Réduction du temps de calcul dans la résolution de Vlasov-Poisson

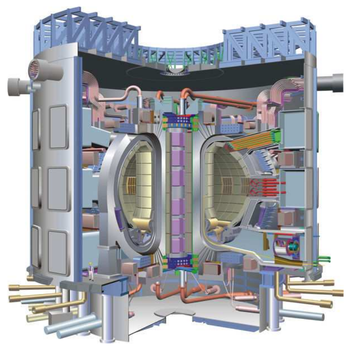
Anaïs Crestetto[a] sous la direction de Philippe Helluy[b]

Université de Strasbourg, INRIA Nancy - Grand Est, thèse financée par la DGA

[a]crestetto@math.unistra.fr, [b]helluy@math.unistra.fr

Motivations

Projet **ITER** : produire de l'électricité grâce à l'énergie de fusion $\Longrightarrow$ modélisation du confinement magnétique des plasmas.



Tokamak ITER en construction à Cadarache

Utilisation de **modèles cinétiques** $\Longrightarrow$ calculs lourds car la fonction de distribution dépend en général de 7 variables : $f = f(\vec{x}, \vec{v}, t) \Longrightarrow$ **temps de calcul importants**.

Couplage de modèles cinétiques et fluides

Résolution du système d'équations de **Vlasov-Poisson** :

$$\begin{cases} \frac{\partial f}{\partial t} + v\frac{\partial f}{\partial x} + E(x)\frac{\partial f}{\partial v} = 0, \\ \\ \frac{dE}{dx} = \int_{-\infty}^{+\infty} f(x,v,t)dv - n_0, \end{cases}$$

où $f(x,v,t)$ est la fonction de distribution, $E(x)$ le champ électrique et n_0 la densité du fond d'ions neutralisant.

Méthode PIC (Particle In Cell)

Méthode particulaire : on considère un ensemble de N macroparticules, de positions x_k, de vitesses v_k et de poids ω_k, $k = 1, \dots, N$, et on approche la fonction de distribution f par

$$f_N(x,v,t) = \sum_{k=1}^{N} \omega_k \delta(x - x_k(t))\delta(v - v_k(t)),$$

où δ est la masse de Dirac.

Les macroparticules évoluent suivant les équations du mouvement et on résout le champ électrique sur maillage par différences finies.

La position initiale des macroparticules est choisie de façon pseudo aléatoire, ce qui génère du **bruit numérique**.

Méthode δf

Lorsque la fonction de distribution reste proche d'un état d'**équilibre**, on écrit

$$f = f^0 + \delta f,$$

où f^0 est la solution d'équilibre et δf est une perturbation.

On ne résout que la perturbation par la méthode PIC $\Longrightarrow$ on diminue le bruit.
On obtient la même précision avec moins de particules $\Longrightarrow$ on **diminue le temps de calcul**.

Bibliographie

S.J. Allfrey and R. Hatzky, *A revised δf algorithm for nonlinear PIC simulation*, Computer Physics Communications, 154 (2003) 98-104.

R. D. Sydora, *δf Particle-in-Cell Plasma Simulation Model : Properties and Applications*, Advanced Methods for Space Simulations, edited by H. Usui and Y. Omura, pp. 47-60, 2007.

R.O. Fox, F. Laurent and M. Massot, *Numerical simulation of spray coalescence in an Eulerian framework : Direct quadrature method of moments and multi-fluid method*, Journal of Computational Physics 227, 2008.

Couplage de la méthode des moments à la méthode δf

Lorsque la fonction d'équilibre dépend de la variable d'espace, on la calcule à chaque pas de temps par un **modèle fluide**.
On se donne a priori la forme de f^0 (maxwellienne, waterbags, etc.) et on prend les P premiers moments de l'équation de Vlasov.
On obtient un système d'EDP plus simple que le modèle cinétique car il ne fait plus intervenir la variable vitesse :

$$\partial_t \int v^k f^0 dv + \partial_x \int v^{k+1} f^0 dv + E \int v^k \partial_v f^0 dv$$

$$+\partial_t \int v^k \delta f dv + \partial_x \int v^{k+1} \delta f dv + E \int v^k \partial_v \delta f dv = 0,$$

pour $k = 0, \dots, P-1$.

