COMMISSARIAT A L'ENERGIE ATOMIQUE
CENTRE D'ETUDES DE LIMEIL-VALENTON
Département de MATHEMATIQUES APPLIQUEES
Service Mathématiques et Codes Numériques
94195 VILLENEUVE SAINT GEORGES Cedex

CEA - N - 2724

IMPLEMENTATION SUR IPSC 860 D'UNE METHODE DE MONTE CARLO POUR LA RESOLUTION DE L'EQUATION DE BOLTZMANN

***

ALOUGES F

IMPLEMENTATION SUR IPSC-860 D'UNE METHODE DE MONTE-CARLO POUR LA RESOLUTION DE L'EQUATION DE BOLTZMANN

François Alouges

Résumé

Nous nous intéressons à l'implantation sur une machine massivement parallèle (l'IPSC-860) d'une méthode de Monte-Carlo en vue de résoudre l'équation de Boltzmann. Le parallélisme de cette machine incite à envisager une approche multi-domaines et pose entre autres le problème d'une génération automatique de maillages locaux à partir d'un maillage global 3-D non-structuré.

1 Introduction

En 1872 Boltzmann propose une équation pour modéliser les écoulements gazeux raréfiés. A faible densité (par exemple dans la haute atmosphère), les gaz ne peuvent plus être assimilés à des fluides et une approche statistique devient indispensable. Cette équation a pour inconnue la densité de gaz et dépend dans le modèle le plus simplifié que nous utilisons, de sept variables : la position (3 variables), la vitesse des particules de gaz (3 variables) et le temps. Evidemment un tel espace de paramètres pose des problèmes de discrétisation lorsque l'on cherche à résoudre cette équation d'un point de vue numérique. C'est sans doute pour cette raison que les méthodes de type Monte-Carlo furent d'abord employées (les méthodes directes dites déterministes commencent tout juste à être utilisées, la puissance des ordinateurs actuels permettant enfin d'étudier des cas non triviaux et physiquement plausibles) . Dans ce rapport, nous discutons une implantation d'une méthode de Monte-Carlo, sur une machine massivement parallèle. La première partie présente l'équation de Boltzmann et ses principales propriétés. La deuxième traite des méthodes Monte-Carlo et en particulier de celle qui a été utilisée. Ensuite nous proposons l'algorithme et sa parallélisation dans la troisième partie. Divers problèmes liés à une parallélisation efficace y sont également soulevés. Les résultats sont présentés dans la quatrième partie tandis qu'en annexe figurent une présentation de la machine ainsi que les procédures de communication qui permettent de l'utiliser.

2 L'Equation de Boltzmann

Cette équation est constituée de deux termes. Notons $f = f(t, x, v)$ la densité de probabilité de présence au temps t des molécules de gaz possédant la vitesse v. En

l'absence de toute interaction, les molécules de gaz se déplacent en ligne droite de sorte que f est solution de

$$\frac{\partial f}{\partial t} + v \cdot \frac{\partial f}{\partial x} = 0 \tag{1}$$

Dans un gaz dilué, les forces agissant sur les particules se réduisent à des chocs élastiques entre les molécules du gaz. Si l'on considère que toutes les molécules de gaz possèdent la même masse, la conservation de l'énergie cinétique et de la quantité de mouvement lors d'un choc élastique impose des relations sur les vitesses des particules après le choc. Appelons v, v_* les vitesses des molécules avant la collision et v', v'_* leurs vitesses respectives après. On doit avoir

$$\begin{aligned} v' + v'_* &= v + v_* \\ |\, v' \,|^2 + |\, v'_* \,|^2 &= |\, v \,|^2 + |\, v_* \,|^2 \end{aligned} \tag{2}$$

On montre facilement que v, v_* et v', v'_* sont deux diamètres d'une même sphère, et que l'on peut écrire le système 2 sous la forme

$$\begin{aligned} v' &= \tfrac{1}{2}(v + v_*) + \tfrac{\omega}{2} \,|\, v - v_* \,| \\ v'_* &= \tfrac{1}{2}(v + v_*) - \tfrac{\omega}{2} \,|\, v - v_* \,| \end{aligned} \tag{3}$$

pour une direction ω de S^2.

L. Boltzmann propose alors de modifier l'équation 1 de la façon suivante

$$\frac{\partial f}{\partial t} + v \cdot \frac{\partial f}{\partial x} = Q(f, f) \tag{4}$$

où $Q(f, f)$ est l'opérateur de collision

$$Q(f, f)(t, x, v) = \int_{\mathbf{R}^3} \int_{S^2} q(v - v_*, \omega)(f' f'_* - f f_*) dv_* d\omega, \tag{5}$$

$$f = f(t, x, v), f_* = f(t, x, v_*), f' = f(t, x, v'), f'_* = f(t, x, v'_*). \tag{6}$$

La quantité $q(v, \omega)$ qui apparaît dans (5) est appelée section efficace et nous supposerons qu'elle est de la forme

$$q(v, \omega) = |\, v \,|^\alpha \tag{7}$$

pour un certain paramètre α qui dépend du gaz considéré. D'autres modèles plus complexes tiennent aussi compte du paramètre angulaire ω de façon à favoriser certaines collisions par rapport à d'autres moins probables (collisions rasantes).

On peut montrer que l'équation 4 conserve tous les invariants physiques (masse, quantité de mouvement et énergie).

Enfin, en utilisant les invariants de collision, on peut prouver que la quantité

$$H = \int_{\mathbf{R}^3} \int_{\mathbf{R}^3} f Log(f) dx dv \tag{8}$$

décroît au cours du temps. Ceci permit à Boltzmann de démontrer que les solutions de (4) tendent vers des Maxwelliennes en v, du moins lorsque les problèmes de conditions aux limites ne se posent pas.

D'un point de vue théorique, l'existence de solutions de (4) a fait l'objet de nombreux travaux. La plus grande avancée en la matière a sans doute été faite par Di Perna et Lions (voir [8]) qui montrent un résultat d'existence de solutions dans un cadre très général (voir aussi [7]).

