Examen de Modélisation et calcul scientifique

Jocelyne Erhel
INSA, Rennes

Février 2013

Durée 2 heures. Tous les documents sont autorisés. Les questions ne sont pas par ordre de difficulté et peuvent être traitées dans un ordre quelconque. Il est conseillé de lire tout l'énoncé avant de commencer.

1 Résolution d'un système linéaire

Un ingénieur informaticien doit résoudre un système linéaire. Il sait que la matrice est carrée, inversible et symétrique. Il choisit d'utiliser une factorisation de Cholesky. Le calcul s'arrête sur une erreur, avec un message du style: la racine carrée d'un nombre réel négatif n'est pas un nombre réel.

1. Expliquer pourquoi: quelle propriété la matrice doit-elle satisfaire pour que la factorisation de Cholesky soit possible ?
2. Quel autre algorithme aurait pu utiliser l'ingénieur ?

2 Résolution d'un système triangulaire plein

On considère une matrice L triangulaire inférieure inversible (donc les termes diagonaux sont non nuls), avec N lignes et N colonnes. On veut résoudre le système $Lx = b$.

Dans un premier temps, on stocke la matrice dans un tableau, avec une structure pleine. Voici l'algorithme utilisé (version stockage plein):

* initialisation
$x = b$
* boucle
for $j = 1$ to N
$x(j) = x(j)/L(j,j)$
for $i = j + 1, N$
$x(i) = x(i) - L(i,j) * x(j)$
endfor
endfor

1. Montrer que la matrice L est lue par colonnes.
2. Indiquer (en le justifiant) le nombre d'opérations de l'algorithme et la quantité de données stockées.
3. Indiquer (en le justifiant) la catégorie BLAS de cet algorithme.

3 Résolution d'un système triangulaire creux

On considère toujours un système $Lx = b$ avec une matrice triangulaire inversible, avec N lignes et N colonnes. Cette fois, on stocke la matrice sous forme creuse. On introduit l'ensemble des indices de lignes i tels que le coefficient $L(i, j)$ de la colonne j soit non nul:

$$L_{*j} = \{i; j + 1 \leq i \leq N; L(i, j) \neq 0\}$$

On note $card(L_{*j})$ le nombre d'éléments dans cet ensemble, autrement dit, le nombre de coefficients non nuls et non diagonaux dans la colonne j.

On note $nz(L)$ le nombre total d'éléments non nuls dans L (y compris les termes diagonaux).

1. Montrer que
$$nz(L) = \sum_{j=1}^{N} card(L_{*j}) + N.$$

2. Modifier l'algorithme précédent (section 2), en remplaçant la boucle "for $i = j + 1, N$" par une boucle utilisant L_{*j}.

3. Indiquer (en le justifiant) le nombre d'opérations de cette nouvelle version de l'algorithme (version stockage creux).

4. On utilise un stockage compressé par colonnes (CSC), qui stocke les valeurs des coefficients non nuls et qui permet de repérer le début de chaque colonne.

   Décrire les tableaux utilisés et préciser la quantité de données stockées.

5. Caractériser L_{*j} avec ces tableaux et écrire l'algorithme avec ces tableaux.

6. On suppose que $nz(L) = kN$, avec k un entier assez petit.

   Comparer le nombre d'opérations et la quantité de données des deux versions de l'algorithme, avec stockage plein ou stockage creux.

4 Partition en sous-domaines

On considère un carré discrétisé par une grille de cellules carrées, avec $N = n^2$ cellules (n dans chaque dimension). On effectue un algorithme sur ce carré, dont le nombre d'opérations est N. On veut effectuer l'algorithme sur une machine parallèle avec P processeurs. Pour cela, on utilise une partition en sous-domaines et on cherche le découpage le plus efficace.

On suppose que $n = p_1 c_1 = p_2 c_2$. On découpe le carré en sous-domaines rectangulaires de même taille, avec c_1 cellules dans le sens horizontal et c_2 cellules dans le sens vertical. On obtient ainsi $p_1 p_2$ sous-domaines avec $c_1 c_2$ cellules chacun.

1. Faire un dessin avec $p_1 = 2$, $p_2 = 8$ et un autre dessin avec $p_1 = 4$, $p_2 = 4$.

2. On appelle interface d'un sous-domaine les cellules de ce sous-domaine qui ont un côté commun avec une cellule d'un sous-domaine voisin. On note L la longueur totale des interfaces, mesurée en nombre de cellules (une cellule de coin est comptée deux fois, si elle est adjacente à deux sous-domaines).

   Montrer que
   $$L = 2 * [(p_1 - 1) * n + (p_2 - 1) * n].$$

3. On suppose que $P = p_1 p_2$ et on attribue un sous-domaine à chaque processeur. Le nombre d'opérations dans chaque processeur est $c_1 c_2 = N/P$, de sorte que les calculs sont bien répartis. Les processeurs doivent communiquer et le nombre total de données échangées est proportionnel à L. On veut minimiser le nombre de données échangées donc L.

   Ecrire L uniquement en fonction de p_1.

4. Déterminer la valeur p de p_1 telle que L soit minimale.

5. En déduire que la partition optimale est une partition en carrés avec n/p cellules de chaque côté.