High Performance Computing

Project : Jacobi Method in Parallel for Equation resolution

Plessia Stanislas

Mars 2018

Table des matières

1 Introduction

2 Methode de Jacobi

2.1 Problème

Soit $n \in \mathbb{N}$, $A = (a_{i,j})_{i,j \in [\![1,n]\!]^2}$ matrice carrée de taille n, $b = (b_i)_{i \in [\![1,n]\!]}$ vecteur de taille n.
On cherche alors le vecteur $x = (x_i)_{i \in [\![1,n]\!]}$ tel que :

$$Ax = b \tag{1}$$

2.2 Méthode de résolution

Afin de résoudre ce problème, on va utiliser une méthode itérative appellée Méthode de Jacobi.

Tout d'abord, on va séparer la matrice A en deux sous matrices D et R de la manière suivante :

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

Notons D la matrice diagonale, et R la matrice du reste. Comme la matrice D est diagonale (qu'on suppose à coefficients nons nuls dans notre problème), D est trivialement inversible d'inverse :

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

On peut donc réécrire la formule (1) de la manière qui suit :

$$(D + R)x = b \tag{2}$$

$$\Leftrightarrow \quad Dx = b - Rx \tag{3}$$

$$\Leftrightarrow \quad x = D^{-1}(b - Rx) \tag{4}$$

La Méthode de Jacobi est alors unt méthode itérative qui va chercher à trouver un point fixe à cet équation. On peut alors définir la suite $x^{(k)}, k \in \mathbb{N}$ telle que :

$$\begin{cases} x^{(0)} = \vec{0} \\ x^{(k+1)} = D^{-1}(b - Rx^{(k)}) \end{cases}$$

L'avantage flagrant de cette méthode de calcul, est que les éléments n'ont aucune dépendance verticale. Elle est donc très simple à parallèliser.
En effet,en écrivant la formule de recurance pour un élement du vecteur $x^{(k+1)}$, on obtient :

$$\forall k \in \mathbb{N}, \forall i \in [\![1, n]\!], \qquad x_i^{(k+1)} = \frac{1}{a_{i,i}}(b_i - \sum_{i \neq j} a_{i,j} \times x_j^{(k)}) \tag{5}$$

La seul contrainte devient alors que chaque processeur doit connaître toutes les composantes de $x^{(k)}$ afin de pouvoir itérer, et il faut donc les communiquer, ce que nous feront en utilisant la bibliothèque MPI.

2.3 Preuve et Convergence

D'après le théorème de la méthode du point fixe, on sait que la suite $x_{n+1} = f(x_n)$ converge si la fonction f est contractante aka k-lipshitzienne avec $k < 1$
La matrice $D^{-1}R$ étant une application linéaire, trouvons une condition nécessaire et suffisante pour qu'elle soit contractante

Soit a, b deux vecteurs dans $\mathbb{R}^n$, $k \in [0, 1[$ et notons $C = D^{-1}R$,

$$|Ca - Cb| \leq k|a - b| \tag{6}$$
$$\Leftrightarrow \qquad ||C - kI|| \leq 0 \tag{7}$$

Cela revient à dire que le rayon spectral de la matrice C noté $\rho(C)$ doit être strictement inférieur à 1
On peut donc affirmer que :

$$\lim_{k \to \infty} x^{(k)} = x \Leftrightarrow \rho(C) = \rho(D^{-1}R) < 1$$

Cherchons une condition simple suffisante pour affirmer que le rayon spectral de la matrice C soit inférieur à 1.

Soit λ valeur propre de C ie $\forall y \in \mathbb{R}^n, Cy = \lambda y$
On a alors, $\forall (i,j) \in [\![1,n]\!]^2$

$$
\begin{aligned}
& ||\lambda y||_\infty = |\sum_{j=1}^{n} c_{i,j} y_j && (8)\\
\Leftrightarrow \quad & \lambda ||y||_\infty = |\sum_{j \neq i} \frac{a_{i,j}}{a_{i,i}} y_j| && (9)\\
\Leftrightarrow \quad & \lambda ||y||_\infty = |\frac{1}{a_{i,i}}| \times |\sum_{j \neq i} a_{i,j} y_j| && (10)\\
\Leftrightarrow \quad & \lambda ||y||_\infty \leq |\frac{1}{a_{i,i}}| \times \sum_{j \neq i} |a_{i,j} y_j| && (11)\\
\Leftrightarrow \quad & \lambda ||y||_\infty \leq |\frac{1}{a_{i,i}}| \times \sum_{j \neq i} |a_{i,j}| \times ||y||_\infty && (12)\\
\Leftrightarrow \quad & \lambda \leq |\frac{1}{a_{i,i}}| \times \sum_{j \neq i} |a_{i,j}| && (13)
\end{aligned}
$$

