Simulation de l'atmosphère sur GPU

3 février 2010

1 Contexte

1.1 Évolution des architectures de calcul scientifique

Avec la stagnation de la puissance par processeur, des gains de puissance ne peuvent être réalisés que par la multiplication des processeurs résolvant un même problème en coopérant (parallélisation massive). Dans les années 90 et 2000 cette multiplication s'est faite principalement selon un modèle à mémoire distribuée. Depuis quelques années on "découvre" le potentiel pour le calcul scientifique des processeurs de cartes graphiques (Graphical Processing Unit, GPU). Ces processeurs rassemblent sur une même puce plusieurs dizaines voire centaines de coeurs de calcul strictement synchronisés (modèle SIMD) et beaucoup moins complexes qu'un processeur superscalaire de type x86-64. Des noeuds ont été installés en 2009 dans le supercalculateur civil du CEA (CCRT).

1.2 Systèmes-modèles envisagés

La modélisation du temps et du climat est une grosse consommatrice de puissance de calcul. Néanmoins les GPUs ne peuvent exécuter tels quels ces codes numériques "complexes". Il est nécessaire d'évaluer le potentiel de ces architectures sur des systèmes de complexité intermédiaire représentatifs du type de calcul réalisés par ces modèles, tout en restant beaucoup plus manipulables. Cette approche est celle qui a conduit dans les années 1960 à l'adoption des méthodes numériques encore en cours aujourd'hui. Dans ce cadre, l'équation-modèle de référence est l'équation de Saint-Venant, décrivant l'évolution d'une couche de fluide homogène incompressible soumise à la gravité et la force de Coriolis :

$$\begin{aligned} \rho\left(\partial_t u - fv + u\partial_x u + v\partial_y u\right) + \partial_x p &= 0 \quad (1)\\ \rho\left(\partial_t v + fu + u\partial_x v + v\partial_y u\right) + \partial_y p &= 0 \quad (2)\\ \partial_t p + \partial_x\left(up\right) + \partial_y\left(vp\right) &= 0 \quad (3) \end{aligned}$$

Ici (u, v) sont les deux composantes du vent/courant, p la pression au sol/fond, égale au poids de la colonne d'air/eau, ρ est la masse volumique de l'air/eau (constante) et f est le paramètre de Coriolis (constant). Un modèle météorologique ou climatique résout essentiellement les mêmes équations à la différence près qu'un grand nombre de couches sont prises en compte et non une seule (figure 1).

1.3 Discrétisation spatiale

Les équations (1-6) font intervenir les champs $u(x, y, t)$, $v(x, y, t)$, $p(x, y, t)$ où x et y sont deux coordonnées cartésiennes dans un plan horizontal et t est le temps. On peut les réécrire sous la forme :

$$\partial_t u = \dot{u}(u, v, p), \qquad \partial_t v = \dot{v}(u, v, p), \qquad \partial_t p = \dot{p}(u, v, p)$$

où

$$\begin{aligned} \dot{u} &= \eta V - \partial_x H\\ \dot{v} &= -\eta U - \partial_y H\\ \dot{p} &= -\partial_x U - \partial_y V \end{aligned}$$

