similaires pour l'ensemble de ces quatre simulations, nous ne présentons ici que les résultats relatifs à l'une de ces simulations en considérant l'espace d'intégration (x, y).

Ces quatre simulations ont été exécutées sur des grilles 4-D de l'espace des phases de taille $64\times64\times64\times64$ pour 40 pas de temps, ce qui représente un volume de données de 128Mo pour un diagnostique donné. Afin de simplifier la paramétrisation de la compression, nous avons fixé les trois erreurs à une seule même valeur ($\epsilon_0 = \epsilon_1 = \epsilon_2 = \epsilon$). Expérimentalement, nous avons observé que l'erreur absolue obtenue est inférieure à l'erreur ϵ fixée pour le schéma de compression.

Notons que la phase de pré-calcul dure environ $2-3$s pour compresser un diagnostique donné par rapport à un espace d'intégration particulier. La compression de la simulation entière dure ainsi moins de 6 minutes pour quatre espaces d'intégration différents.

Sur la figure 11 nous pouvons voir que le taux de compression obtenu est supérieur à 90% même pour des précisions faibles ($\epsilon = 10^{-8}$). Le volume de données pour un diagnostique donné est alors compris entre 3Mo ($\epsilon = 1$) et 11Mo ($\epsilon = 10^{-8}$), permettant de charger entre respectivement 5 et 25 pas de temps par seconde si nous considérons une bande passante de disque dur de 50Mo/s.

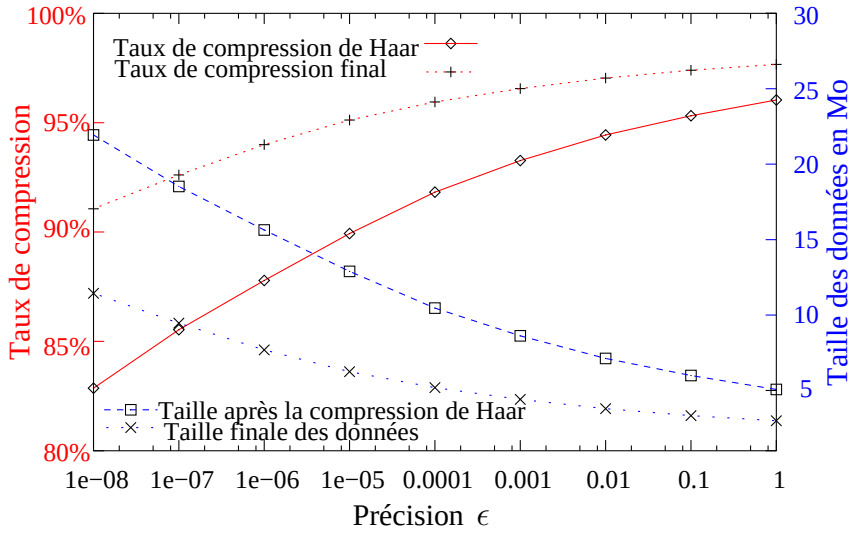

Figure 11: Taux de compression et volume de données d'un diagnostique en Mo après avoir effectué une compression d'une erreur ϵ pour l'espace d'intégration (x, y)

Notons par ailleurs que le temps de décompression du dictionnaire commun est d'au moins $11-12$ms ($\epsilon = 10^{-8}$).

La figure 12 présente le temps passé pour effectuer la transformée inverse de Haar pour différentes résolutions en fonction de la précision ϵ. Ce temps, constant pour les différentes précisions choisies, est compris entre 426ms (pour une résolution 64×64) et 2ms (pour une résolution 4×4).

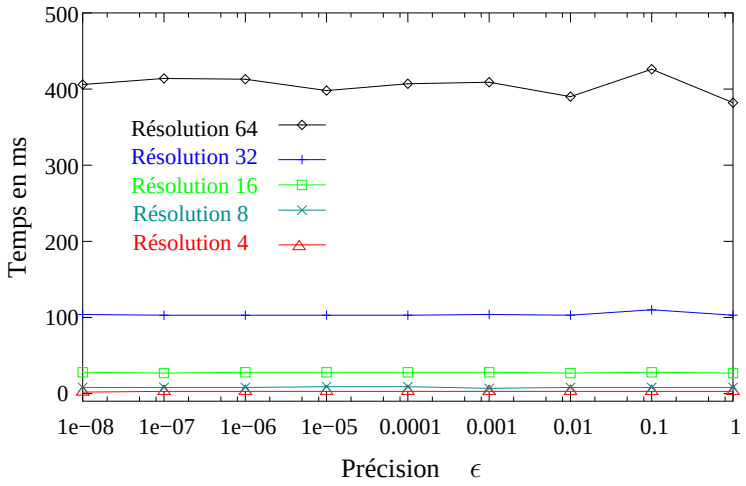

Figure 12: Temps en ms passé au sein de la transformée inverse de Haar pour différentes résolutions $(64, 32, 16, 8, 4)$ selon la précision ϵ pour l'espace d'intégration (x, y)

La figure 13 montre les temps de chargement des sous-espaces sélectionnés par une loupe de taille 8×8, ainsi que le temps de construction des cartes de hauteurs pour la résolution maximale (reconstruction de $\mathcal{H}'(D_{u,v})$ + transformée inverse de Harr). La troisième courbes (somme des deux premières) représente le temps total nécessaire lors du déplacement de la loupe.

