

ANALYSE NUMÉRIQUE

T.P. N° 2

SHERYL LAFOSSE-MARIN

SOPHIE RUELLAND

N° Groupe : D17

Observations :

| | | | |
|---------|---------------|---|----------------|
| | ANALYSE | : | |
| | RÉSULTATS | : | |
| Notes : | PROGRAMMATION | : | Total : |
| | RAPPORT | : | |

EISTI, 19 mai 2008

Etude des méthodes de résolution de systèmes linéaires

19 mai 2008

Table des matières

| | | |
|----------|--|----------|
| 1 | Introduction | 2 |
| 2 | La méthode de décomposition LU | 3 |
| 2.1 | Etude préalable | 3 |
| 2.2 | Résultats obtenus | 3 |
| 2.2.1 | Questions a,b et c | 3 |
| 2.2.2 | Discussion des résultats obtenus | 3 |
| 3 | La méthode de Jacobi | 4 |
| 3.1 | Etude préalable | 4 |
| 3.2 | Résultats obtenus | 5 |
| 3.2.1 | Questions a, b et c | 5 |
| 3.2.2 | Discussion des résultats obtenus | 5 |
| 4 | La méthode relaxée de Jacobi | 5 |
| 4.1 | Etude préalable | 5 |
| 4.2 | Résultats obtenus | 6 |
| 4.2.1 | Questions a, b et c | 6 |
| 4.2.2 | Discussion des résultats obtenus | 6 |
| 5 | Bilan de l'étude des différents méthodes de résolution des systèmes linéaires | 6 |

1 Introduction

Le but du TP est de comparer trois méthodes d'inversion des matrices : une méthode directe, résolution par décomposition LU et deux méthodes itératives, la méthode de Jacobi et la méthode relaxée de Jacobi.

2 La méthode de décomposition LU

Dans cette partie, nous allons étudier le principe de la décomposition LU puis rendre compte des résultats obtenus.

2.1 Etude préalable

La méthode de Décomposition LU est une méthode directe, elle établit en effet la solution en un nombre fini et prédéterminé d'étapes. Le nombre d'étapes est fonction de la taille de la matrice.

Les méthodes directes ont pour principe de remplacer la résolution du système $Ax=b$ par un système plus facile à résoudre, par exemple par un système équivalent de matrice triangulaire ou diagonale.

Pour diagonaliser une matrice, on doit calculer ses éléments propres mais c'est un problème difficile à résoudre. On utilise donc la méthode de décomposition LU, c'est à dire : remplacer A par le produit de deux matrices triangulaires, notées L triangulaire inférieure, et U triangulaire supérieure.

On résout alors successivement deux systèmes triangulaires : $Ly = b$ puis $Ux = y$ d'où $LUx = Ly$ et ainsi $A = LU$

Le premier système permet de calculer y par substitution directe, le second système permet d'obtenir la solution cherchée par substitution inverse.

2.2 Résultats obtenus

2.2.1 Questions a,b et c

On étudie la précision numérique du résultat de la méthode de la décomposition LU en utilisant le critère de la norme infinie de $\|Ax - b\|_\infty$ en fonction de la taille de la matrice A de taille n.

On effectue cette étude pour différentes tailles de la matrice A, $n = 10, 20, 40, 80$. Pour cela nous avons développé notre fonction `resoudreLU()`.

La console de calcul du logiciel Scilab nous donne : `resoudreLU()`

Pour $n = 10$ la norme $\|Ax - b\|_\infty$ vaut $2.220D - 16$, le nombre d'itérations vaut 90. Le temps de calcul de LU et la résolution en seconde valent 0.

Pour $n = 20$, la norme $\|Ax - b\|_\infty$ vaut $2.220D - 16$, le nombre d'itérations vaut 380. Le temps de calcul de LU et la résolution en seconde valent 0.

Pour $n = 40$, la norme $\|Ax - b\|_\infty$ vaut $2.220D - 16$, le nombre d'itérations vaut 1560. Le temps de calcul de LU et la résolution en seconde valent 0.03125.

Pour $n = 80$, la norme $\|Ax - b\|_\infty$ vaut $2.220D - 16$, le nombre d'itérations vaut 6320. Le temps de calcul de LU et la résolution en seconde valent 0.046875.