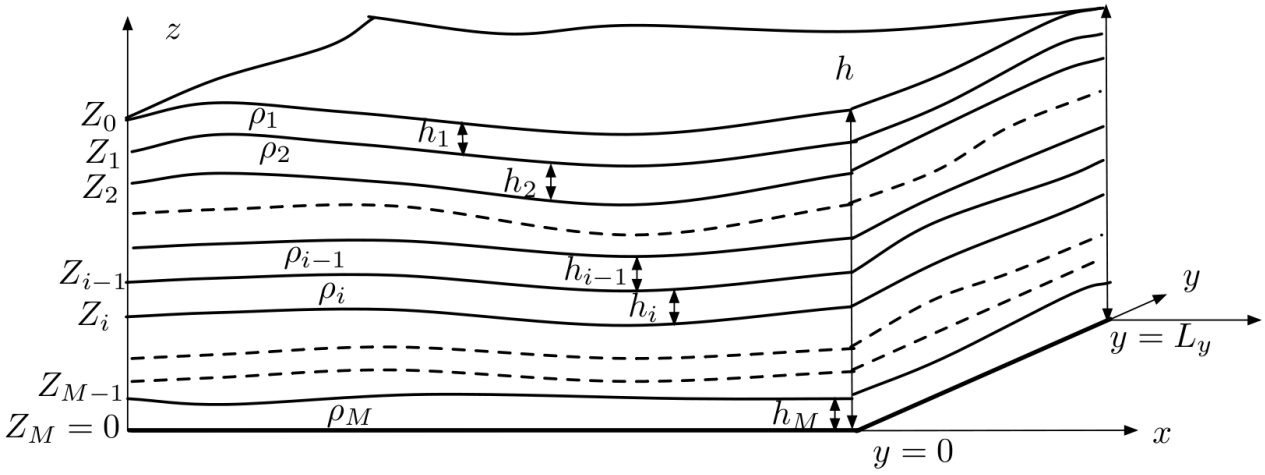

FIG. 1 – Modélisation de l'atmosphère en couches

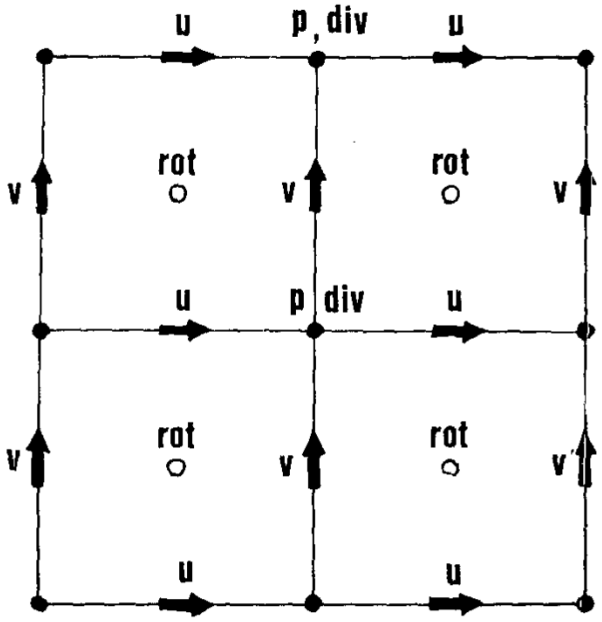

FIG. 2 – Disposition des variables sur la grille

et

$$\begin{aligned} H &= p + \frac{u^2+v^2}{2} \\ \zeta &= \partial_x v - \partial_y u \\ \eta &= \frac{\zeta + f}{p} \\ U &= pu \\ V &= pv. \end{aligned}$$

Les machines ne peuvent faire de calcul que sur un nombre fini de variables. On doit donc *discrétiser* les équations précédentes. Pour cela on définit une grille régulière

$$x_k = k\delta, \qquad y_l = l\delta$$

Étant donné les valeurs d'un champ en (x_k, y_l), ses dérivées peuvent alors être approchées par des

différences finies. Plus précisément on note

$$\begin{array}{ll} p_{kl}(t) = p(x_k, y_l, t), & H_{kl}(t) = H(x_k, y_l, t) \\ u_{kl}(t) = u(x_{k+1/2}, y_l, t), & U_{kl}(t) = U(x_{k+1/2}, y_l, t), \\ v_{kl}(t) = v(x_k, y_{l+1/2}, t), & V_{kl}(t) = V(x_k, y_{l+1/2}, t), \\ \zeta_{kl}(t) = \zeta(x_{k+1/2}, y_{l+1/2}, t), & \eta_{kl}(t) = \eta(x_{k+1/2}, y_{l+1/2}, t) \end{array}$$

Dans le cadre de ce projet on considèrera un domaine de taille $(L_x, L_y) = (M\delta,\, N\delta)$ avec des conditions aux limites périodiques (voir plus bas). Alors :

$$H_{kl} \simeq p_{kl} + \frac{1}{4}\left(u_{kl}^2 + u_{k-1\,l}^2 + v_{kl}^2 + v_{k\,l-1}^2\right) \tag{4}$$

$$\zeta_{kl} \simeq \frac{1}{\delta}\left(v_{k+1\,l} - v_{kl} + u_{k\,l+1} - u_{kl}\right) \tag{5}$$

$$\eta_{kl} \simeq 4\frac{\zeta_{kl} + f}{p_{kl} + p_{k+1\,l} + p_{k\,l+1} + p_{k+1\,l+1}} \tag{6}$$

$$U_{kl} \simeq \frac{1}{2}\left(p_{kl} + p_{k+1\,l}\right) u_{kl} \tag{7}$$

$$V_{kl} \simeq \frac{1}{2}\left(p_{kl} + p_{k\,l+1}\right) v_{kl} \tag{8}$$

$$\tag{9}$$

$$\dot{u}_{kl} \simeq \frac{1}{8}\left(\eta_{kl} + \eta_{k\,l-1}\right)\left(V_{kl} + V_{k+1\,l} + V_{k\,l-1} + V_{k+1\,l-1}\right) - \frac{1}{\delta}\left(p_{k+1\,l} - p_{kl}\right) \tag{10}$$

$$\dot{v}_{kl} \simeq \frac{1}{8}\left(\eta_{kl} + \eta_{k\,l-1}\right)\left(V_{kl} + V_{k+1\,l} + V_{k\,l-1} + V_{k+1\,l-1}\right) - \frac{1}{\delta}\left(p_{k\,l+1} - p_{kl}\right) \tag{11}$$

$$\dot{p}_{kl} \simeq -\frac{1}{\delta}\left(U_{kl} - U_{k-1\,l} + V_{kl} - V_{k\,l-1}\right) \tag{12}$$