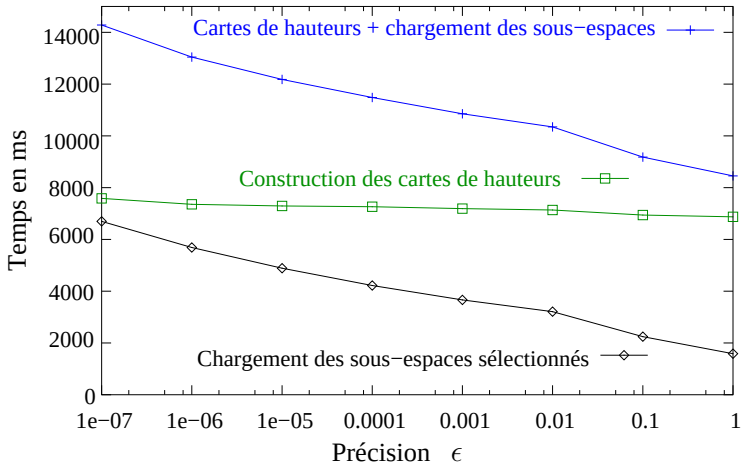

Figure 13: Temps maximal (en ms) passé quand la loupe (8×8) est déplacée selon l'erreur ϵ

Au final, une fois la phase de pré-calcul effectuée, notre méthode permet de visualiser en temps réel des données 4-D de la forme $f(x, y, v_x, v_y)$ issues d'une simulation de plasma. Considérons désormais l'emploi d'une loupe de taille 8×8 pour évaluer les fréquences d'affichage de notre outil. Cette loupe dispose d'une résolution maximale de 64×64 en son centre, qui décroît graduellement vers les bords en une résolution minimale de 16×16. Les fréquences d'affichages alors mesurées sont de : $15-20$fps lors du changement de diagnostique, $15-20$fps lors du changement de l'espace d'intégration et $25 - 34$fps quand la loupe est déplacée.

7 Conclusion et perspectives

Les différents composants d'un outil de visualisation de données 4-D+t issues de simulations des plasmas ont été cernés. Cet outil se caractérise par son choix dans la représentation des données 4-D+t et par l'implantation d'une méthode de compression totalement dédiée à la visualisation.

Le système de visualisation est basé sur l'emploi d'une loupe faisant apparaître localement les détails de la fonction de distribution par l'emploi de cartes de hauteurs. Les données en dehors de la loupe correspondent, quant à elles, aux densité moyennes affichées à l'aide d'une carte de couleurs. Les caractéristiques de la loupe (taille, répartition des résolutions et répartition de la qualité de rendu) peuvent par ailleurs être adaptées en fonction de la machine employée afin de visualiser un grand volume de données en temps interactif.

Ensuite, il faut noter que le principal problème pour le développement d'un outil interactif traitant une masse de données de plusieurs giga-octets réside dans le goulot d'étranglement que constitue la transmission des données entre les interfaces de stockage et la mémoire centrale d'une machine. Les temps de transferts rendent en effet impossible la satisfaction des contraintes de temps réel. C'est pourquoi, nous avons proposé un schéma de compression des données avec perte, afin de palier le problème de la lenteur des transferts entre interfaces de stockages et mémoire centrale des machines. Ce schéma de compression présente deux niveaux de compression. Le premier niveau se base sur la technique classique de compression par ondelettes afin de compresser les différents sous-espaces d'un diagnostique donné, tandis que le second niveau effectue une compression globale du diagnostique.

L'outil développé est ainsi un outil performant, souple et facile d'utilisation. Il tire parfaitement profit de toutes les techniques innovantes détaillées dans ce rapport afin de permettre une analyse intuitive et interactive des phénomènes les plus complexes de la cinétique des plasmas. Les travaux futurs se tournent désormais vers une phase de parallélisation permettant le traitement de données issues de simulation sur grilles de tailles supérieures à celles pour le moment considérées.

Références

[AC91] B. Alpern and L. Carter. The hyperbox. In *Proceedings of the 2nd conference on Visualization '91*, pages 133–139. IEEE Computer Society Press, 1991.