3 Principes des méthodes numériques

3.1 Méthodes directes - Méthodes de Monte-Carlo

Lors de la résolution du point de vue numérique de l'équation de Boltzmann[4] le principal problème est la grande dimension de l'espace des paramètres (3 en espace, et 3 en vitesse). Une discrétisation brutale de cet espace en vue de construire un schéma numérique classique est donc à proscrire. Toutefois, des méthodes directes sont apparues récemment dans des cas simplifiés (dimension 2x2 ou 1x3 en espace-vitesse). Avec l'arrivée de très gros ordinateurs, on peut raisonnablement penser que de telles méthodes seront bientôt implantées dans le cas physiquement concret 3D espace x 3D vitesse. Les autres méthodes utilisées pour la résolution du problème sont toutes probabilistes. Ce sont des méthodes de Monte-Carlo basées sur les deux idées suivantes:

- Approximation de f au moyen de particules;
- Découplage transport collision.

Lors de la discrétisation de f en particules, on approche f par une expression de la forme

$$f = \sum_{i}^{N} \alpha_i(t)\delta_{x_i}\delta_{v_i}. \tag{9}$$

Ceci revient à considérer que l'on a dans le domaine d'étude N particules de gaz de poids α_i, positionnées en x_i et possédant la vitesse v_i. La méthode numérique consiste à faire évoluer au cours du temps ces particules. Les diverses informations (densité, quantité de mouvement, énergie, et température) seront obtenues par des statistiques sur ces particules. Le découplage transport-collision se fait de la façon suivante:

- On résout d'abord l'équation de transport au cours d'un pas de temps, c'est-à-dire que l'on avance les particules à partir de x_i à la vitesse v_i pendant un pas de temps Δt. Cette étape revient à résoudre

$$\frac{\delta f}{\delta t} + (v \cdot \vec{\nabla})f = 0. \tag{10}$$

- Puis, on effectue les collisions entre les particules. Du point de vue mathématique, on résout

$$\frac{\delta f}{\delta t} = Q(f, f). \tag{11}$$

  Diverses formulations existent correspondant à plusieurs modélisations différentes. Par la suite nous ne détaillerons que la méthode de Monte-Carlo symétrique que nous avons implémentée.

3.2 La Méthode de Monte-Carlo Symétrique

Dans cette méthode, on effectue les collisions entre les particules de façon symétrique afin de conserver les principaux invariants (masse, énergie et quantité de mouvement). On procède en deux étapes. Tout d'abord, on regroupe les particules par maille dans le domaine en faisant la convention que les particules qui sont physiquement dans une même maille peuvent interagir. La collision se passe alors ainsi: Si deux particules i et j forment une paire de collision, on tire aléatoirement

uniformément sur la sphère de diamètre (v_i, v_j) un autre diamètre (v_i', v_j'). v_i' et v_j' sont les vitesses des particules i et j après la collision.

La principale restriction concerne le poids des particules. En effet, afin d'avoir conservation de la masse lors de l'étape de collision, on est obligé d'imposer aux particules d'avoir la même masse à l'intérieur d'une même maille. Dans le travail présenté ici, toutes les particules ont la même masse et ne représentent qu'une seule espèce de gaz. Toutefois, il peut s'avèrer utile de pouvoir changer la masse des particules dans certains sous-domaines du domaine étudié (par exemple, si l'on étudie la rentrée d'un corps dans l'atmosphère, la très grosse différence de densité entre l'avant et l'arrière de l'engin pose des problèmes de représentativité avec des particules de même masse. On associe alors des poids différents aux particules suivant l'endroit où elles se trouvent. Apparaissent alors des difficultés lorsque des particules passent d'un domaine "lourd" à un domaine "léger" et vice-versa).

Le modèle retenu est celui appelé VHS ("Variable Cross-section Hard Spheres"). Il se situe entre le modèle HS ("Hard Sphere") où les particules sont assimilées à des sphères de diamètre constant et le modèle PLR ("Power Law Repulsion") plus réaliste où les molécules se repoussent avec une force qui dépend de la distance qui les sépare. Le modèle VHS introduit par Bird [6] assimile les particules à des sphères rigides mais adapte leur diamètre (ou la section efficace de collision) au cours des chocs. L'énorme avantage de ce modèle est qu'il cumule le calcul des collisions du modèle HS (qui est très simple) avec la loi de viscosité du modèle PLR qui est très réaliste.

3.3 Les Conditions de paroi

Certaines faces du domaine sont considérées comme étant des faces de l'objet et l'on doit définir le comportement des particules lorsque celles-ci heurtent la paroi. Ces conditions de paroi sont de deux types (voir [1] [2]):

- Soit des conditions de réflexion spéculaire: La composante de la vitesse normale à la paroi change de signe. Hormis cela, il n'y a aucune interaction entre la particule et la paroi;
- Soit la réflexion se fait avec accomodation parfaite, et la particule est ré-émise aléatoirement dans le domaine avec une vitesse tirée au hasard selon une loi Maxwellienne dépendant de la température de la paroi.

Le choix entre l'une ou l'autre de ces conditions se fait aléatoirement en privilégiant si on le désire une condition sur l'autre.

3.4 L'injection des Particules

Par les faces qui communiquent avec l'extérieur, on injecte les particules. Il faut simplement préciser les quantités physiques dont dépend cette injection à savoir:

- Les trois composantes de la vitesse moyenne des particules qui entrent dans le domaine.
- La température du flux de particules.

Le programme doit ensuite pouvoir tirer une vitesse aléatoire (suivant une loi Maxwellienne centrée autour de la vitesse moyenne et dépendant de la température donnée) pour chacune des particules qui entrent dans le domaine.

4 L'Algorithme Utilisé

Les impératifs que l'on s'était fixés étaient multiples.

- Traiter des cas 3D-espace et 3D-vitesse sur des maillages qui ne sont pas structurés;
- Pouvoir prendre le maximum de particules;
- Utiliser la machine massivement parallèle de l'ONERA;
- Tâcher de minimiser les communications.