On définit le moment d'ordre k, en la variable v, d'une fonction f intégrable en v par l'intégrale suivante :

$$M_k(f)(x,t) = \int v^k f(x,v,t) dv.$$

Le système fait donc apparaître les moments de f^0 et ceux de δf.
On considère les P premiers moments de δf nuls et on résout ce système pour obtenir f^0 au pas de temps suivant.

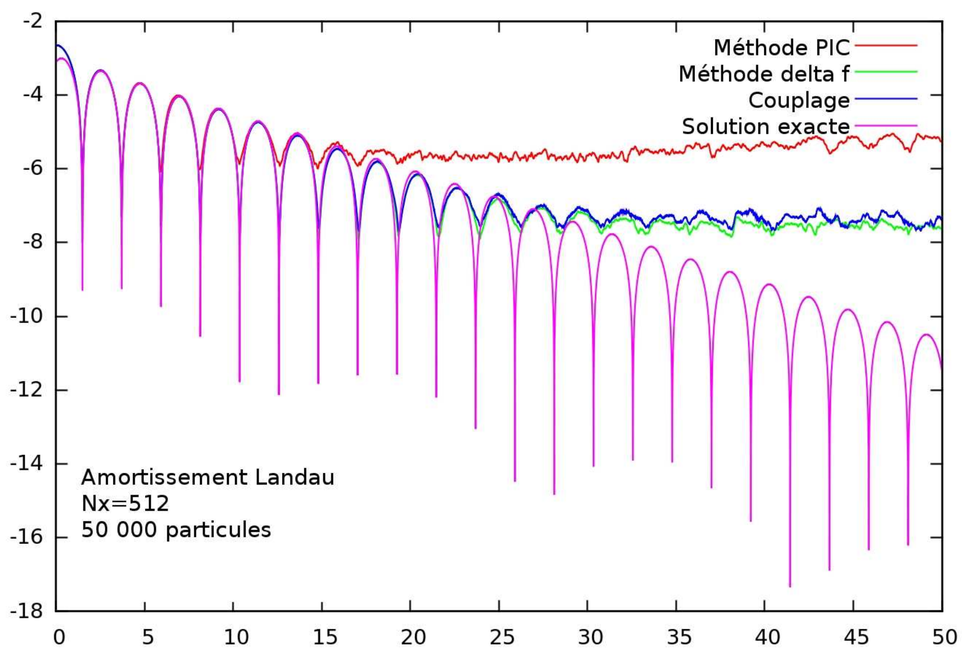
Application à l'amortissement Landau

La fonction de distribution initiale s'écrit :

$$f(x,v,0) = \frac{1+\epsilon\cos(k_0 x)}{\sqrt{2\pi}} e^{-v^2/2},$$

où ϵ est une constante mesurant l'ampleur de la perturbation.

On compare les différentes méthodes :



La méthode δf et le couplage sont nettement plus précis que la méthode PIC : le bruit est réduit.
En raffinant le maillage en x, le couplage donne des résultats comparables à ceux de la méthode δf.

Programmation sur GPU

La programmation sur carte graphique (ou GPU) suscite de plus en plus d'intérêt, à mesure que les performances des cartes augmentent.
L'implémentation de la résolution de Vlasov-Poisson sur GPU, dans le langage OpenCL (Open Computing Language), donne des résultats très satisfaisants.

Le code **PIC** va presque 20 fois plus vite que sur CPU :

	CPU	NVIDIA GeForce GTX 470
Time	75s.	4s.
Accélération		**18,75**

Paramètres : 1048576 particules, 128 mailles en x, 400 itérations.