Une condition suffisante pour que $\lambda < 1$ serait alors :

$$\forall (i,j) \in [\![1,n]\!]^2 \quad |a_{i,i}| > \sum_{i \neq j} |a_{i,j}|$$

ie A est à diagonale strictement dominante

2.4 Vecteur erreur et résidu

Afin de mesurer la convergence de la méthode de Jacobi (etd'avoir une condition d'arrêt), on défini le vecteur erreur relative suivant :

$$e^{(k+1)} = ||x^{(k+2)} - x^{(k+1)}||_\infty = || - C(x^{(k+1)} - x^{(k)})||_\infty = || - C||_\infty e^{(k)}$$

On a donc $e^{(k)} = (-1)^k ||C||_\infty^k e^{(0)}$
On a de plus le vecteur erreur :

$$\epsilon^{(k)} = ||x - x^{(k)}||_\infty$$

$e^{(k)}$ est l'erreur relative de la méthode à l'itération k, mais comme on a pas connaissance du véritable vecteur x, on l'utilise comme référence pour le test d'arrêt.
En effet,

$$\begin{aligned}
e^{(k)} &= ||x^{(k+1)} - x^{(k)}||_\infty && (14)\\
&= ||D^{-1}b - Cx^{(k)} - x^{(k)}||_\infty && (15)\\
&= ||x + Cx - Cx^{(k)} - x^{(k)}||_\infty && (16)\\
&= ||x - x^{(k)} + C(x - x^{(k)})||_\infty && (17)\\
&= ||(C + I)(x - x^{(k)})||_\infty && (18)
\end{aligned}$$

On obtient alors par inégalité triangulaire :

$$e^{(k)} = ||I + C||_\infty \epsilon^{(k)} \leq \epsilon^{(k)} + ||C||_\infty \epsilon^{(k)}$$

Et comme on a vu précedemment que le rayon spectrale de C est nécessairement inférieur à 1, et donc sa norme est également inférieure à 1 on a donc :

$$e^{(k)} \leq 2\epsilon^{(k)}$$

On peut alors considérer que l'erreur relative est suffisament proche de l'erreur réelle pour l'utiliser comme test d'arrêt.

3 Implémentation

3.1 Architecture

Le projet est divisé en plusieurs sous-dossiers
- **/lib** qui contient des librairies pour manipuler les matrices, les vecteurs ainsi que les opérations basiques de manipulation de ces deux structures de données.
- **/src** qui contient le fichier *main* qui fait tourner le programme de l'agorithme de Jacobi.
- **/data** qui contient les fichiers des matrices et des vecteurs ainsi que les métadonnées contenant la taille des matrices considérées.
- **/bin** qui contientles binaires *main* et *test* afin de faire tourner le programme ou de tester les librairies
- **/test** qui contient le code source du binaire de test dont l'objectif est de tester les fonctions des lirbairies.

Le projet est de plus constitué d'un Makefile qui contient les procédures suivantes :
- *make all* (ou simplement *make*) pour compiler les librairies, main et les tests
- *make test* pour compiler les tests
- *make clean* pour supprimer les fichiers objets

3.2 Structures de données

Dans ce projet, j'ai créé deux structures de données realitvement similaires pour les vecteurs et les matrices. Chaque structure dispose d'un type complexe qui lui est associé pour des raisons d'ergonomie.

Ces structures sont contenues dans les fichiers `lib/matrix.h` et `lib/vector.h`.

La structure Matrice est composé de deux entiers non signés pour les dimensions et d'un tableau de *double* à 2 dimensions pour contenir les valeurs de la matrice.

La structure Vecteur est très similaire mais n'a qu'une seule dimension.

3.3 Librairies

Ce projet est fourni avec 3 libraries situées dans les fichiers C du dossier **/lib**.