RAPPORT TD/TP 5 CALCUL NUMÉRIQUE

Dorian Abaul

Décembre 2024

Table des matières

1 Introduction

L'objectif de ce TD/TP est d'appliquer un ensemble d'algorithmes étudiés en cours et en TD à la résolution d'un système linéaire obtenu par discrétisation de l'équation de la chaleur 1D stationnaire via la méthode des différences finies. Les implémentations sont réalisées en C, en s'appuyant sur les bibliothèques BLAS et LAPACK. Pour validation, des comparaisons avec des implémentations Scilab réalisées en TD sont effectuées.

Discrétisation de l'équation de Poisson 1D (équation de la chaleur stationnaire)

L'équation de Poisson 1D stationnaire est donnée par :

$$\frac{d^2u(x)}{dx^2} = f(x),$$

avec des conditions aux limites, typiquement $u(0) = 0$ et $u(L) = 0$.

1. Discrétisation par la méthode des différences finies

En discrétisant l'équation de Poisson à l'aide des différences finies sur un maillage uniforme de points $x_0 = 0, x_1, \ldots, x_n = L$, avec un pas de discrétisation $\Delta x = \frac{L}{n}$, la dérivée seconde est approximée par :

$$\frac{d^2u(x_i)}{dx^2} \approx \frac{u_{i+1} - 2u_i + u_{i-1}}{(\Delta x)^2}.$$

Ainsi, l'équation devient :

$$\frac{u_{i+1} - 2u_i + u_{i-1}}{(\Delta x)^2} = f(x_i).$$

2. Formulation du système linéaire

Le système linéaire résultant est de la forme :

$$A\mathbf{u} = \mathbf{f},$$

où la matrice A est tridiagonale et a la forme :

$$A = \frac{1}{(\Delta x)^2} \begin{bmatrix} -2 & 1 & 0 & \cdots & 0 \\ 1 & -2 & 1 & \cdots & 0 \\ 0 & 1 & -2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & 1 \\ 0 & 0 & 0 & \cdots & -2 \end{bmatrix}.$$

Le vecteur $\mathbf{u}$ contient les inconnues $u_1, u_2, \ldots, u_{n-1}$, et le vecteur $\mathbf{f}$ contient les valeurs $f(x_1), f(x_2), \ldots, f(x_{n-1})$.

2 Bibliothèques BLAS et LAPACK

Utilisation de BLAS et LAPACK

Les bibliothèques BLAS et LAPACK offrent des routines optimisées pour les calculs mathématiques, notamment les opérations sur les matrices et les vecteurs. Pour les utiliser en C, il est important de bien configurer les appels à ces routines et de comprendre le format de stockage des matrices.

Format de stockage des matrices

LAPACK utilise par défaut un stockage en colonnes (*column-major order*), contrairement à C, qui utilise un stockage par lignes (*row-major order*). La constante `LAPACK_COL_MAJOR` permet d'indiquer le mode de stockage à utiliser.

La dimension principale (*leading dimension*)

La dimension principale, `ld`, représente l'espacement mémoire entre les lignes ou colonnes successives d'une matrice. Elle est cruciale pour manipuler les matrices linéarisées en mémoire.

Fonctions principales de LAPACK

- `dgbmv` : Produit matrice-vecteur pour une matrice stockée en bande.
- `dgbtrf` : Factorisation LU d'une matrice en bande.
- `dgbtrs` : Résolution d'un système linéaire avec une matrice précédemment factorisée.
- `dgbsv` : Combinaison de `dgbtrf` et `dgbtrs` pour résoudre directement un système linéaire.

3 Résolution des systèmes linéaires avec LAPACK

Matrice Poisson 1D au format GB

La matrice Poisson 1D est généralement creuse, avec des éléments non nuls concentrés autour de la diagonale principale. Elle est stockée au format GB à l'aide de la fonction `set_GB_operator_colMajor_poisson1D`.

Utilisation de `dgbmv`

La fonction `cblas_dgbmv` permet de calculer efficacement le produit matrice-vecteur. Par exemple, pour un vecteur donné, les résultats sont :

$$[-5.000000, 0.000000, -0.000000, 0.000000, 0.000000, 0.000000, -0.000000, 5.000000].$$

Validation des résultats

Pour valider les résultats, une solution exacte est calculée à partir de la solution analytique ou d'une résolution avec Scilab. Le résidu relatif est évalué à l'aide de `dnrm2`.