En vue d'une parallélisation efficace sur l'iPSC-860 de l'ONERA, nous avons opté pour une approche multi-domaines. Le domaine global est découpé en sous-domaines et l'on attribue un sous-domaine à chacun des processeurs. De cette façon, la phase de collision est entièrement parallèle puisque chaque processeur regroupe les particules dans chaque maille de son sous-domaine, puis effectue un certain nombre de collisions entre elles. Si les domaines contiennent un nombre similaire de particules, on peut espérer obtenir ainsi un parallélisme presque optimal. En revanche, la phase de transport nécessite des communications entre les processeurs. En effet, si lors d'un pas de temps, une particule passe du domaine i au domaine j par exemple, le processeur i doit alors l'envoyer au processeur j et celui-ci sera chargé de terminer d'avancer la particule pendant le reste du pas de temps (et éventuellement transmettre de nouveau la particule à un processeur k). Ce système permet de n'avoir d'échanges qu'entre des processeurs qui correspondent à des domaines physiquement voisins. L'inconvénient est qu'il découpe le pas de temps en plusieurs morceaux ce qui peut ralentir l'algorithme.

De plus afin de gagner de l'espace mémoire, nous ne voulions pas répéter tout le maillage sur chaque processeur. Ainsi, chaque processeur ne devait connaître que la géométrie de son sous-domaine propre et des informations sur les mailles voisines à son sous-domaine. Nous avons créé un générateur de sous-domaines qui répond à toutes ces exigences.

4.1 Le générateur de sous-domaines

Le générateur de sous-domaines reçoit en entrée un fichier contenant le maillage tétraèdrique global sous le format suivant:

```
Nbre de points - Nbre de tétraèdres
Pour chaque point
Numéro du point - Coordonnés du point - Type du point
Pour chaque tétraèdre
Numéro du tétraèdre - Numéros des sommets dans la liste
précédente
```

Le type d'un point est un entier qui contient des informations sur le point (par exemple, si le point appartient à l'objet, alors le deuxième bit de poids faible de son type est 1).

Le générateur construit à partir de ce fichier et du nombre P de processeurs voulu (qui doit être une puissance de deux) P fichiers sur disques. Chacun de ces fichiers contient pour chaque processeur

- Ses processeurs voisins (avec lesquels il devra communiquer);
- Les noeuds de son domaine local (renumérotés à partir de 1);

- les faces de son domaine local;
- les mailles de son domaine local;
- Pour chaque maille m et chaque face f de m, le numéro de la maille voisine à m en traversant f. Ce numéro est donné dans la numérotation locale du processeur à qui appartient cette maille voisine. Le numéro de ce processeur est aussi donné. On précise de plus si la face f est une face de l'objet.
- Pour chaque face de l'objet dans le domaine local, on construit un trièdre orthonormé dont le premier vecteur est normal à la face et rentrant dans le domaine. Ce trièdre est utilisé lors des reflexions sur l'objet.

Comme nous le verrons ensuite, le processeur 0 sera chargé du domaine extérieur. On lui donne donc de plus les faces par lesquelles se fera l'injection des particules ainsi que leur surface.

Précisons que la distribution des mailles aux processeurs n'est pas du tout optimale. En effet, on répartit les M mailles du domaine aux P processeurs dans l'ordre:

Processeur 0 : Exterieur
Processeur 1 : Mailles 1 à $M/(P-1)$
Processeur 2 : Mailles $M/(P-1)+1$ à $2M/(P-1)$
$\vdots$
Processeur P-1 : Mailles $M - M/(P-1)$ à M

On ne sait pas en particulier si le découpage que l'on a fait, crée des sous-domaines connexes (il faudrait par exemple faire une renumérotation frontale du maillage pour cela). On ne sait pas non plus si les sous-domaines ont des volumes similaires, ni si des sous-domaines voisins correspondent à des processeurs voisins sur le réseau en hypercube de l'iPSC-860. Ces problèmes reliés au plongement d'un graphe sur un autre graphe de manière optimale commencent à apparaître avec l'arrivée des ordinateurs parallèles.

4.2 L'algorithme

L'algorithme que nous avons implanté assigne un rôle particulier au processeur 0. En effet, celui-ci devra non seulement gérer le domaine exterieur, mais aussi se charger de la synchronisation de tout l'algorithme. D'autre part, les autres processeurs qui se chargent du domaine interieur doivent se partager les deux phases (trajectographie et collisions) le plus équitablement possible. Nous allons donc détailler l'implémentation de ces deux phases en commençant par la partie "collisions" puisque celle-ci est naturellement parallèle.

COLLISIONS: Cette partie de l'algorithme est parallèle compte tenu du parallélisme choisi (découpage en sous-domaines). Chaque processeur regroupe donc ses particules par mailles dans son sous-domaine, puis effectue un certain nombre de collisions fictives entre celles-ci. Les résultats montrent qu'effectivement le temps nécessaire pour effectuer les collisions est inversement proportionnel au nombre de processeurs utilisés. Il faut noter que contrairement à ce que l'on pourrait penser, la phase de collisions n'est pas du tout la plus coûteuse du point de vue du temps de calcul. En effet elle ne nécessite aucune communication.

TRAJECTOGRAPHIE: Cette partie est beaucoup plus délicate à implanter. C'est ici que se feront toutes les communications, et elle se montrera comme étant la plus coûteuse en temps de calcul. Les particules sont repérées par mailles. Lorsque la particule sort de la maille au cours du pas de temps, on calcule ce point de sortie, la nouvelle maille dans laquelle elle est entrée et le temps qu'il lui reste à avancer. Puis on recommence avec la nouvelle maille. Ce procédé qui pourrait s'avèrer peu efficace ne consomme en fait pas tellement de temps puisque les particules ne franchissent que très peu de mailles au cours d'une itération.