[BC87] R. A. Becker and W. S. Cleveland. Brushing scatterplots. *Technometrics*, 29(2):127–142, 1987.

[BMMS91] A. Buja, J. A. McDonald, J. Michalak, and W. Stuetzle. Interactive data visualization using focusing and linking. In *Proceedings of the 2nd conference on Visualization '91*, pages 156–163. IEEE Computer Society Press, 1991.

[CCKT83] J. M. Chambers, W. S. Cleveland, B. Kleiner, and P. Tukey. Graphical method for data analysis, 1983.

[Cle93] W. S. Cleveland. *Visualizing Data*. Hobart Press, Summit, NJ, USA, 1993.

[CS94a] C. Constantinescu and J. A. Storer. Improved techniques for single-pass adaptive vector quantization. *Proceedings of the IEEE*, 82(6):933–939, June 1994.

[CS94b] C. Constantinescu and J. A. Storer. Online adaptive vector quantization with variable size codebook entries. *Inf. Process. Manage.*, 30(6):745–758, 1994.

[Dau88] I. Daubechies. Orthogonal bases of compactly supported wavelets. *Comm. Pure Appl. Maths*, 16:909–996, 1988.

[DGH03] H. Doleisch, M. Gasser, and H. Hauser. Interactive feature specification for focus+context visualization of complex simulation data. In *Proceedings of the symposium on Data visualisation 2003*, pages 239–248. Eurographics Association, 2003.

[DJL92] R. A. Devore, B. Jawerth, and B. J. Lucier. Image compression through wavelet transform coding. *IEEE Transactions on Information Theory*, 38(2 (Part II)):719–746, March 1992.

[DWS+97] M. A. Duchaineau, M. Wolinsky, D. E. Sigeti, M. C. Miller, C. Aldrich, and M. B. Mineev-Weinstein. ROAMing terrain: real-time optimally adapting meshes. In *IEEE Visualization*, pages 81–88, 1997.

[FB90a] S. K. Feiner and C. Beshers. Visualizing n-dimensional virtual worlds with n-vision. In *Proceedings of the 1990 symposium on Interactive 3D graphics*, pages 37–38. ACM Press, 1990.

[FB90b] S. K. Feiner and C. Beshers. Worlds within worlds: Metaphors for exploring n-dimensional virtual worlds. In Scott E. Hudson, editor, *User interface software and technology*. ACM Press, October 1990.

[FS03a] F. Filbet and E. Sonnendrücker. Comparison of Eulerian Vlasov solvers. *Comput. Phys. Comm.*, 150(3):247–266, 2003.

[FS03b] F. Filbet and E. Sonnendrücker. Numerical methods for the vlasov equation. In F. Brezzi, A. Buffa, S. Corsaro, and Murli A., editors, *in Numerical Mathematics and Advanced Applications, ENUMATH 2001*. Springer-Verlag, 2003.

[FSB01] F. Filbet, E. Sonnendrücker, and P. Bertrand. Conservative numerical schemes for the vlasov equation. *J. Comput. Phys.*, 172(1):166–187, 2001.

[Fur86] G. W. Furnas. Generalized fisheye views. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 16–23. ACM Press, 1986.

[GPS03] M. Gutnic, I. Paun, and E. Sonnendrücker. Vlasov simulations on an adaptive phase-space grid. Technical Report 03032, Institut de Recherche Mathématiques Avancée, Université Louis Pasteur, Strasbourg, 2003.

[GPW89] G. Grinstein, R. Pickett, and M. G. Williams. Exvis: An exploratory visualization environment. In *Graphics Interlace '89*, pages 254–261, London, Ontario, 1989.

[ID87] A. Inselberg and B. Dimsdale. Parallel coordinates for visualizing multi-dimensional geometry. In *CG International '87 on Computer graphics 1987*, pages 25–44. Springer-Verlag New York, Inc., 1987.

[ID90] A. Inselberg and B. Dimsdale. Parallel coordinates: a tool for visualizing multi-dimensional geometry. In *Proceedings of the 1st conference on Visualization '90*, pages 361–378. IEEE Computer Society Press, 1990.

[IRC87] A. Inselberg, M. Reif, and T. Chomut. Convexity algorithms in parallel coordinates. *J. ACM*, 34(4):765–801, 1987.

[JN02] S. Jayaraman and C. North. A radial focus+context visualization for multi-dimensional functions. In *Proceedings of the conference on Visualization '02*, pages 443–450. IEEE Computer Society, 2002.

[Kan00] E. Kandogan. Star coordinates: A multi-dimensional visualization technique with uniform treatment of dimensions. In *IEEE Information Visualization, Hot Topics*, pages 4–8, 2000.