Exemple de validation :

— **Erreur relative directe :** relres $= 6.813851 \times 10^{-1}$

— **Erreur après validation :** 4.36×10^{-1}

---

4 Méthodes de résolution itérative

Richardson

La méthode de Richardson est utilisée pour améliorer la convergence avec un paramètre α, calculé par :

$$\alpha = \frac{2}{\max(\lambda) + \min(\lambda)},$$

à partir des valeurs propres de la matrice. On obtient comme valeurs optimales pour la méthode de alpha-Richardson

```
Alpha optimal pour Richardson : 0.500000
Méthode de Richardson : nombre d'itérations = 50
Méthode de Richardson : erreur avant relative = 1.000000e+00
Premières valeurs de la solution : -120.099010 6376214746.942475 -187035640598.084900
```

En traçant l'historique de convergence, on constate que la méthode tend rapidement vers une solution.En effet, l'erreur avant relative sert à évaluer la précision d'une valeur approchée par rapport à la solution exacte ainsi plus l'erreur est faible plus la valeur approchée s'approche de la solution exacte.

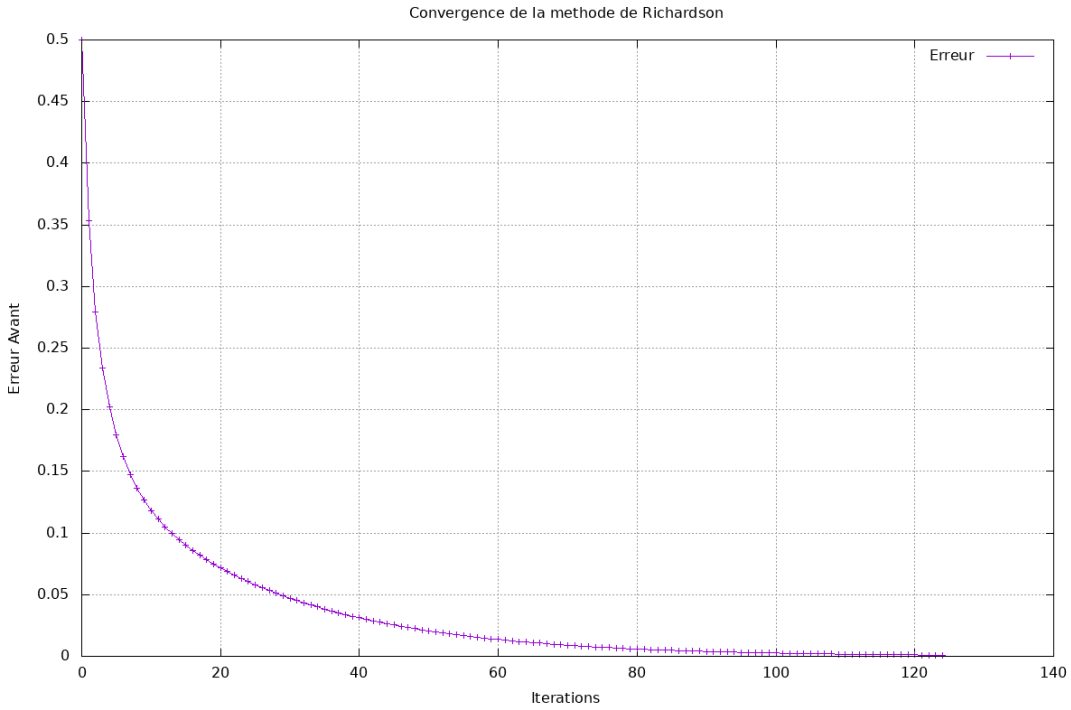

FIGURE 1 – Convergence de Richardson en fonction des itérations

Jacobi

D'après la discrétisation de l'équation on obtient la matrice :

$$A = \begin{bmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{bmatrix}$$

On peut décomposer cette matrice en D, L et U grâce à la formule $A = DLU$

- D est une matrice diagonale, où chaque élément sur la diagonale est égal à 2

$$D = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix}$$

- L est une matrice triangulaire inférieure, avec -1 dans les positions sous la diagonale

$$L = \begin{bmatrix} 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

- Et enfin U est une matrice triangulaire supérieure, avec -1 dans les positions au-dessus de la diagonale

$$U = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

La matrice d'itération de Jacobi est $M = D^{-1}$:

$$M = D^{-1} = \begin{bmatrix} 0.5 & 0 & 0 & 0 \\ 0 & 0.5 & 0 & 0 \\ 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0.5 \end{bmatrix}$$