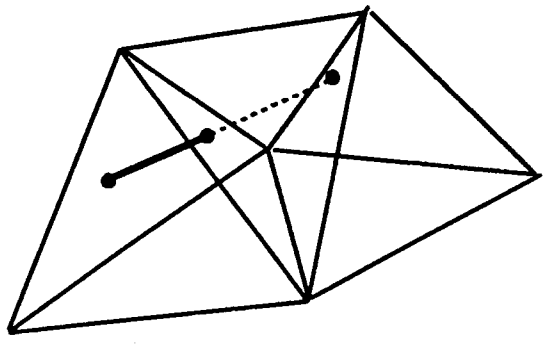


Figure 1: La particule sort de la maille.

L'avantage de cette procédure est que lors d'une rencontre avec l'objet, on calcule le point où la particule frappe l'objet. Il est alors aisé de faire une réflexion sur cette face (spéculaire ou avec accomodation parfaite) puisque l'on connaît un trièdre ortho-normé associé à cette face.

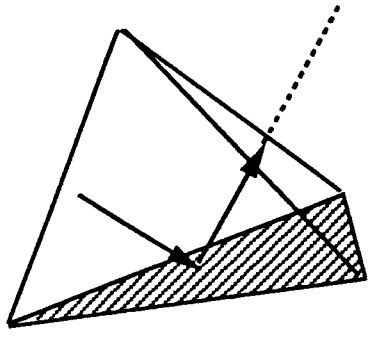


Figure 2: Réflexion spéculaire sur l'objet.

La principale difficulté apparaît lorsque des particules changent de sous-domaine au cours d'un pas de temps. Il faut alors la transmettre au processeur chargé du domaine voisin. Néanmoins dans un souci d'efficacité lors de cette communication entre les processeurs, on regroupe les particules qui changent de domaine dans un tableau et l'on envoie tout le tableau en une seule fois. Le processeur responsable du sous-domaine voisin finira donc le pas de temps avec les particules qui viennent de lui parvenir. Toutefois, certaines particules peuvent éventuellement traverser plusieurs (plus de deux) sous-domaines au cours d'un même pas de temps. La phase de trajectographie ne doit s'arrêter que lorsque toutes les particules ont été avancées à leur vitesse pendant un pas de temps Δt. La synchronisation de tout

ce processus est effectuée par le processeur 0. L'algorithme s'écrit alors de la façon suivante:

Processeur 0

- (1) Lecture de la géométrie (faces d'injection)
- (2) Injection des particules
- (3) Réception des particules qui sortent du domaine
- (4) Réception de l'état de chacun des processeurs (compte ceux qui sont arrêtés)
- (5) Si un processeur n'est pas arrêté envoyer CONT et aller en (3) sinon envoyer STOP et aller en (6)
- (6) Envoie un message pour effectuer des sondages et retourner à (2) pour le pas de temps suivant

Pour les processeurs de 1 à N

- (1) Lecture de la géométrie locale
- (2) Trajectographie - on fait avancer les particules pendant un pas de temps si possible et on stocke celles qui sortent du sous-domaine dans des tableaux que l'on transmettra au processeur voisin
- (3) Envoi des particules qui sortent du sous-domaine
- (4) Réception des particules qui entrent dans le sous-domaine
- (5) Envoi au proc. 0 d'un message s'il n'y a plus de particule à traiter. Le processeur est alors arrêté
- (6) Réception d'une commande du processeur 0. Si CONT aller en (2) sinon la commande est STOP, aller en (7)
- (7) Effectuer les collisions et attendre une commande du proc. 0 pour faire un sondage ou pour recommencer un nouveau pas de temps. Dans ce cas retourner à (2)

4.3 La synchronisation

Comme nous venons de le voir, la synchronisation de tout l'algorithme est réalisée entièrement par le processeur 0. Elle se fait par envois de messages aux autres processeurs (CONT, STOP, SOND, etc.) afin de leur dire s'il faut recommencer une étape de trajectographie, faire les collisions, effectuer un sondage ou bien refaire un pas de temps supplémentaire. L'inconvénient mineur d'avoir des communications entre le processeur 0 et tous les autres processeurs au cours d'un pas de temps est largemement pallié par le fait que ces communications sont effectuées avec des messages courts (4 octets) et qu'ainsi il n'y a pas d'attente d'établissement d'un lien de communication sur le réseau de l'iPSC-860. Puisque cette synchronisation est très rapide, le processeur 0 gère aussi l'extérieur du domaine (injection de particules dans le domaine et réception des particules qui sortent du domaine). La seule petite difficulté lors de l'implantation de ces communications réside dans le fait qu'il faut donner deux types de message (voir Annexe) différents pour deux messages différents. Ainsi des communications sont faites

- entre le processeur 0 et les autres pour la synchronisation
- entre le processeur 0 et les autres pour l'injection de particules

- entre deux processeurs voisins pour la transmission de particules

De plus lorsque l'on transmet des particules d'un sous-domaine à un autre, on doit transmettre la position, la vitesse, la maille dans laquelle est la particule ainsi que le temps qui lui reste à effectuer en trajectographie. A chaque fois les types des messages doivent être différents et doivent être connus des processeurs émetteur et recepteur du message.

Signalons, aussi que dans l'algorithme présenté précédemment, tous les processeurs effectuent une remontée si au moins l'un d'eux n'a pas terminé son étape de trajectographie. Ceci peut sembler à priori très coûteux, mais s'avère en fait relativement négligeable comme nous le verrons dans la partie qui suit.

5 Résultats

Dans cette partie, nous présentons les différents résultats obtenus avec le code que nous avons écrit pour l'iPSC-860. Dans un premier temps, nous présentons le cas-test étudié, puis nous discutons les résultats numériques et les comparons avec ceux obtenus par Lengrand (voir [2]). Par la suite, nous donnons les temps de calcul obtenus dans plusieurs configurations (nombre de processeurs, nombre de particules, etc.)