2.2.2 Discussion des résultats obtenus

On observe différents résultats.

Tout d'abord, notons que la norme est indépendante de la taille de la matrice. Cette norme est en effet bornée pour tous les calculs effectués. Elle est bornée par le plus petit nombre après 0 pour le logiciel Scilab. De plus, le temps CPU

et le nombre de calculs effectués au sein de l'algorithme croissent en fonction de la taille de la matrice (la complexité est exponentielle pour le nombre de calculs effectués).

La décomposition LU est particulièrement utile quand on veut inverser une matrice. Dans ce cas, on a $AX = I$ et la décomposition LU donne un premier calcul $LY = I$, d'où $Y = L^{-1}$ et ensuite on obtient la matrice inverse par la résolution de $UX = Y$. L'avantage de cette technique est de pouvoir effectuer une approche de la solution optimale avec une erreur très faible, et cela quelque soit la taille de la matrice.

3 La méthode de Jacobi

Dans cette partie, nous allons étudier le principe de la méthode de Jacobi puis rendre compte des résultats obtenus.

3.1 Etude préalable

La méthode de Jacobi est une méthode itérative de résolution de système linéaire de la forme $Ax = b$. Nous pouvons réécrire ce système sous la forme $x = (A+I)x - b$.

Pour résoudre ce système, on utilise une suite $x(k)$ qui converge vers un point fixe x , solution du système d'équations linéaires.

La suite $x(k)$ est définie par

$$x(0) \in R^n, x(k+1) = (A+I)x(k) - b.$$

Ce qui est important, c'est la capacité de l'algorithme à atteindre un point fixe et ceci le plus rapidement possible.

Les algorithmes itératifs stationnaires procèdent à la décomposition de la matrice A en deux matrices, selon le schéma

$$A = M - N, \text{ avec } M \text{ régulière.}$$

Il faut que la matrice M soit facile à inverser (ou que le système correspondant soit facile à résoudre).

Pour les matrices M faciles à extraire de A et immédiatement inversibles, on peut penser à la diagonale de A ou à une matrice triangulaire. On décompose la matrice A de la façon suivante :

$$A = D + L + U$$

avec D la diagonale, L la partie en dessous de la diagonale et U , la partie au dessus.

En prenant $M = D$ et $N = -L - U$ on obtient :

$$x(0) \in R^n, x(k+1) = -D^{-1}(L+U)x(k) + D^{-1}b$$

3.2 Résultats obtenus

3.2.1 Questions a, b et c

On effectue cette étude pour différentes taille de la matrice A, $n = 10, 20, 40, 80$. Pour cela nous avons développé notre fonction `resolveJacobi(A, b, n)` qui reçoit la matrice A, la matrice b, et la dimension n en paramètres.

La console de calcul du logiciel Scilab nous donne : `resoudreJacobi()`

Pour $n = 10$, la norme $\|Ax - b\| \propto$ vaut $8,195D - 12$, le nombre d'itérations nécessaires vaut 562. Le temps CPU utilisé en secondes vaut 0.0625

Pour $n = 20$, la norme $\|Ax - b\| \propto$ vaut $6,146D - 12$, le nombre d'itérations nécessaires vaut 1981. Le temps CPU utilisé en secondes vaut 0.15625

Pour $n = 40$, la norme $\|Ax - b\| \propto$ vaut $4,401D - 12$, le nombre d'itérations nécessaires vaut 7232. Le temps CPU utilisé en secondes vaut 0.484375

Pour $n = 80$, la norme $\|Ax - b\| \propto$ vaut $3,142D - 12$, le nombre d'itérations nécessaires vaut 26887. Le temps CPU utilisé en secondes vaut 3.234375

3.2.2 Discussion des résultats obtenus

A partir de ces résultats, on observe que la norme diminue en fonction de la taille de la matrice. Donc, plus la matrice est grande de taille, plus l'erreur commise par les calculs est petite.

Le nombre d'itérations en revanche augmente de façon exponentielle, ce qui est expliqué par le nombre de boucle à l'intérieur de la fonction `resolveJacobi(A, b, n)`, mais aussi à cause des produits de matrice, et des inversions de matrice sous-jacentes au calcul de la solution du système linéaire.