La méthode de Jacobi repose sur une matrice diagonale D :

$$x^{k+1} = x^k + D^{-1}r^k,$$

avec $r^k = b - Ax$. On obtient comme valeurs optimales pour la méthode de Jacobi

```
Méthode de Jacobi : nombre d'itérations = 50
Méthode de Jacobi : erreur avant relative = 1.022539e+00
Premières valeurs de la solution : 47.210833 85.522827 116.037141
```

On constate que la méthode tend elle aussi rapidement vers une solution.

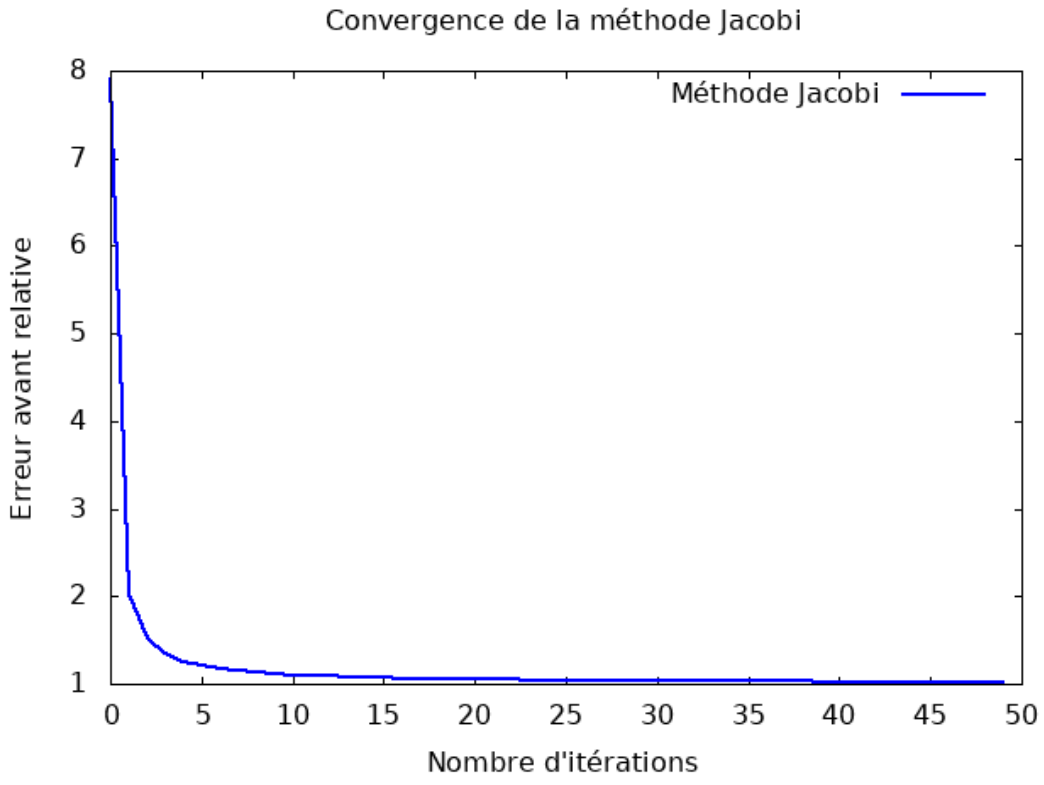

FIGURE 2 – Convergence de Jacobi en fonction des itérations

Gauss-Seidel

En séparant la matrice comme pour Jacobi la matrice A en D, L et U grâce à la formule $A = DLU$ on peut déterminer la matrice d'itération de Gauss-Seidel est définie par la relation $M = D - E = D + L$, où D est la matrice diagonale et L est la matrice triangulaire inférieure. Cette matrice d'itération peut être exprimée comme suit :

$$M = D + L = \begin{bmatrix} 2 & 0 & 0 & 0 \\ -1 & 2 & 0 & 0 \\ 0 & -1 & 2 & 0 \\ 0 & 0 & -1 & 2 \end{bmatrix}$$