5.1 Le Cas-test Etudié: La rampe de Compression

Le cas-test que nous avons étudié est celui de la rampe de compression. En fait cet exemple a fait l'objet d'un travail effectué par J.C. Lengrand (voir [2]). Le maillage que l'on voit figure 3 en est la trace sur un plan vertical, le maillage complet en trois dimensions n'a pas été dessiné dans un souci de clarté. L'objet considéré est constitué de deux plaques, l'une est horizontale, l'autre est inclinée de $\alpha = 10°$ par rapport à l'horizontale. On étudie en quelque sorte l'écoulement autour d'un aileron.

La plaque horizontale se situe à $y = 0cm$ entre $x = 0cm$ et $x = 5cm$, et la plaque inclinée est dans son prolongement. Il faut préciser que le cas étudié par J.-C. Lengrand est le même cas mais en dimension deux d'espace seulement. Ce cas test simule par exemple l'écoulement autour d'un aileron incliné. Le maillage 3D utilisé pour ce cas test est constitué de 1408 points et de 3870 mailles tétraèdriques.

5.2 Résultats Numériques

Dans cette section nous discutons deux cas test que nous avons utilisés. Ces deux cas sont :

- de l'air à mach 4;
- de l'azote à mach 20.

Ces deux cas test figurent dans le rapport de J.C.Lengrand (voir [2]). Dans le premier cas test, les paramètres physiques et informatiques utilisés pour l'algorithme sont donnés figure (4).

Une coupe de la densité est donnée. On remarque que l'onde de choc en amont est très bien représentée ainsi que la raréfaction du gaz au bout de la rampe.

Le deuxième cas test concerne de l'azote injecté à mach 20. De la même façon, les données utilisées sont présentées figure (5).

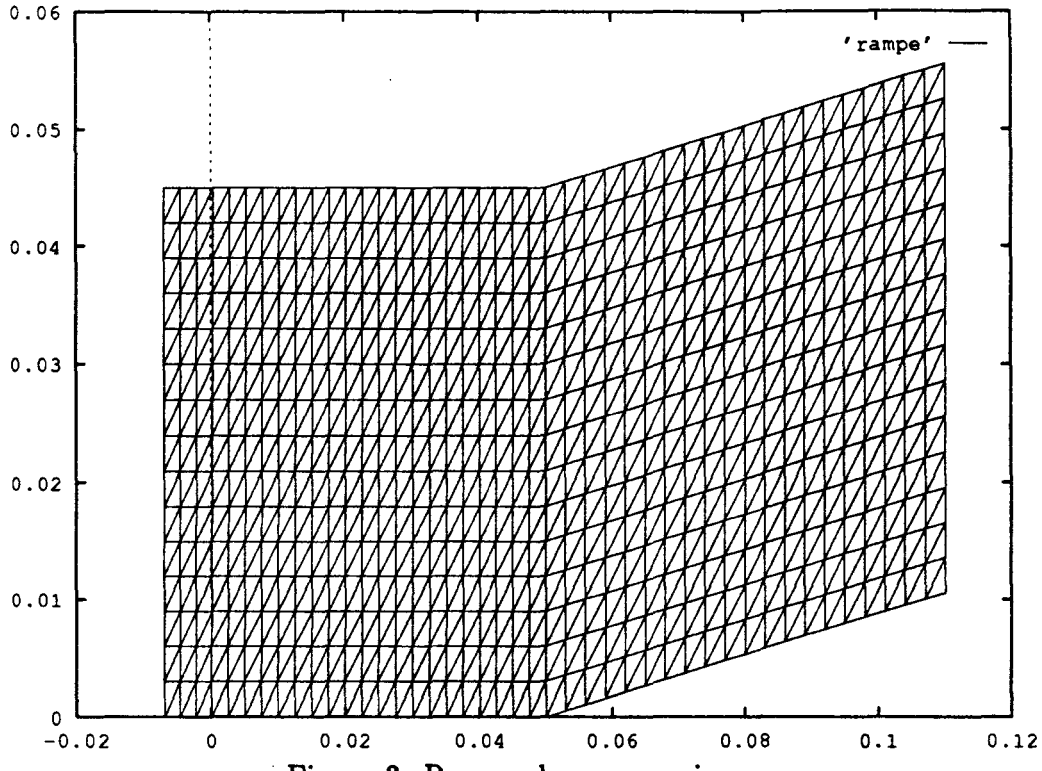


Figure 3: Rampe de compression.

On a représenté aussi les isovaleurs de la densité. On retrouve aussi des phénomènes présentés aussi dans [2]. En particulier la raréfaction du gaz au voisinage de la plaque plane, près de l'angle de la rampe. Le comportement général est à peu près le même que celui présenté par J.C. Lengrand.

On remarque tout de même que les résultats sont très bruités. Deux explications peuvent être données. Tout d'abord nous avons pris assez peu de particules pour des cas 3D (rappelons que J.C. Lengrand prend environ autant de particules pour son calcul en 2D). De plus le maillage que nous avons utilisé n'est pas du tout raffiné au niveau de la plaque, et l'on perd ainsi beaucoup de précision pour les valeurs au voisinage de la rampe.

Pas de temps	$10^{-6}s$
Nombre de pas de temps	1000
Gaz utilisé	AIR
Représentativité	10^{14}
Coefficient d'accomodation	0.8
Température de l'objet	$260K$
Masse des molécules	4.81510^{-26}
Diamètre de référence	$4.31910^{-10}m$
Nombre de particules à la fin du calcul	180000
Vitesse d'injection	$669.3m/s$

Figure 4: Air Mach 4

Pas de temps	$10^{-6}s$
Nombre de pas de temps	1000
Gaz utilisé	AZOTE
Représentativité	5^{12}
Coefficient d'accomodation	0.8
Température de l'objet	$290K$
Masse des molécules	4.65110^{-26}
Diamètre de référence	$3.91210^{-10}m$
Nombre de particules à la fin du calcul	580000
Vitesse d'injection	$1503m/s$

