High Performance Computing

Projet final : Parallélisation de l'algorithme de Jacobi

Rémi Garde

Mars 2018

Table des matières

1 Méthode de Jacobi

1.1 Définition du problème

Soit $n \in \mathbb{N}$, $A = (a_{ij})_{(i,j)\in[\![1,n]\!]^2}$ une matrice carrée de taille n, $B = (b_i)_{i\in[\![1,n]\!]}$ un vecteur de taille n. Nous cherchons à résoudre le système d'équations linéaires

$$AX = B \tag{1}$$

pour $X = (x_i)_{i\in[\![1,n]\!]}$ vecteur de taille n.

1.2 Résolution

Pour résoudre ce problème, A est séparée en 2 matrices D et R, avec D matrice de la diagonale de A et R les éléments non diagonaux de A :

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \end{pmatrix} = \begin{pmatrix} a_{1,1} & 0 & \cdots & 0 \\ 0 & a_{2,2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{n,n} \end{pmatrix} + \begin{pmatrix} 0 & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & 0 & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & 0 \end{pmatrix}$$

Ceci est intéressant car le calcul de D^{-1} est trivial :

$$D^{-1} = \begin{pmatrix} \frac{1}{a_{1,1}} & 0 & \cdots & 0 \\ 0 & \frac{1}{a_{2,2}} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{1}{a_{n,n}} \end{pmatrix}$$

La formulation (1) devient donc équivalente à $(D + R)X = B$ ce qui donne :

$$X = D^{-1}(B - RX) \tag{2}$$

Nous pouvons ensuite poser la suite $X^{(k)}, n \in \mathbb{N}$ définie comme suit :

$$\begin{cases} X^{(0)} = 0 \\ X^{(k+1)} = D^{-1}(B - RX^{(k)}), \quad k \in \mathbb{N}^* \end{cases}$$

La formule de récurrence s'écrit pour chacun des éléments de X :

$$\forall k \in \mathbb{N}^*, \forall i \in [\![1,n]\!], x_i^{(k+1)} = \frac{1}{a_{ii}}(b_i - \sum_{i\neq j} a_{ij}x_j^{(k)}) \tag{3}$$

1.3 Convergence

Déterminons une condition telle que $\lim_{k\to\infty} X^{(k)} = X$). En posant $A' = -D^{-1}R$ et $B' = D^{-1}B$, (2) devient alors :

$$X = A'X + B'$$

et ainsi

$$X^{(k+1)} - X = A'X^{(k)} + B' - (A'X + B') = A'(X^{(k)} - X)$$

ce qui indique

$$\lim_{k\to\infty} \|X^{(k+1)} - X\| = 0 \Leftrightarrow \rho(A') < 1$$

On remarque que les coefficients de A' s'écrivent :

$$a'_{i,j} = \begin{cases} 0 & \text{si } i = j \\ \frac{-a_{ij}}{a_{ii}} & \text{si } i \neq j \end{cases}$$

Soit λ une valeur propre de A' et $Y = (y_i)$ un vecteur propre associé. $A'Y = \lambda Y$ donne donc pour tout i :

$$\begin{aligned}
|\lambda y_i| &= |\sum_j a'_{ij} y_j| \\
|\lambda y_i| &= |\sum_{j\neq i} \frac{-a_{ij}}{a_{ii}} y_j| \\
|\lambda y_i| &= |\frac{1}{a_{ii}} y_i| \cdot |\sum_{j\neq i} -a_{ij}| \\
|\lambda y_i| &\leq |\frac{1}{a_{ii}} y_i| \sum_{j\neq i} |a_{ij}|
\end{aligned}$$

Or, pour toute matrice $M = (m_{ij})$ à diagonale strictement dominante :

$$\forall i, |m_{ii}| > \sum_{j\neq i} |m_{ij}|$$

Ainsi, si A est diagonale strictement dominante, alors son rayon spectral est inférieur à 1, ce qui fait converger la méthode de Jacobi.

2 Implémentation

2.1 Structures de données

J'ai choisi de représenter mes matrices et vecteurs de façon très simple : des tableaux de `double` et des entiers présentant les dimensions du tableau. Les deux fichiers les implémentant, `matrix.c` et `vector.c` sont donc très similaires. Ils contiennent des fonctions très simples pour la création, destruction et l'affichage de leur données, ainsi qu'une fonction leur permettant la création depuis un fichier. Cette dernière ne va charger qu'une partie de la matrice ou du vecteur pour limiter l'utilisation de mémoire.