Les équations (4-12) permettent, étant donné les valeurs $(u_{kl},\, v_{kl},\, p_{kl})$, de calculer les tendances $(\dot{u}_{kl},\, \dot{v}_{kl},\, \dot{p}_{kl})$.

1.4 Conditions aux limites périodiques

Pour respecter des conditions aux limites périodiques, il suffit de périodiser chaque champ $u, v, p, \ldots$ après l'avoir calculé. La périodisation est l'opération

$$p_{0\,l} := p_{M-2\,l}, \qquad p_{M-1\,l} := p_{1\,l}, \qquad p_{k\,0} := p_{k\,N-2}, \qquad p_{k\,N-1} := p_{k\,1} \tag{13}$$

Les points du maillage situés en $k = 0$, $k = M-1$, $l = 0$, $l = M-1$ sont appelés "cellules fantômes " (Fig. 3).

1.5 Discrétisation temporelle

On discrétise également le temps t : on définit

$$p_{kl}^n = p_{kl}(n\tau)$$

et de même pour u et v. Alors :

$$\dot{p}_{kl}^n \simeq \frac{p_{kl}^{n+1} - p_{kl}^{n-1}}{2\tau}.$$

Cette formule de différence finie permet de calculer p_{kl}^{n+1} étant donné p_{kl}^{n-1} et p_{kl}^n :

$$p_{kl}^{n+1} = p_{kl}^{n-1} + 2\tau\, \dot{p}_{kl}^n. \tag{14}$$

La tendance $\dot{p}_{kl}^n$ est calculée en fonction de $(u_{kl}^n, v_{kl}^n, p_{kl}^n)$. La formule (14) est appelée schéma saute-mouton. Néanmoins (14) ne s'applique qu'à partir de $n > 1$ et il faut un traitement spécial du premier pas de temps ($n = 1$). On utilisera :

$$p_{kl}^1 = p_{kl}^0 + \tau \dot{p}_{kl}^0. \tag{15}$$

Les formules de récurrence (14) et (15) permettent de résoudre des problèmes à valeur initiale, i.e. un problème où (p, u, v) sont connus à $t = 0$ et où on veut les calculer pour $t > 0$.