Figure 5: Azote Mach 20

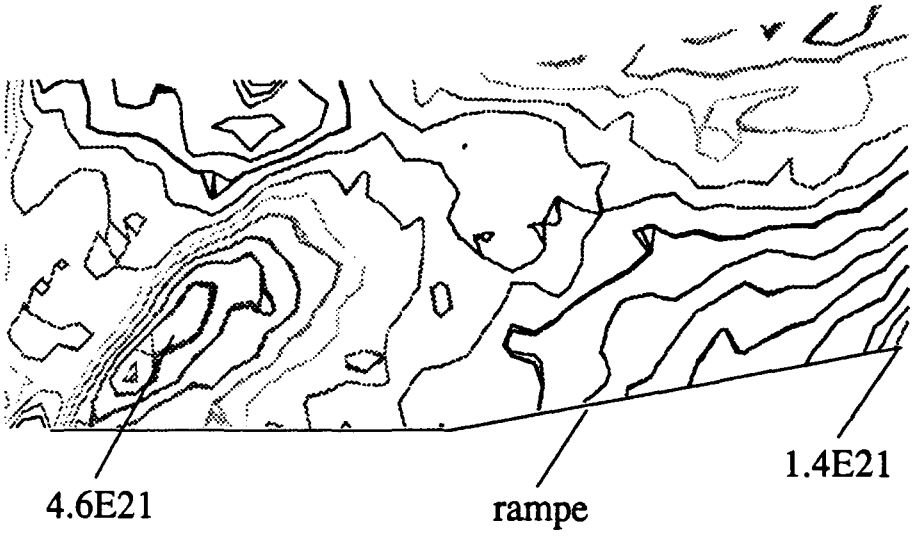
300 itérations					
Nombre de processeurs		4	8	16	32
sous-it. 1	Transport	0.1542 s	0.1194 s	0.0937 s	0.0825 s
	Communications	0.3317 s	0.3733 s	0.3304 s	0.3561 s
sous-it. 2	Transport	0.0394 s	0.0197 s	0.0096 s	0.0071 s
	Communications	0.1196 s	0.0948 s	0.0440 s	0.0354 s
sous-it. 3	Transport		0.0194 s		0.0049 s
	Communications		0.0873 s		0.0270 s
sous-it. 4	Transport				0.0047 s
	Communications				0.0255 s
Collisions		1.016 s	0.4244 s	0.1821 s	0.0933 s
Temps moyen d'une it.		1.661 s	1.138 s	0.660 s	0.636
Temps par it. par part. (μs)		13.5	9.2	5.3	5
Temps total d'execution		419 s	316 s	182 s	176 s
Nombre final de particules		123579	124089	124104	124143

Figure 6: Résultats avec 124000 particules

5.3 Discussion des Temps de Calcul

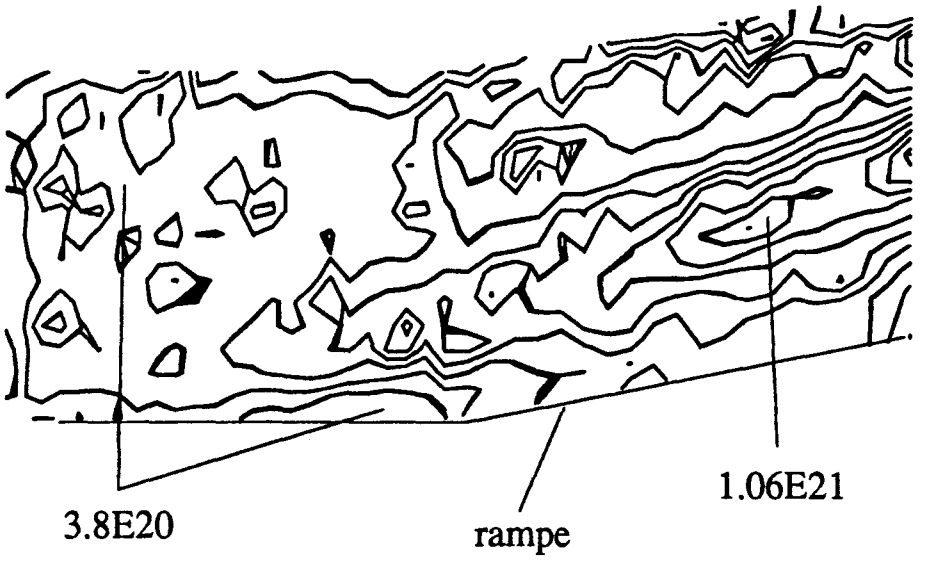
Afin de valider notre algorithme et de mesurer le taux de parallélisme atteint, nous donnons quelques mesures de performances dans des configurations de 4, 8, 16, et 32 processeurs. Les mesures présentées dans le tableau suivant montrent les résultats sur 300 itérations. Les mesures de temps effectuées sont des moyennes sur les cinq dernières itérations et correspondent à chaque fois sensiblement au même nombre de particules. Dans les tableaux apparaissent les temps de chaque sous-itération. Une sous-itération correspond à une étape de transport plus une étape de communication. Lorsque tous les processeurs remontent l'algorithme (c'est-à-dire que la trajectographie n'est pas terminée), on commence la sous-itération suivante.

Plusieurs remarques peuvent être faites. Tout d'abord la phase de collision est comme on le pensait très parallèle. Le temps d'exécution est directement relié au nombre de processeurs qui effectuent le calcul (3, 7, 15, et 31). Il ne faut pas oublier que le processeur 0 ne rentre pas en compte dans le calcul des collisions, ce qui explique qu'en passant de 4 à 8 processeurs, la vitesse soit plus que doublée. Ensuite, on remarque que dans tous les cas, toutes les sous-itérations sont négligeables devant la première. Ceci signifie qu'il n'est pas trop pénalisant de faire remonter l'algorithme à tous les processeurs lorsque l'un d'entre eux n'a