Le temps CPU, lui augmente très fortement en fonction de la taille de la matrice.

Les deux inconvénients de cette méthode sont d'une part que D peut être assez éloigné de A, d'autre part qu'il faut conserver en mémoire, à chaque itération, toutes les composantes du vecteur $x(k)$ et toutes les composantes du vecteur $x(k+1)$.

Ce que l'on peut dire c'est que cette méthode produit un résultat avec une erreur proportionnelle à la taille de la matrice. Elle conviendrait pour la résolution de système linéaire de grande taille.

4 La méthode relaxée de Jacobi

Dans cette partie, nous allons étudier le principe de la méthode relaxée de Jacobi puis rendre compte des résultats obtenus.

4.1 Etude préalable

On injecte au calcul de $x(k+1)$ une portion μ de la valeur issue du calcul du schéma de l'itération, le reste $1 - \mu$ de la portion est réservé à la valeur $x(k)$, avec $0 < \mu < 2$. Le coefficient μ s'appelle coefficient de relaxation.

Le schéma itératif est donné par

$$x(0) \in R^n, x(k+1) = (1 - \mu)x(k) - \mu D^{-1}(L + U)x(k) + \mu D^{-1}b.$$

Si $\mu < 1$, nous avons une sous-relaxation et si $\mu > 1$, une sur-relaxation. Pour $\mu = 1$, nous avons le schéma de Jacobi non-relaxé.

4.2 Résultats obtenus

4.2.1 Questions a, b et c

Dans un premier temps, nous étudions l'influence du nombre μ sur le nombre d'itérations lors de la résolution. Le but est de trouver μ tel que, le nombre d'erreurs soit le plus petit possible, c'est à dire tel qu'il y ait le moins d'itérations possibles.

En fabriquant un graphique qui représente le nombre d'itérations en fonction de μ , on voit que le nombre d'itérations est minimum pour $\mu = 1$. Le μ optimal est donc égal à 1.

Dans cette partie, on effectue une étude de la résolution de système linéaire en utilisant la méthode de Jacobi relaxée en choisissant comme coefficient de relaxation $\mu = 1$.

On effectue cette étude pour différentes tailles de matrice A, $n = 10, 20, 40, 80$.

La console de calcul du logiciel Scilab nous donne : `resolveJacobiRelaxe()`

Pour $n = 10$, la norme $\|Ax - b\|_\infty$ vaut $2,220D - 12$, le nombre d'itérations nécessaires vaut 824. Le temps CPU utilisé en secondes vaut 0.09375

Pour $n = 20$, la norme $\|Ax - b\|_\infty$ vaut $2,220D - 12$, le nombre d'itérations nécessaires vaut 2949. Le temps CPU utilisé en secondes vaut 0.359375

Pour $n = 40$, la norme $\|Ax - b\|_\infty$ vaut $2,220D - 12$, le nombre d'itérations nécessaires vaut 10868. Le temps CPU utilisé en secondes vaut 2.5

Pour $n = 80$, la norme $\|Ax - b\|_\infty$ vaut $2,220D - 12$, le nombre d'itérations nécessaires vaut 40768. Le temps CPU utilisé en secondes vaut 24.421875 .

4.2.2 Discussion des résultats obtenus

La norme $\|Ax - b\|_\infty$ est bornée à ε .

Cette méthode nous permet donc d'avoir une norme bornée, et cela indépendamment de la taille de la matrice. Cependant le nombre d'itérations est très grand. On observe une variation exponentielle du nombre d'itérations en fonction de la taille de la matrice.

Le temps CPU est lui encore plus grand, beaucoup plus grand que le temps CPU observé par les autres méthodes.

5 Bilan de l'étude des différents méthodes de résolution des systèmes linéaires

En se basant sur le critère de la précision, seul la méthode LU et la méthode de Jacobi relaxée donnent une précision sur la norme assez satisfaisante ; norme inférieure à ε , et surtout indépendante de la taille de la matrice.

L'occupation de mémoire peut être évaluée par le nombre d'itérations effectuées par la fonction. La méthode de Jacobi relaxée a la plus grande valeur.

D'un point de vue temporel, on observe différents temps de calculs. En effet les temps les plus grands sont atteints par la méthode de Jacobi relaxée. La méthode la plus rapide étant la méthode LU.