UNIVERSITÉ DE VERSAILLES SAINT QUENTIN EN YVELINES

MASTER 1 :

Calcul Haute Performance, Simulation

RAPPORT SUR LE TRAVAIL DE CALCUL NUMÉRIQUE (TD/TP5)

Elaboré par :
Chabane Khaled

Année universitaire : 2023 / 2024

Table des matières

Chapitre I : Introduction

I.1. Introduction

Ce travail est structuré en plusieurs parties, chaque partie correspond à un aspect spécifique de la résolution numérique de l'équation de la chaleur. Premièrement, on établit un cas de test en discrétisant l'équation à résoudre par la méthode des différences finies centrées d'ordre 2. Ensuite, nous mettons en place l'environnement de développement en langage C, et on intègre les bibliothèques BLAS et LAPACK.

Ensuite, nous allons aborder les méthodes directes, en mettant l'accent sur le stockage bande des matrices tridiagonales. Les fonctions de factorisation LU et les résolutions de systèmes linéaires par des méthodes directes sont implémentées et évaluées en termes de performances.

Dans la troisième partie, nous allons travailler sur les méthodes itératives, telles que Richardson, Jacobi et Gauss-Seidel, en mettant en lumière l'implémentation de ces algorithmes avec des matrices tridiagonales. Nous examinons aussi la convergence de ces méthodes.

Enfin, nous étendons notre bibliothèque pour supporter d'autres formats de stockage, comme CSR (Compressed Sparse Row) et CSC (Compressed Sparse Column). Cette extension vise à améliorer la flexibilité de notre approche en prenant en compte des structures de matrices creuses.

I.2. Mise en Place de l'Environnement de Développement

I.2.1. Création du projet avec Git

On a créé un projet pour le travail en cours, et on l'a versionné avec Git. On a créé dépôt public.

I.2.2. Vérification de l'installation des bibliothèques CBLAS et CLAPACK

On a vérifié la disponibilité des bibliothèques CBLAS et CLAPACK. On a inspecté les fichiers nécessaires, situés généralement dans /usr/local/lib et /usr/local/include, ont été inspectés. Si nécessaire, les bibliothèques ont été installées via la commande apt-get sous Ubuntu (apt-get install libblas-dev liblapack-dev).

I.2.3. Écriture du makefile et du code de test

On a créé un makefile pour automatiser le processus de compilation et faciliter la gestion du projet. Ce makefile assure la compilation du code source (en langage c), avec l'intégration des bibliothèques BLAS et LAPACK.

Un fichier de test (testenv.c) a été élaboré pour vérifier le bon fonctionnement de l'environnement de développement. Ce fichier teste la capacité à inclure et à utiliser les fonctions des bibliothèques BLAS et LAPACK.

Par cela on vise à instaurer un environnement robuste et fonctionnel pour la réalisation du travail. Les étapes suivantes consisteront à développer le code du TP en C, en mettant particulièrement l'accent sur l'utilisation efficiente des bibliothèques BLAS et LAPACK pour la résolution des systèmes linéaires.

Chapitre II : Méthode directe et stockage bande

II.1. Référence et utilisation de BLAS/LAPACK

II.1.1. Déclaration et allocation de matrices

-En langage C, la déclaration et l'allocation de matrices pour utiliser BLAS et LAPACK impliquent l'utilisation de pointeurs. Pour déclarer une matrice, on peut utiliser un pointeur double, par exemple "double *A".

-L'allocation de mémoire pour une matrice de taille m × n se fait comme suit : "A = (double *)malloc(m * n * sizeof(double));".

-Il est important de libérer la mémoire, lorsque la matrice n'est plus utilisée : "free(A);".

II.1.2. Signification de LAPACK_COL_MAJOR

LAPACK_COL_MAJOR signifie que les matrices sont stockées en format colonne majeure. Cela signifie que les éléments de la colonne sont stockés consécutivement en mémoire.

II.1.3. Dimension principale (leading dimension)

La dimension principale (ld) est le nombre d'éléments entre deux éléments successifs de la même colonne. Elle est souvent utilisée pour représenter la largeur de la mémoire allouée à une matrice. Par exemple, si on a une matrice A de dimensions m × n et que la mémoire est allouée en format COL MAJOR, ld sera égal à m.

II.1.4. Fonction dgbmv:

La fonction dgbmv effectue une multiplication matrice-vecteur avec une matrice stockée en format bande général (GB). Elle prend en compte des matrices tridiagonales, pentadiagonales..., et effectue le produit matrice-vecteur avec un vecteur x.

II.1.5. Fonction dgbtrf

La fonction dgbtrf réalise la factorisation LU d'une matrice générale stockée en format bande.

II.1.6. Fonction dgbtrs

La fonction dgbtrs résout un système linéaire pour une matrice générale stockée en format bande.

