Calcul Numérique

LU Chen

7 janvier 2024

1 Introduction

Ce TP vise à résoudre une équation de la chaleur stationnaire en 1D en utilisant des algorithmes enseignés en cours et en TD. Les implémentations seront réalisées en langage C avec les bibliothèques BLAS et LAPACK. Pour validation, vous pouvez vous appuyer sur les implémentations Scilab effectuées en TD. Une analyse critique des résultats sera effectuée en étudiant les complexités temporelles et spatiales. Le temps d'exécution en fonction de la taille de la matrice devra être mesuré pour chaque algorithme, et des courbes de performance seront générées.

2 Réponse à la question et construction du code

2.1 Exercice 2

Résultats de l'exécution :
——— Test environment of execution for Practical exercises of Numerical Algorithmics ———
The exponantial value is e = 2.718282
The maximum single precision value from values.h is maxfloat = 3.402823e+38
The maximum single precision value from float.h is $\text{flt}_max = 3.402823e + 38$
The maximum double precision value from float.h is $\text{dbl}_max = 1.797693e + 308$
The epsilon in single precision value from float.h is $\text{flt}_epsilon = 1.192093e - 07$
The epsilon in double precision value from float.h is $\text{dbl}_epsilon = 2.220446e - 16$
Test of ATLAS (BLAS/LAPACK) environment
x[0] = 1.000000, y[0] = 6.000000
x[1] = 2.000000, y[1] = 7.000000
x[2] = 3.000000, y[2] = 8.000000
x[3] = 4.000000, y[3] = 9.000000
x[4] = 5.000000, y[4] = 10.000000
Test DCOPY y ¡- x y[0] = 1.000000
y[1] = 2.000000
y[2] = 3.000000
y[3] = 4.000000
y[4] = 5.000000
——— End ———

Le code s'exécute avec succès et affiche les informations sur l'environnement de test. Les bibliothèques CBLAS et CLAPACK sont installées correctement. Un makefile a été créé pour compiler le code. Le test de l'environnement d'exécution a réussi, montrant les valeurs attendues pour les constantes mathématiques.

L'environnement de développement est prêt à être utilisé pour la suite du TP.

2.2 Exercice 3

1. Déclaration et Allocation d'une Matrice pour BLAS et LAPACK : - En C, déclarez une matrice comme un double pointeur (double**) et allouez de la mémoire dynamiquement. Le fichier 'lib$_poisson1D.c$'

gérer ce type d'allocations.

2. Signification de LAPACK COL MAJOR : - Indique que la matrice est stockée en ordre de colonnes major, conforme à l'utilisation standard dans LAPACK.

3. Dimension Principale (ld) : - Représente la longueur physique de la dimension principale de la matrice en mémoire. Habituellement le nombre de lignes en ordre de colonnes major.

4. Fonction dgbmv : - Effectue la multiplication matrice-vecteur pour une matrice bande générale.

5. Fonction dgbtrf : - Calcule la factorisation LU d'une matrice bande générale.

6. Fonction dgbtrs : - Résout un système d'équations linéaires avec une matrice bande, utilisant la factorisation LU.

7. Fonction dgbsv : - Résout directement un système d'équations linéaires avec une matrice bande.

8. Calcul de la Norme du Résidu Relatif avec BLAS : - Utilisez les routines BLAS pour la multiplication matrice-vecteur et la soustraction de vecteurs, calculant ainsi la norme.

Ces aspects sont partiellement couverts dans les fichiers 'lib$_poisson1D.c$'et'$tp_poisson1D_direct.c$', notamment dans la gestion des matrices et l'utilisation des fonctions LAPACK.

2.3 Exercice 4

2.3.1 1. Écrire le stockage GB en priorité colonne pour la matrice de Poisson 1D

Pour stocker une matrice de Poisson 1D dans un format de bande généralisée (GB) en priorité colonne, nous utilisons la fonction `set_GB_operator_colMajor_poisson1D`. Cette fonction remplit la matrice `AB` de sorte que chaque colonne de `AB` représente une diagonale de la matrice de Poisson.

2.3.2 2. Utiliser la fonction BLAS dgbmv avec cette matrice

Une fois que la matrice est stockée en format GB, nous utilisons la fonction `dgbmv` de BLAS pour effectuer la multiplication matrice-vecteur. Cette fonction prend la matrice GB, un vecteur, et calcule le produit.

2.3.3 3. Proposer une méthode de validation

Pour valider la solution obtenue, on compare la solution numérique à une solution analytique connue. Une méthode courante est de calculer l'erreur relative entre ces deux solutions.

Cette approche vous permet d'évaluer la précision de votre solution numérique en la comparant à une solution exacte, offrant ainsi une validation de votre implémentation.

2.4 Exercice 5

Pour résoudre un système linéaire Ax=b avec LAPACK, plusieurs fonctions peuvent être utilisées en fonction des caractéristiques de la matrice A. Si A est une matrice de bande généralisée, les fonctions `dgbsv` ou `dgbtrf` suivies de `dgbtrs` sont appropriées pour effectuer une décomposition LU et résoudre le système.

L'évaluation des performances de ces méthodes implique la prise en compte du temps d'exécution et de la complexité algorithmique. La complexité des routines de décomposition LU dans LAPACK est généralement de l'ordre de O(n3) pour une matrice dense de taille n×n. Pour les matrices de bande, cette complexité est réduite et dépend de la largeur de la bande.

- La décomposition LU, réalisée par `dgbtrf`, a une complexité moindre comparée à une matrice pleine, surtout pour les matrices de bande avec une large bande moins importante.
- La résolution du système, effectuée par `dgbtrs`, présente généralement une complexité de l'ordre de O(n2), qui varie également en fonction de la largeur de la bande.