[LBG80] Y. Linde, A. Buzo, and R. Gray. An algorithm for vector quantizer design. *IEEE Trans. on Communications*, 1:84–95, Jan. 1980.

[Lev91] H. Levkowitz. Color icons: merging color and texture perception for integrated visualization of multiple parameters. In *Proceedings of the 2nd conference on Visualization '91*, pages 164–170. IEEE Computer Society Press, 1991.

[Lew95] J. Lewalle. *Tutorial on Continuous Wavelet Analysis of Experimental Data*, April 1995.

[LWW90] J. LeBlanc, M. O. Ward, and N. Wittels. Exploring n-dimensional databases. In *Proceedings of the 1st conference on Visualization '90*, pages 230–237. IEEE Computer Society Press, 1990.

[Mey92] Y. Meyer, editor. *Wavelets and applications*, volume 20 of *RMA: Research Notes in Applied Mathematics*, Paris, 1992. Masson.

[MGTS90] T. Mihalisin, E. Gawlinski, J. Timlin, and J. Schwegler. Visualizing a scalar field on an n-dimensional lattice. In *Proceedings of the 1st conference on Visualization '90*, pages 255–262. IEEE Computer Society Press, 1990.

[MRC91] J. D. Mackinlay, G. G. Robertson, and S. K. Card. The perspective wall: detail and context smoothly integrated. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 173–176. ACM Press, 1991.

[MSW+02] K.-L. Ma, G. Schussman, B. Wilson, K. Ko, J. Qiang, and R. Ryne. Advanced visualization technology for terascale particle accelerator simulations. In *Proceedings of the 2002 ACM/IEEE conference on Supercomputing*, pages 1–11. IEEE Computer Society Press, 2002.

[MTS91a] T. Mihalisin, J. Timlin, and J. Schwegler. Visualization and analysis of multivariate data: a technique for all fields. In *Proceedings of the 2nd conference on Visualization '91*, pages 171–178. IEEE Computer Society Press, 1991.

[MTS91b] T. Mihalisin, J. Timlin, and J. Schwegler. Visualizing multivariate functions, data, and distributions. *IEEE Comput. Graph. Appl.*, 11(3):28–35, 1991.

[Mul97] C. Mulcahy. "image compression using the haar wavelet transform". *Spelman Science and Mathematics Journal*, 1(1):22–31, 1997.

[PG88] R. M. Pickett and G. G. Grinstein. "iconographics displays for visualizing multidimensional data". In *IEEE Conference on Systems, Man, and Cybernetics*, pages 514–519, 1988.

[RB99] R. M. Rao and A. S. Bopardikar. *Wavelet Transforms: Introduction to Theory and Applications*. Addison-Wesley, 1999.

[RC94] R. Rao and S. K. Card. The table lens: merging graphical and symbolic representations in an interactive focus + context visualization for tabular information. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 318–322. ACM Press, 1994.

[RE04] G. Reina and T. Ertl. Volume visualization and visual queries for large high-dimensional datasets. In *Proceedings of the symposium on visualisation 2004*. Eurographics Association, 2004.

[RT97] .R Rao and .T Tenev. Extending table lens to multidimensional data and olap operations. In *Euro-American Workshop: Visualization of Information and Data*, pages 24–25, June 1997.

[SFF+03] E. Sonnendrücker, F. Filbet, A. Friedman, E. Oudet, and J.-L. Vay. Vlasov simulations of beams with a moving grid. Technical Report 03031, Institut de Recherche Mathématiques Avancée, Université Louis Pasteur, Strasbourg, 2003.

[SGB91] S. Smith, G. Grinstein, and R. D. Bergeron. Interactive data exploration with a supercomputer. In *Proceedings of the 2nd conference on Visualization '91*, pages 248–254. IEEE Computer Society Press, 1991.

[VF02] E. Violard and F. Filbet. Parallelization of a vlasov solver by communication overlapping. In *proceedings of PDPTA'02*, pages 1049–1055. CSREA Press, June 2002.

[vv93] J. J. van Wijk and R. van Liere. Hyperslice: visualization of scalar functions of many variables. In *Proceedings of the 4th conference on Visualization '93*, pages 119–125, 1993.

[War94] M. O. Ward. Xmdvtool: integrating multiple methods for visualizing multivariate data. In *Proceedings of the conference on Visualization '94*, pages 326–333. IEEE Computer Society Press, 1994.

[WMQR02] B. Wilson, K.-L. Ma, J. Qiang, and R. Ryne. Interactive visualization of particle beams for accelerator design. In *Workshop on High Performance Computing in Particle Accelerator Science and Technology*. 2002 International Conference on Computational Science, April 2002.

Table des matières

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

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

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