II.1.7. Fonction dgbsv

La fonction dgbsv est une fonction de LAPACK qui résout un système linéaire avec une matrice générale stockée en format bande.

II.1.8. Calcul de la norme du résidu relatif avec BLAS

Pour calculer la norme du résidu relatif avec des appels BLAS, on peut utiliser la fonction dnrm2 de BLAS.

II.2. Stockage GB et Appel à DGBMV

Dans cette section, on a abordé le stockage de la matrice de poisson_1D en formatGB (General Band) en privilégiant la priorité colonne. Ensuite, on a utilisé la fonction de BLAS dgbmv pour effectuer une multiplication matrice-vecteur avec cette matrice particulière. Enfin, on a présenté une méthode de validation pour garantir la précision des résultats obtenus.

II.2.1. Stockage GB en Priorité Colonne pour Poisson 1D

Pour représenter la matrice de poisson_1D dans un format GB avec priorité colonne, les éléments non nuls sont disposés dans un tableau à deux dimensions de manière compacte. Les colonnes de la matrice sont stockées dans les colonnes correspondantes du tableau, et les diagonales sont stockées dans les lignes. Les éléments non nuls sont placés en accord avec la priorité colonne, suivant le schéma GB.

II.2.2. Utilisation de la Fonction DGBMV

La fonction BLAS dgbmv est utilisée pour effectuer le produit matrice-vecteur avec la matrice de poisson_1D stockée en GB. Cette fonction permet de tirer parti de la structure spécifique de la matrice tridiagonale stockée en GB pour optimiser les calculs.

II.2.3. Méthode de Validation

La validation des résultats obtenus à l'aide de dgbmv a été réalisée en comparant les résultats avec une solution analytique connue pour la matrice de poisson_1D. En calculant l'erreur entre la solution obtenue par dgbmv et la solution analytique, on peut évaluer la précision de l'implémentation.

II.3. DGBTRF, DGBTRS, DGBSV

II.3.1. Utilisation de DGBTRF, DGBTRS, et DGBSV

La fonction dgbtrf est employée pour effectuer la factorisation LU de la matrice tridiagonale stockée en GB. Cette factorisation LU est utilisée pour résoudre efficacement le système linéaire à l'aide des fonctions dgbtrs et dgbsv.

II.3.2. Évaluation des Performances

Les performances de ces méthodes directes sont évaluées en mesurant le temps d'exécution en fonction de la taille de la matrice. Des courbes de performances sont présentées pour illustrer la complexité en temps de ces méthodes.

LU pour les matrices tridiagonales :

La factorisation LU est une méthode de décomposition d'une matrice en deux matrices, une inférieure (L) et une supérieure (U). Cette décomposition permet de résoudre plus efficacement des systèmes linéaires et d'inverser la matrice.

II.3. Implémentation de la factorisation LU pour les matrices tridiagonales

Dans cette section, on a réalisé l'implémentation de la factorisation LU spécifiquement adaptée aux matrices tridiagonales et stockées au format GB. L'objectif principal était de tirer parti des fonctionnalités offertes par les bibliothèques BLAS et LAPACK pour assurer une efficacité et une précision optimales.

II.3.1. Implémentation en C de la factorisation LU

On a élaboré un code en C qui utilise les fonctions LAPACK pour effectuer la factorisation LU.

II.3.2. Méthode de validation

Afin de valider notre implémentation, on a mis en place une procédure de validation, qui génère une matrice tridiagonale aléatoire, applique la factorisation LU, puis reconstruit la matrice originale à partir des matrices L et U. Cette reconstruction permet ensuite une comparaison avec la matrice d'origine pour vérifier l'exactitude de la factorisation.

Chapitre III : Méthode de résolution itérative

III.1. Implémentation de la méthode itérative de Richardson

L'étape suivante consiste à mettre en œuvre l'algorithme de Richardson pour résoudre l'équation de la chaleur de manière itérative. Cette approche, basée sur le format GB pour la représentation des matrices tridiagonales, a été réalisée, en exploitant les fonctionnalités des bibliothèques BLAS et LAPACK.

III.1.1. Implémentation en C de l'algorithme de Richardson

Pour cette section on a implémenter l'algorithme de Richardson.

III.1.2. Calcul de l'erreur et analyse de la convergence

Afin d'évaluer l'efficacité de la méthode itérative de Richardson, on a mis en place un mécanisme pour calculer l'erreur par rapport à la solution analytique à chaque itération. L'erreur est déterminée en mesurant la norme euclidienne du vecteur des résidus.

L'analyse de la convergence a été effectuée en générant un graphique de l'historique des erreurs au cours des itérations. Cette visualisation offre un aperçu clair de la convergence de la méthode itérative.