Une amélioration de Jacobi, utilisant une décomposition stricte pour améliorer la convergence. On obtient comme valeurs optimales pour la méthode de Gauss-Seidel

```
Méthode de Gauss-Seidel : nombre d'itérations = 50
Méthode de Gauss-Seidel : erreur avant relative = 1.023979e+00
Premières valeurs de la solution : 35.143060 61.074209 83.172337
```

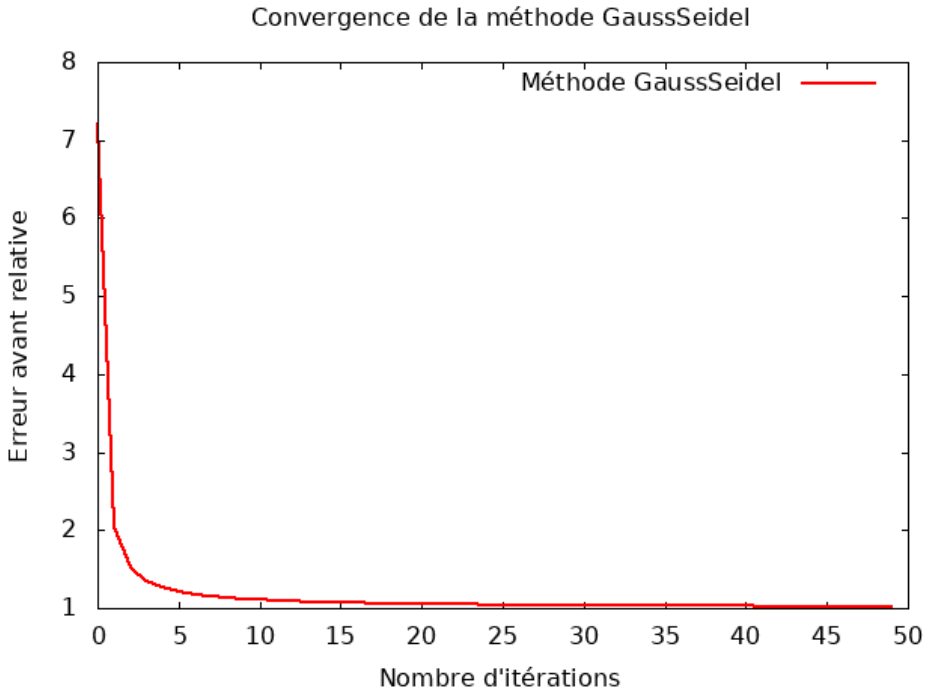

FIGURE 3 – Convergence de Gauss-Seidel en fonction des itérations

5 Analyse des performances

— **Complexité de la factorisation LU (`dgbtrf`) :** $O(n \cdot kl \cdot ku)$, avec kl et ku les largeurs des bandes.

— **Temps d'exécution (mesuré avec `time`) :**

— `TRF` : 18×10^{-6} s

— `TRI` : 1×10^{-6} s
— `SV` : 6×10^{-6} s

sectionAnalyse des performances et complexité des calculs

Complexité de la factorisation LU

Pour une matrice en bande avec n lignes, et des largeurs de bande inférieure et supérieure kl et ku, la complexité de la factorisation LU (`dgbtrf`) est :

$$O(n \cdot kl \cdot ku).$$

Cette complexité provient du traitement des $kl + ku + 1$ bandes non nulles de la matrice.

Complexité de la résolution triangulaire

La résolution triangulaire (`dgbtrs`) a une complexité proportionnelle à la factorisation LU, soit :

$$O(n \cdot kl \cdot ku).$$

Comparaison entre algorithmes directs et itératifs

— **Algorithmes directs (LU) :**
  — Efficaces pour des systèmes linéaires bien conditionnés.
  — Complexité polynomiale en fonction de la taille de la matrice.
— **Algorithmes itératifs (Richardson, Jacobi, Gauss-Seidel) :**
  — Adaptés aux matrices creuses et mal conditionnées.
  — Complexité dépend du nombre d'itérations requises pour atteindre une tolérance donnée.

Performance mesurée

Les temps d'exécution mesurés (en secondes) pour une matrice tridiagonale sont :

— `TRF` : 18×10^{-6} s
— `TRI` : 1×10^{-6} s
— `SV` : 6×10^{-6} s

Les algorithmes sont optimisés pour les matrices tridiagonales et les matrices creuses.