Rapport de TP : Méthodes directes et itératives pour résoudre l'équation de la chaleur 1D

Auteur : Xiang BIAN

Enseignant : T. Dufaud, J. Gurhem

3 Janvier 2026

Contents

1 Introduction

La résolution numérique des équations aux dérivées partielles joue un rôle central en analyse numérique et en modélisation scientifique. De nombreux phénomènes physiques — tels que la diffusion de la chaleur, la conduction électrique ou encore les écoulements stationnaires — sont décrits par des équations elliptiques dont la résolution analytique n'est possible que dans des cas très particuliers.

Dans ce travail, on s'intéresse à la résolution numérique du problème de Poisson unidimensionnel, qui constitue un modèle fondamental et un banc d'essai classique pour l'étude des méthodes de discrétisation et de résolution de systèmes linéaires. L'objectif principal est de transformer le problème continu en un système linéaire issu d'une discrétisation par différences finies, puis de le résoudre efficacement à l'aide de méthodes directes et/ou itératives.

2 Problème étudié et mise en équation

2.1 Modèle mathématique

On considère le problème de Poisson unidimensionnel posé sur l'intervalle $(0, 1)$:

$$\begin{cases} -u''(x) = f(x), \quad x \in (0,1), \\ u(0) = \alpha, \\ u(1) = \beta, \end{cases} \tag{1}$$

où $f : (0,1) \to \mathbb{R}$ est une fonction donnée, supposée suffisamment régulière, et $\alpha, \beta \in \mathbb{R}$ sont des conditions aux limites de type Dirichlet.

2.2 Discrétisation par différences finies

On introduit une discrétisation uniforme de l'intervalle $[0, 1]$ avec n points intérieurs et $n + 2$ points au total :

$$x_i = ih, \quad i = 0, \dots, n+1, \qquad h = \frac{1}{n+1}.$$

La fonction inconnue $u(x)$ est approchée sur le maillage par les valeurs discrètes $u_i \approx u(x_i)$. Pour $i = 1, \dots, n$, la dérivée seconde est approchée par le schéma centré d'ordre deux :

$$u''(x_i) \approx \frac{u_{i-1} - 2u_i + u_{i+1}}{h^2}.$$

En injectant dans (1) on obtient, pour chaque point intérieur,

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

2.3 Système linéaire obtenu

La discrétisation conduit à un système linéaire de taille n de la forme

$$Au = b,$$

où $u = (u_1, \dots, u_n)^\top$, $A \in \mathbb{R}^{n \times n}$ est une matrice tridiagonale symétrique définie positive, et $b \in \mathbb{R}^n$ est le second membre. La matrice A s'écrit explicitement :

$$A = \frac{1}{h^2} \begin{pmatrix} 2 & -1 & & \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2 \end{pmatrix}.$$

Le second membre est donné par $b_i = f(x_i)$, auquel s'ajoutent les contributions des conditions aux limites :

$$b_1 \leftarrow b_1 + \frac{\alpha}{h^2}, \qquad b_n \leftarrow b_n + \frac{\beta}{h^2}.$$

3 Méthodes de résolution

3.1 Méthode directe (Exercices 3 à 6)

Dans cette section, nous présentons les outils et algorithmes de résolution directe utilisés pour résoudre le système linéaire issu de la discrétisation de l'équation 1D. L'accent est mis sur le stockage bande, l'utilisation des bibliothèques BLAS/LAPACK et l'analyse théorique des complexités.

3.1.1 Déclaration et allocation des matrices (Exercice 3.1)

Pour utiliser efficacement les bibliothèques BLAS et LAPACK en langage C, les matrices doivent être stockées dans des tableaux contigus en mémoire, compatibles avec les conventions de stockage de LAPACK. Dans notre implémentation, les matrices sont stockées sous forme de tableaux `double*`, alloués dynamiquement, et organisés en ordre colonne (*column-major order*), comme en Fortran. Ce choix garantit la compatibilité avec les routines LAPACK sans nécessiter de copies ou de transpositions supplémentaires.

3.1.2 Ordre de stockage et constante `LAPACK_COL_MAJOR` (Exercice 3.2)

La constante `LAPACK_COL_MAJOR` indique que les matrices sont stockées en priorité colonne, c'est-à-dire que les éléments d'une même colonne sont contigus en mémoire. Ce format est imposé par LAPACK et correspond au stockage natif utilisé dans les routines Fortran. Dans ce cadre, l'élément a_{ij} d'une matrice est stocké à une position mémoire dépendant de l'indice de ligne i et de la dimension principale.