Cette structure de données n'est pas idéale pour les matrices, car dans les cas réels elles seront creuses, diminuant de beaucoup l'efficacité mémoire. Une amélioration possible serait de réécrire `matrix.c` en l'implémentant sous la forme de tableaux représentant les coordonnées des valeurs. On gagnera ainsi en taille mémoire, et l'on perdra en temps d'accès aux valeurs.

2.2 Parallélisation

On cherche donc à paralléliser (3). Pour cela, on va partager les n lignes entre les q processus. Cela donne les relations suivantes pour un processus n°p, en notant $d = \lfloor \frac{n}{p} \rfloor$ et $r = \text{mod}(n, p)$:

- Nombre de lignes : d si $p < r$, $d + 1$ sinon ;
- Sa première ligne est $dp + \min(r, p)$
- le processus contenant la ligne i est le n°$\lfloor \frac{iq}{n} \rfloor$

Ces relations sont valables pour une numérotation des lignes et des processus commençant tous deux à 0. Cela correspond aux variables `num_rows` et `first_row` dans le code.

Bien sûr, une telle séparation n'est a priori pas réalisable, le calcul de x_i dépendant des $x_j, j \neq i$. Pour y remédier, chaque processus va contenir deux vecteurs : `x_local` et `x_global`. Le premier correspond aux lignes spécifiques aux processus, le deuxième à la totalité du vecteur $X^{(k)}$. Lors de chaque itérations, les processus commencent par envoyer à chacun leurs valeurs dans leur `x_local`. Tous les processus remplissent alors `x_global`, ce qui leur permet de calculer la nouvelle itération de `x_local`. Il ne reste plus qu'à répéter cette opération jusqu'à ce qu'on soit satisfait de l'approximation de X obtenue.

2.3 Arrêt

Afin de déterminer l'erreur commise $\epsilon_k = \|X^{(k)} - X\|_\infty$, la première réaction est de calculer $AX^k - b = A(X^k - X)$. Cependant c'est là un nouveau calcul de pruduit matrice-vecteurs, que l'on cherche justemet à accélérer.
En remarquant qu'en fin d'itération, `x_local` correspond à $X^{(k+1)}$ et `x_global` à $X^{(k)}$, on va plutôt calculer : $\mu_k = \|X^{(k+1)} - X^{(k)}\|_\infty$. En effet :

$$\begin{aligned}
\mu_k &= \|A'X^{(k)} + B' - X^{(k)}\|_\infty \\
&= \|A'X^{(k)} + X - A'X - X^{(k)}\|_\infty \\
&= \|A'(X^{(k)} - X) - (X^{(k)} - X)\|_\infty \\
\mu_k &\geq \left| \|A'(X^{(k)} - X)\|_\infty - \epsilon_k \right|
\end{aligned}$$

Or, nous avions vu que pour la convergence, le rayon spectral de A' est nécessairement strictement inférieur à 1. Donc $\|A'(X^{(k)} - X)\|_\infty \leq \epsilon_k$ et enfin $\mu_k \geq \epsilon_k$
Ainsi, pour obtenir une valeur approchée de X à 10^{-6} près, il suffira d'itérer jusqu'à obtenir une différence entre `x_local` et `x_global` de moins de 10^{-6}.

En pratique, pour limiter le nombre de calculs, à chaque coefficient `x_local[i]` on associe un booléen `residues[i]` qui indique si $x_i^{(k)}$ doit encore itérer. Si ce n'est pas le cas, on ne fais plus le calcul associé : $x_i^{(k+1)} = x_i^{(k)}$. Pour tester la fin du programme dans sa totalité, chaque processus va vérifier si au moins une valeurs de `residues` est à `true`. Ensuite chaque processus va envoyer ce résultat au premier processus, qui va pouvoir déterminer si le la solution est suffisament bien approchée dans sa globalité. Il répondra donc à tous les processus un ordre de continuer ou de s'arréter suivant le cas, gérée par la variable `run`.

3 Résultats