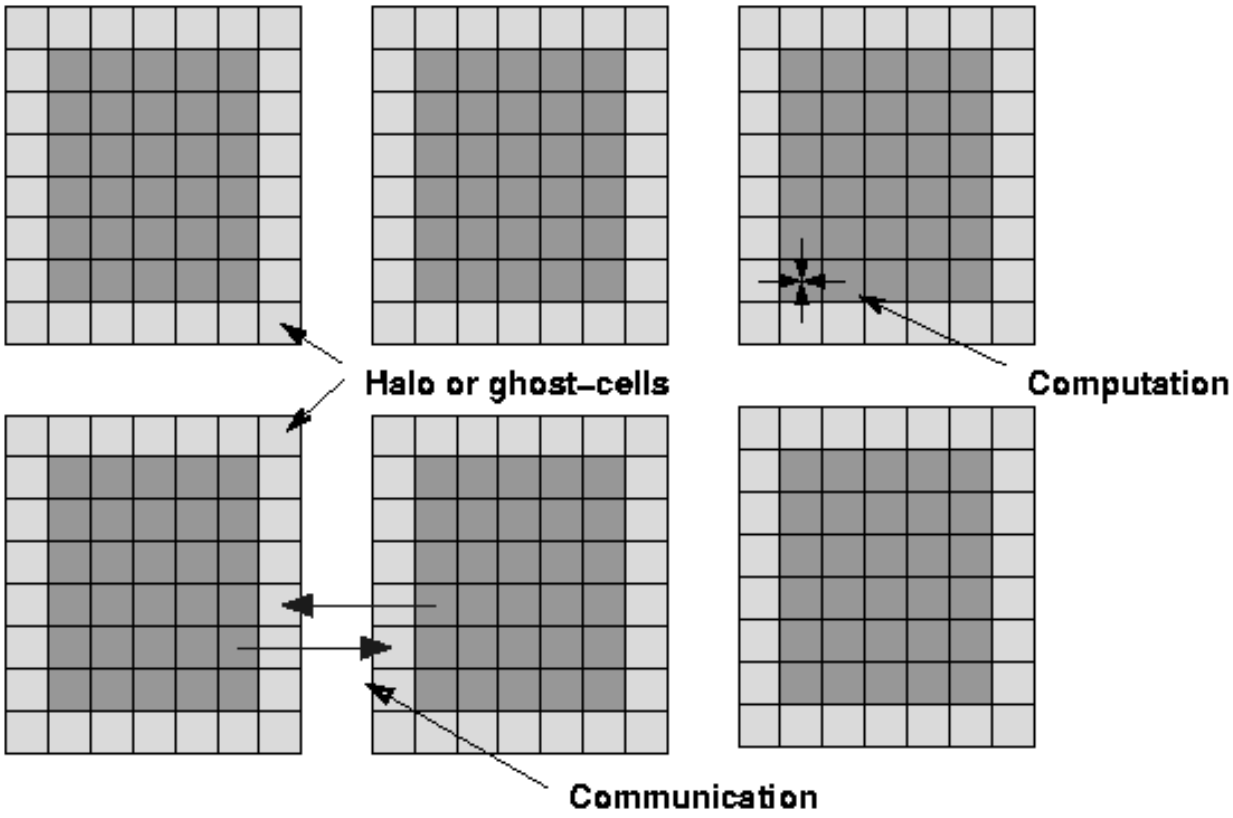

FIG. 3 – Cellules fantômes

2 Projet

2.1 Objectif

Le but du projet est de produire un certain nombre de simulations en utilisant les relations discrètes (**??**-15), qui réalisent une approximation des équations de Saint-Venant (1-3). De plus l'objectif est de sauver dans un fichier le résultat du calcul (!), à intervalles réguliers $P'\tau$, i.e. on sauve les champs $(u^n_{kl}, v^n_{kl}, p^n_{kl})$ pour $n = 0, P', 2P', \ldots, P$. Une simulation est caractérisée par :

- le paramètre de Coriolis f,
- la résolution δ, la taille du domaine ($M\delta$, $N\delta$),
- la condition initiale $(u, v, p)(t = 0)$,
- le pas de temps τ, la durée $P\tau$ de la simulation et l'intervalle $P'\tau$ entre deux sauvegardes.

Un certain nombre de simulations sont définies plus loin. Le travail se basera sur un code existant en langage C. L'algorithme a la forme générale :

- initialiser $M, N, P, P',\ f, \delta, \tau$
- allouer les tableaux nécessaires (de taille au moins $M \times N$) : (u, v, p) doit être alloué en deux exemplaires $(u^{n-1}_{kl}, v^{n-1}_{kl}, p^{n-1}_{kl})$ et $(u^n_{kl}, v^n_{kl}, p^n_{kl})$, les autres champs en un seul exemplaire
- initialiser $(u^0_{kl}, v^0_{kl}, p^0_{kl})$, périodiser selon (13) et sauver
- calculer $(\dot{u}^0_{kl}, \dot{v}^0_{kl}, \dot{p}^0_{kl})$ selon (4-13)
- calculer $(u^1_{kl}, v^1_{kl}, p^1_{kl})$ selon (15)
- éxécuter la boucle temporelle calculant par récurrence $(u^n_{kl}, v^n_{kl}, p^n_{kl})$ selon (14) pour $n = 2 \ldots P$ en sauvant lorsque n est un multiple de P'.
- libérer les tableaux alloués

2.2 Travail demandé

Il se fera en principe en environnement Linux. Le travail de développement se fera en CUDA, une extension de C ajoutant quelques mots-clés permettant d'indiquer quelles routines doivent s'éxécuter

sur le GPU et dans quelle type de mémoire résident les variables/tableaux manipulés. Des tests seront réalisés pour que le calcul sur GPU donne les mêmes résultats que l'implémentation CPU. En cas de problème il pourra être utile de sauver des résultats de calculs intermédiaires (champs $H, \zeta, \eta U, V$) afin d'en déterminer l'origine. Il s'agira ensuite d'analyser/améliorer la performance à la fois en termes de calcul (Gflops) et de débit mémoire (Go/s), ce dernier étant généralement le facteur limitant pour ce type de calcul. Par conséquent il faudra probablement dans un deuxième temps regrouper plusieurs opérations afin d'améliorer le ratio calcul/transfert et s'approcher de la performance théorique du matériel (800 GFlops selon le constructeur).

2.3 Simulations

Des simulations de référence seront définies au début du projet (séance 5)