3.1.3 Dimension principale (*leading dimension*) (Exercice 3.3)

La dimension principale, généralement notée `ld`, correspond au nombre de lignes effectivement allouées en mémoire pour représenter une colonne de la matrice. Dans le cas d'un stockage bande de type *General Band* (GB) avec k_ℓ sous-diagonales et k_u sur-diagonales, la dimension principale vaut :

$$\texttt{ld} = k_\ell + k_u + 1.$$

Cette valeur permet à LAPACK de localiser correctement les éléments de la matrice stockée de manière compacte.

3.1.4 Produit matrice–vecteur en format bande : `DGBMV` (Exercices 3.4 et 4)

La routine BLAS `DGBMV` permet de calculer le produit matrice–vecteur $y = Ax$ dans le cas où la matrice A est stockée au format *General Band*. Cette fonction exploite directement la structure bande et évite les opérations inutiles sur les coefficients nuls. Dans le cadre de ce TP, `DGBMV` est utilisée pour valider le stockage bande de la matrice de Poisson 1D, et pour calculer le résidu $r = Au - f$ lors de la vérification de la solution.

3.1.5 Résolution directe avec LAPACK : `DGBTRF`, `DGBTRS`, `DGBSV` (Exercices 3.5 à 3.7 et Exercice 5)

La résolution directe du système linéaire est réalisée à l'aide des routines LAPACK dédiées aux matrices bandes :

`DGBTRF` effectue la factorisation LU de la matrice bande A ;

`DGBTRS` résout le système linéaire à partir de la factorisation LU ;

`DGBSV` combine les deux étapes précédentes dans une seule routine.

Ces méthodes permettent d'obtenir la solution du système (à l'erreur d'arrondi près) sans implémenter manuellement l'algorithme de Thomas. Le pivotement partiel est pris en charge automatiquement par LAPACK, garantissant la stabilité numérique de la méthode.

3.1.6 Factorisation LU pour matrices tridiagonales (Exercice 6)

Dans le cas particulier des matrices tridiagonales, la factorisation LU peut être réalisée avec une complexité réduite en exploitant la faible largeur de bande. Une factorisation LU spécifique repose sur la mise à jour séquentielle des coefficients de la diagonale principale et de la sous-diagonale ; cette approche est équivalente à l'algorithme de Thomas et permet de conserver une complexité linéaire en temps et en mémoire.

3.1.7 Analyse théorique des complexités (Exercices 5 et 6)

Grâce au stockage bande, les coûts de calcul et de stockage sont considérablement réduits par rapport à une approche dense. Dans le cas tridiagonal :

Complexité mémoire : $\mathcal{O}(n(k_\ell + k_u + 1)) = \mathcal{O}(n)$;

Complexité en temps : $\mathcal{O}(n)$.

Ces résultats justifient l'utilisation des méthodes directes basées sur le stockage bande pour la résolution du problème 1D.

3.2 Méthodes itératives (Exercices 7 à 9)

Les méthodes itératives constituent une alternative aux méthodes directes pour la résolution des systèmes linéaires de grande taille. Contrairement aux méthodes directes, elles construisent une suite d'approximations successives de la solution, en s'appuyant sur une décomposition de la matrice et un critère d'arrêt basé sur le résidu. Dans cette section, nous présentons les méthodes itératives étudiées : Richardson, Jacobi et Gauss–Seidel.

3.2.1 Principe général des méthodes itératives (Exercice 7)

Soit le système linéaire $Au = f$, où $A \in \mathbb{R}^{n\times n}$ est une matrice tridiagonale issue de la discrétisation par différences finies. Une méthode itérative repose sur une décomposition $A = M - N$ et sur la construction d'une suite $(u^{(k)})_{k\geq 0}$ définie par

$$Mu^{(k+1)} = Nu^{(k)} + f.$$

La convergence dépend du rayon spectral de la matrice d'itération $M^{-1}N$, qui doit être strictement inférieur à 1.

3.2.2 Méthode de Richardson (Exercice 7)

La méthode de Richardson s'écrit sous la forme :

$$u^{(k+1)} = u^{(k)} + \alpha\,(f - Au^{(k)}),$$

où $\alpha > 0$ est un paramètre de relaxation. Cette méthode peut être interprétée comme une descente de gradient appliquée au problème de minimisation associé au système linéaire. Dans le cas où A est symétrique définie positive, la convergence est garantie pour un choix approprié de α. Dans ce TP, le résidu $r^{(k)} = f - Au^{(k)}$ est calculé à chaque itération et son évolution est utilisée comme critère d'arrêt.