III.2. Implémentation de la méthode itérative de Jacobi

Dans cette section, on a implémenté la méthode itérative de Jacobi en utilisant le format GB pour représenter les matrices tridiagonales. L'objectif est de résoudre l'équation de la chaleur de manière itérative tout en exploitant les fonctionnalités des bibliothèques BLAS et LAPACK.

III.2.1. Implémentation en C de la méthode de Jacobi

Pour cette section nous avons implémenter l'algorithme de Jacobi.

III.2.2. Calcul de l'erreur et analyse de la convergence

On a calculé l'erreur par rapport à la solution analytique à chaque itération de la méthode de Jacobi. L'erreur est mesurée en utilisant la norme euclidienne du vecteur des résidus.

L'analyse de la convergence a été effectuée en générant un graphique illustrant l'historique des erreurs au cours des itérations. Cette représentation graphique permet de visualiser la convergence de la méthode itérative de Jacobi.

III.3. Implémentation de la méthode itérative de Gauss-Seidel

III.3.1. Implémentation en C de la méthode de Jacobi

On a développé une fonction mettant en œuvre la méthode de Gauss-Seidel pour une matrice tridiagonale au format GB.

III.3.2. Calcul de l'erreur et analyse de la convergence

L'erreur par rapport à la solution analytique a été calculée à chaque itération, et une analyse de la convergence a été réalisée en traçant l'historique des erreurs.

Chapitre IV : Autres formats de stockages

IV.1. Format CSR / CSC

Dans les sections précédentes, nous avons implémenté les méthodes LU, Richardson, Jacobi et Gauss-Seidel pour des matrices tridiagonales au format GB. Cette section propose d'étendre notre bibliothèque pour appliquer ces méthodes à des matrices stockées en format CSR (Compressed Sparse Row) et CSC (Compressed Sparse Column).

IV.1.1. Stockage CSR pour Poisson 1D

Le stockage Compressed Sparse Row (CSR) est un format efficace pour représenter des matrices creuses.

IV.1.2. Stockage CSC pour Poisson 1D

De manière similaire, le format Compressed Sparse Column (CSC) est un moyen efficace de stocker des matrices creuses.

IV.1.3. Fonctions dcsrmv et dcscmv

On a créé les fonctions dcsrmv et dcscmv qui réalisent respectivement le produit matrice-vecteur au format CSR et CSC. Ces fonctions sont essentielles pour la multiplication matricielle dans le contexte de la résolution numérique du problème de la chaleur.

Conclusion

Durant ce travail, nous avons parcouru un ensemble de concepts, méthodes et implémentations pour aborder le problème complexe de la résolution numérique de l'équation de la chaleur en 1D. À travers l'utilisation du langage de programmation C et des bibliothèques BLAS et LAPACK, nous avons exploré différentes facettes de l'analyse numérique.

On a commencé par une étape préliminaire consistant à discrétiser l'équation de la chaleur en utilisant la méthode des différences finies centrées d'ordre 2. Cette discrétisation a conduit à la formation d'un système linéaire, que nous avons résolu à l'aide de méthodes directes et itératives. L'accent a été mis sur l'utilisation efficace des bibliothèques BLAS et LAPACK pour tirer parti de performances optimales.

On a développé un environnement de travail, versionné avec Git, et on a assuré la compatibilité avec les bibliothèques CBLAS et CLAPACK. Cela a créé une base solide pour la mise en œuvre des différentes méthodes de résolution.

Les méthodes directes, telles que la factorisation LU avec stockage bande, ont été examinées en détail. On a également abordé les méthodes itératives telles que Richardson, Jacobi et Gauss-Seidel, en analysant leur convergence et en évaluant leurs performances.

Enfin, pour améliorer la flexibilité de notre bibliothèque, on a étendu notre prise en charge aux formats de stockage CSR et CSC pour les matrices creuses. Cela a permis une manipulation plus efficace des matrices résultantes, offrant des gains de performances dans certains contextes.

Bibliographie

https://www.ceremade.dauphine.fr/~bey/enseignement/Enseignement/All_enseignement_dauphine/Module_analyse_numerique/livre%20analyse%20num%C3%A9rique/fiche_numerique/3.pdf

Matrix storage scheme: http://www.netlib.org/lapack/lug/node121.html

https://maa.org/press/periodicals/loci/joma/iterative-methods-for-solving-iaxi-ibi-gauss-seidel-method

Band Storage: http://www.netlib.org/lapack/lug/node124.html

BLAS Documentation:https://netlib.org/blas/

https://fr.wikipedia.org/wiki/M%C3%A9thode_de_Jacobi

LAPACK Documentation: http://www.netlib.org/lapack

The LAPACKE C Interface to LAPACK: http://www.netlib.org/lapack/lapacke

Cours université Paris Saclay

CLAPACK The Fortran to C version of LAPACK:https://netlib.org/clapack/