—— DENSITE AIR MACH 4

1° cas test



—— DENSITE AZOTE MACH 20

2° cas test

300 itérations		
Nombre de processeurs		32
sous-it. 1	Transport	0.4031 s
	Communications	1.3122 s
sous-it. 2	Transport	0.0276 s
	Communications	0.1343 s
sous-it. 3	Transport	0.0176 s
	Communications	0.1091 s
sous-it. 4	Transport	0.0171 s
	Communications	0.1071 s
Collisions		0.4014 s
Temps moyen d'une it.		2.5295
Temps par it. par part. (μs)		5
Temps total d'execution		696 s
Nombre final de particules		497630

Figure 7: Résultats avec 500000 particules

pas terminé sa trajectographie. On note aussi qu'il y a plus de remontées dans le cas de 32 processeurs (où les sous-domaines sont petits) que dans le cas de 4 processeurs (où les sous-domaines sont grands). Enfin, on voit facilement que le temps de la trajectographie est négligeable devant le temps de communication. Puisque les messages sont envoyés en parallèle, le temps de communication de la première sous-itération est assez grand. De plus les messages envoyés entre les processeurs sont grands (d'une longueur supérieure à 100 octets), ce qui sur cette machine est extrêmement pénalisant. Au cours des autres sous-itérations, les messages deviennent de plus en plus courts, et lorsque ceux-ci ont une longueur inférieure à 100 octets, le temps d'envoi du message est sensiblement plus court (dans le cas de tels messages, les processeurs n'attendent pas que le lien soit effectué pour envoyer leur message). Notons que dans le cas de 32 processeurs, chacun ne possède qu'un assez petit nombre de particules, ce qui explique que le temps de trajectographie soit négligeable devant celui de communication. De plus le débit des communications devrait être augmenté (multiplié par 100) sur la prochaine machine de la société INTEL, ce qui devrait permettre d'accélérer de manière significative le programme. En voyant les résultats du temps total d'exécution des 300 itérations, on peut être déçu de ce que le gain ne soit pas très grand. Rappelons qu'au début de l'exécution, le domaine ne contient aucune particule et ainsi celui-ci se "remplit" petit à petit. Pendant les premières itérations, seul un processeur travaille, puis deux, et ainsi de suite jusqu'à ce que tous les processeurs aient des particules dans leur domaine. A partir de ce moment là seulement, les tâches sont à peu près équilibrées.

Dans le tableau présenté figure 7, nous montrons à titre de comparaison les mêmes mesures qu'avant mais avec quatre fois plus de particules et 32 processeurs.

On remarque sans difficulté qu'à chaque fois le temps de calcul est proportionnel au nombre de particules. Le temps par itération et par particule reste de $5\mu s$ c'est-à-dire inchangé par rapport à l'exemple précédent. Précisons que ce temps est tout à fait comparable à celui obtenu sur un processeur du CRAY YMP (environ $4\mu s$ par particule et par pas de temps).

6 Conclusion

L'implémentation sur architecture distribuée d'une méthode de Monte-Carlo semble performante puisque l'on réalise des performances voisines de celles obtenues sur un processeur CRAY YMP avec un quart de la machine iPSC-860 d'INTEL. De plus on peut se permettre un très grand nombre de particules (jusqu'à 85000 par processeur, c'est-à-dire près 11 millions de particules sur la machine toute entière). Cependant, on peut déplorer le temps passé dans les communications (plus de la moitié du temps par itération). A l'avenir, ce problème devrait partiellement se résorber puisque les constructeurs (aussi bien CRAY que INTEL) annoncent des taux de communication très supèrieurs à celui de l'iPSC-860.

Remerciements

L'auteur a effectué ce travail en tant que scientifique du contingent au centre du CEA de Limeil-Valenton. Il tient tout d'abord à adresser ses plus vifs remerciements à Remi Sentis, Christophe Buet et Guy Ledanois pour leur disponibilité et leur patience. La programmation a été effectuée à l'ONERA, au sein de l'équipe dynamique dirigée par Pierre Leca. En particulier il est tout-à-fait normal qu'apparaissent dans ces lignes MM. François Rogier et Jacques Schneider pour leur aide constante.

A L'iPSC-860

L'iPSC-860 est la troisième génération des ordinateurs iPSC de la société INTEL. Il s'agit d'une machine parallèle dont chaque nœud est basé sur un microprocesseur INTEL i860 et qui communique avec l'exterieur par le biais d'un frontal. Le réseau de communications des nœuds est un hypercube de dimension 7, c'est-à-dire que la machine complète est constituée de 128 (2 puissance 7) nœuds de calcul. De plus l'iPSC-860 possède sa propre unité de stockage sur disques appelée **CFS** (pour *concurrent file system*). 8 nœuds de communications (à base de microprocesseurs INTEL 80386) permettent la sauvegarde et la relecture en parallèle de fichiers écrits sur le CFS.

Le frontal interface la machine avec l'exterieur. C'est lui qui charge les programmes sur les nœuds de calcul, et qui communique les résultats à l'utilisateur. Il est de plus multi-utilisateurs, c'est-à-dire que plusieurs utilisateurs peuvent allouer un sous cube de la machine de dimension inférieure à 7 et travailler en même temps. Par exemple, un utilisateur peut travailler avec les nœuds 0 à 31 (hypercube de dimension 5) tandis qu'un autre utilisateur peut travailler avec les nœuds 32 à 39 (hypercube de dimension 3), laissant les autres nœuds libres et donc utilisables. En pratique le frontal (aussi appelé **SRM**) est rapidement surchargé et il est inconcevable d'executer une partie du programme dessus.