3.2.3 Méthode de Jacobi (Exercice 8)

La méthode de Jacobi repose sur la décomposition $A = D-(E+F)$, où D est la matrice diagonale de A, E la partie strictement inférieure, et F la partie strictement supérieure. L'itération de Jacobi s'écrit :

$$u^{(k+1)} = D^{-1}\big((E+F)u^{(k)} + f\big).$$

Cette méthode est simple à implémenter et naturellement parallélisable, car chaque composante de $u^{(k+1)}$ est calculée indépendamment des autres. Pour la matrice tridiagonale du problème 1D, la méthode de Jacobi est convergente, bien que relativement lente.

3.2.4 Méthode de Gauss–Seidel (Exercice 9)

La méthode de Gauss–Seidel utilise une décomposition différente : $A = (D-E)-F$. L'itération associée est donnée par :

$$(D-E)u^{(k+1)} = Fu^{(k)} + f.$$

Contrairement à Jacobi, Gauss–Seidel utilise immédiatement les composantes mises à jour de $u^{(k+1)}$, ce qui améliore généralement la vitesse de convergence. Pour le problème 1D, la matrice étant symétrique définie positive, la convergence de la méthode est assurée.

3.2.5 Critère d'arrêt et validation (Exercice 9)

Pour les méthodes itératives, un critère d'arrêt est nécessaire afin de déterminer la convergence de la suite $(u^{(k)})$. Dans ce TP, le critère retenu est basé sur la norme du résidu :

$$\frac{\|r^{(k)}\|_2}{\|f\|_2} \leq \varepsilon,$$

où ε est une tolérance fixée a priori. Le calcul du résidu est réalisé à l'aide de routines BLAS adaptées au stockage bande, garantissant une évaluation efficace et cohérente avec les méthodes directes.

3.2.6 Complexité théorique

Pour les matrices tridiagonales issues de la discrétisation du problème 1D, le coût d'une itération des méthodes de Richardson, Jacobi ou Gauss–Seidel est de l'ordre de $\mathcal{O}(n)$, et le coût total dépend du nombre d'itérations nécessaires à la convergence, lui-même lié aux propriétés spectrales de la matrice.

4 Validation et résultats numériques

4.1 Validation des méthodes directes

Une solution analytique (lorsqu'elle est disponible pour le choix de f) sert de référence afin de valider l'implémentation. On compare alors la solution numérique à la solution exacte (par exemple via une norme d'erreur).

4.2 Performances des méthodes directes

Les mesures de temps d'exécution pour différentes tailles de problème sont synthétisées par la Figure 1. On observe une croissance compatible avec la complexité attendue pour les opérations dominantes (assemblage, factorisation/résolution) dans le cadre du stockage bande.

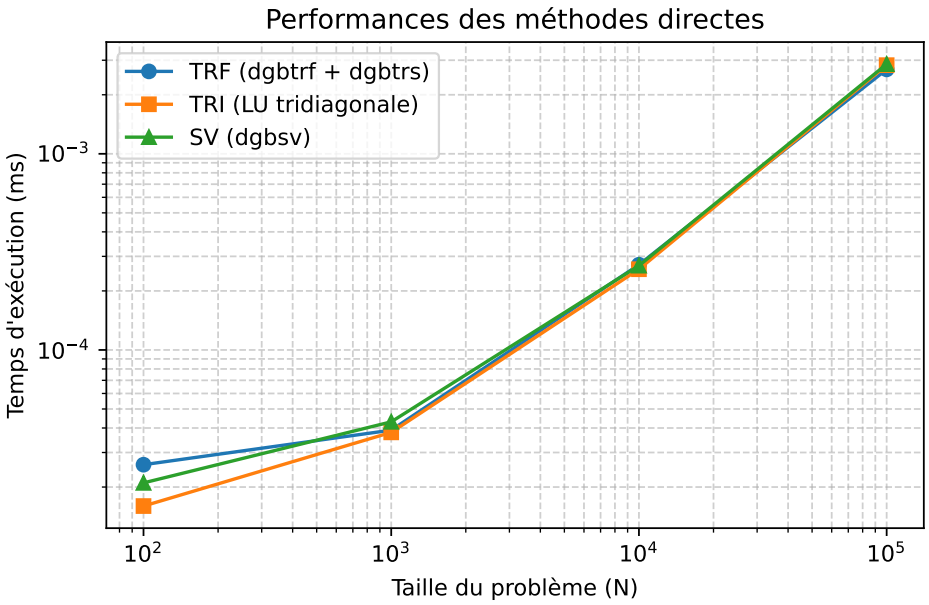

Figure 1: Temps de calcul (méthode directe) en fonction de la taille du problème.

4.3 Convergence des méthodes itératives

La Figure 2 présente l'historique de convergence pour un problème de taille $N = 100$, en échelle semi-logarithmique.

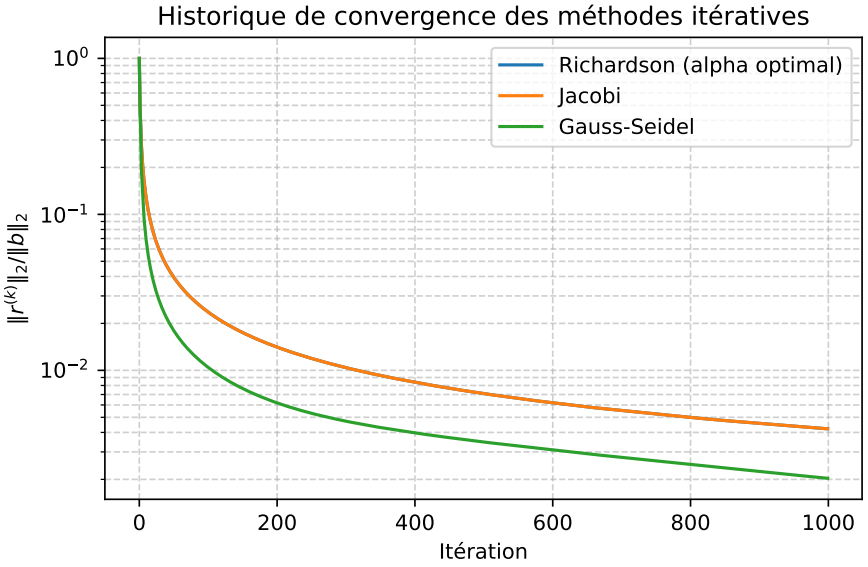

Figure 2: Historique de convergence des méthodes itératives pour $N = 100$.

On observe que les courbes correspondant à la méthode de Richardson (paramètre optimal) et à la méthode de Jacobi sont pratiquement confondues, ce qui est cohérent avec l'analyse théorique pour le problème de Poisson 1D. En revanche, la méthode de Gauss–Seidel converge plus rapidement, grâce à l'utilisation immédiate des composantes nouvellement mises à jour.

4.4 Comparaison du nombre d'itérations et efficacité

Les essais numériques montrent que :

pour de petites tailles (par exemple $N = 10$), toutes les méthodes convergent en un nombre raisonnable d'itérations, avec un avantage visible pour Gauss–Seidel ;

lorsque la taille augmente (par exemple $N = 100$), Richardson (avec α optimal) et Jacobi peuvent nécessiter un nombre d'itérations très élevé ;

Gauss–Seidel reste systématiquement la plus efficace, nécessitant typiquement environ deux fois moins d'itérations pour atteindre la même tolérance.

5 Autres formats de stockage : CSR et CSC

La matrice issue de la discrétisation du Poisson 1D ne contient que trois coefficients non nuls au maximum par ligne (sous-diagonale, diagonale et sur-diagonale). Cette structure se prête naturellement à une représentation creuse.

Stockage CSR et CSC

Le format CSR (*Compressed Sparse Row*) est défini par trois tableaux : `values` (coefficients non nuls), `col_ind` (indices de colonnes) et `row_ptr` (début de chaque ligne dans `values`). De manière duale, le format CSC repose sur `values`, `row_ind` et `col_ptr`. Les deux formats permettent un accès efficace aux éléments non nuls et sont adaptés aux produits matrice–vecteur.

Produit matrice–vecteur

Des fonctions de produit matrice–vecteur en CSR/CSC peuvent remplacer l'appel BLAS utilisé dans le cas bande.

Richardson avec formats creux

L'algorithme de Richardson est alors adapté en ne changeant que l'opération Ax ; le calcul du résidu, le critère d'arrêt et la mise à jour de la solution restent inchangés. Les historiques de convergence obtenus coïncident avec ceux du format bande, ce qui confirme que la convergence dépend de l'opérateur linéaire et non du format de stockage.

6 Conclusion

Ce TP met en évidence la chaîne complète de résolution d'un problème elliptique 1D : modélisation, discrétisation, et résolution du système linéaire associé. Les méthodes directes basées sur BLAS/LAPACK fournissent une solution robuste et efficace, tandis que les méthodes itératives offrent une alternative légère mais dont la convergence dépend fortement de la méthode (Gauss–Seidel étant nettement plus performant que Jacobi/Richardson pour ce cas). Enfin, l'introduction de formats creux (CSR/CSC) permet de généraliser le code tout en conservant les propriétés numériques observées.