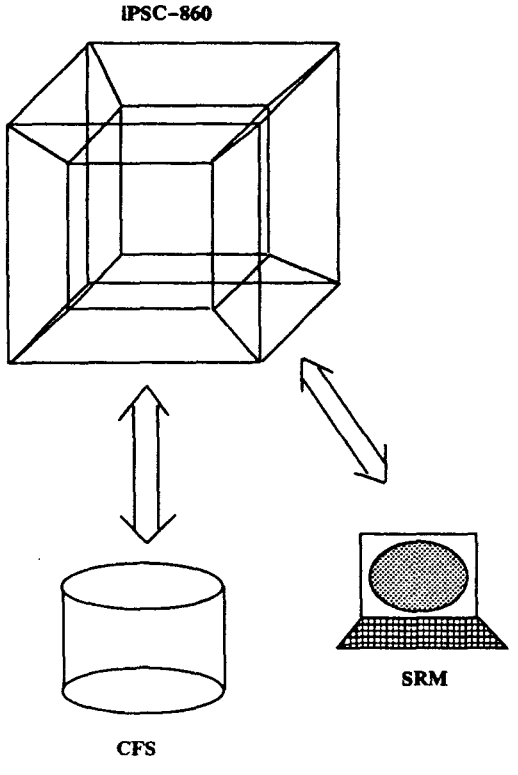


Figure 8: Schéma général de L'iPSC-860.

Le processeur i860 qui constitue donc le noyau de cette machine est un microprocesseur RISC qui intègre dans un même composant:

- Une unité de calcul sur des entiers;
- Un additionneur flottant;

- Un multiplieur flottant;
- Deux mémoires caches.

Puisque la fréquence d'horloge de ce processeur est de 40 MHz, une puissance maximale théorique de 80 MFlops (puisqu'une multiplication et une addition peuvent être effectuées simultanément pendant une période d'horloge) semble envisageable. Evidemment il faudrait pour cela alimenter les deux files d'attente du multiplieur et de l'additionneur à chaque période d'horloge. Pour cette raison, on ne peut raisonnablement espérer une puissance supérieure à 20 MFlops par processeur. Pour pouvoir faire communiquer les processeurs entre eux, un processeur de communication a été ajouté sur chaque nœud. Ce processeur n'est malheureusement pas très rapide puisque sa vitesse de communication maximale n'est que de 2 MegaOctets par seconde. Notons que dans la prochaine machine parallèle de la société INTEL, on annonce que cette vitesse de transfert serait multipliée par un facteur 100. Ceci pourrait rendre les communications sensiblement moins pénalisantes dans notre algorithme.

B Les Primitives de Communication

Sur ce genre de machine, le lien entre les processeurs est effectué par des envois de messages (puisque la mémoire est distribuée, on n'a jamais de vision globale de la mémoire). Fondamentalement, les envois de messages se font grâce à deux procédures: CSEND et CRECV. La structure d'appel de ces procédures est la suivante `Call Csend(Type, Tab, Len, Node, ProcId)` et `Call Crecv(Type, Tab, Len)` En ce qui concerne la procédure Csend, plusieurs paramètres apparaissent

- `Type` Le type du message. C'est un entier long qui sert à identifier le message. En pratique, il vaut mieux donner des types différents pour des messages différents. Dans le programme que nous avons écrit, par exemple, un type différent est donné lors de la transmission d'un processeur à un autre, des positions, vitesses, mailles, faces, temps de vie des particules envoyées.
- `Tab` Tableau à envoyer. Il s'agit ici de l'adresse du premier élément du tableau que l'on désire envoyer.
- `Len` Longueur du message. On spécifie dans cette variable la longueur en octets du message que l'on envoit d'un processeur à un autre. Le message est ainsi constitué de Len octets à partir de l'adresse donnée par Tab.
- `Node` Nœud d'arrivée. Pour spécifier le processeur cible.
- `ProcId` Identificateur de process. Sur la version précédente de la machine de la société INTEL (l'iPSC-2), plusieurs programmes pouvaient tourner sur chacun des nœuds. Cette variable servait à spécifier le programme cible pour l'envoi du message. Sur l'iPSC-860, un seul programme peut tourner sur un seul processeur et cette variable vaut toujours 0.

Evidemment, lorsqu'un message est envoyé du processeur i vers le processeur j, celui-ci doit recevoir le message grâce à l'instruction CRECV. La description des paramètres est la suivante:

- `Type` Type du message. Spécifie le type du message que l'on désire recevoir.
- `Tab` Tableau de reception. Cette variable précise l'endroit oú le message va être stocké.

- Len Longueur du message. Le processeur cible doit spécifier la longueur du message qu'il désire recevoir. Si le message reçu est de longueur supérieure à Len, une erreur est déclenchée. Dans le cas contraire aucune erreur n'apparaît.

Bibliographie

[1] LENGRAND (1988) Mise en œuvre de la Méthode de Monte-Carlo pour la simulation numérique directe d'un écoulement de gaz raréfié. Rapp. Lab. Aérothemique, Meudon CNRS.

[2] LENGRAND J.C., HEFFNER K.S., CHPOUN A. (Décembre 1990) Rampe de Compression en Gaz Raréfié, Rapport du GDR Hypersonique.

[3] NANBU K. (1983) Interrelations between various direct simulation methods for solving the Boltzmann equation, Journal of the Physical Society of Japan, vol. 52, no. 10, pp 3382-3388.

[4] NANBU K. (1983) Stochastic solution method of the master equation and the model Boltzmann equation, Journal of the Physical Society of Japan, vol. 52, no. 8, pp 2654-2658.

[5] CERCIGNANI C. (1975) Theory and application of the Boltzmann equation, Scottish Academic Press.

[6] BIRD G.A. (1976) Molecular Gas Dynamics, Clavendon Press, Oxford.

[7] GERARD P. (Juin 1988) Solutions Globales du Problème de Cauchy pour L'Equation de Boltzmann, Séminaire Bourbaki, 40ème année, n^o 699, pp 257-280.

[8] DI PERNA R.J., LIONS P.L. (1989) On The Cauchy Problem For Boltzmann Equations : Global Existence and Weak Stability, Annals of Mathematics, Vol 130, pp 321-366.

[9] DESVILETTES L. (1991) Une Méthode de Monte-Carlo Symétrique pour la Résolution de l'Equation de Boltzmann, rapport CEA

[10] SENTIS R., ANGRAND F., BUET C., DESVILETTES L., LEDANOIS G. (1991) Sur les Méthodes de Monte-Carlo pour la Résolution des Equations de Boltzmann, Rapport CEA