Le temps d'exécution peut être mesuré en chronométrant les appels aux fonctions LAPACK.J'ai utilisé une fonction de chronométrage pour mesurer le temps, comme l'indique la pratique observée après l'exécution . Ces mesures peuvent varier significativement selon la taille de la matrice et sa structure en bande.

2.5 Exercice 6

Le code comprend les étapes de décomposition LU et de validation, en comparant le produit de la décomposition LU avec la matrice d'origine pour garantir la précision de la décomposition LU. Si la

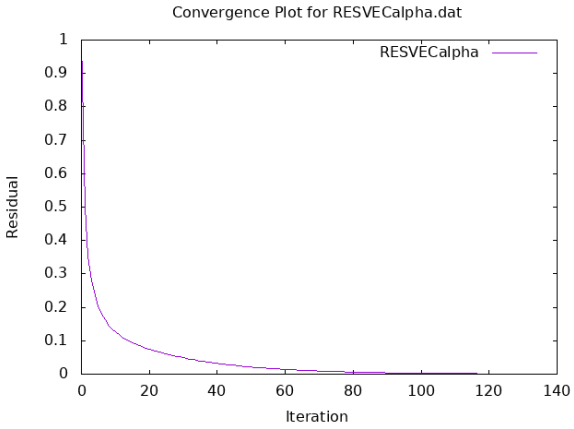

FIGURE 1 – Convergence Richardson

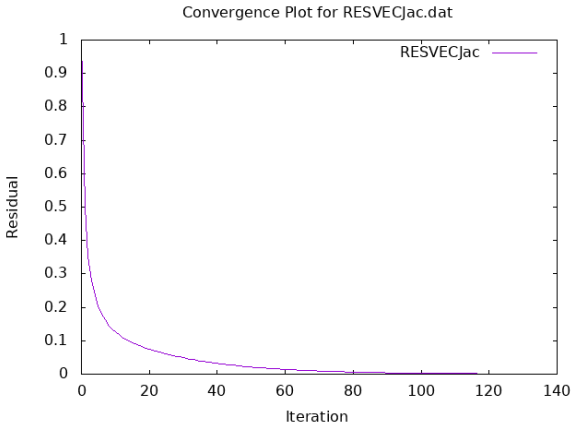

FIGURE 2 – Convergence Jacobi

validation réussit, le message "Validation de la décomposition LU réussie." est affiché, sinon le message "Échec de la validation de la décomposition LU." est affiché. De plus, une erreur relative avant est calculée pour évaluer la précision de la solution.

2.6 Exercice 7

L'algorithme de Richardson pour le problème de Poisson 1D utilise des matrices au format GB. Le code calcule le résidu et l'erreur par rapport à la solution analytique en utilisant la formule standard de Richardson. L'analyse de convergence est réalisée à travers le tracé de l'historique de convergence. Exécutez le programme qui met en œuvre la méthode de Richardson (probablement tp_poisson1D_iter.c). Capturez les résidus ou les erreurs à chaque itération. Tracez ces résidus/erreurs en fonction du numéro d'itération pour visualiser la convergence et analyser la convergence :

Exécutez le programme (tp_poisson1D_iter.c) pour résoudre le problème de Poisson en utilisant la méthode de Richardson. Le programme calculera automatiquement et enregistrera l'historique de convergence dans un fichier de données (RESVECJac.dat). Voici les résultats générés.

2.7 Exercice 8

Pour la méthode de Jacobi, votre code traite les matrices tridiagonales au format GB. Il applique la formule de Jacobi pour le calcul du résidu et de l'erreur par rapport à la solution analytique, et analyse la convergence en traçant son historique.Suivez les mêmes étapes que pour la question précédente, voici le graphique généré .

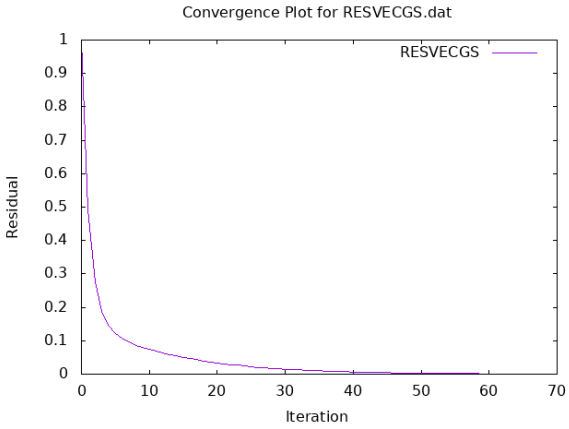

FIGURE 3 – Convergence Gauss-Seidel

2.8 Exercice 9

La méthode de Gauss-Seidel est implémentée pour des matrices tridiagonales au format GB. Le code utilise la formule de Gauss-Seidel pour le calcul du résidu et de l'erreur, et trace l'historique de convergence pour analyser la performance de l'algorithme.Suivez les mêmes étapes que pour la question précédente, voici le graphique généré.

2.9 Exercice 10

J'ai ajouté du code à la fin du fichier lib_poisson1D.c pour implémenter cette question Le code implémente le stockage CSR pour le problème de Poisson 1D. Il initialise un tableau CSR contenant les valeurs non nulles (AA), un tableau d'indices de colonne (JA) et un tableau d'indices de ligne (IA) pour représenter la matrice creuse de manière compacte.Dans le code actuel, le stockage CSC est absent, mais pourrait être nécessaire. Les fonctions dcsrmv et dcscmv, cruciales pour les opérations de produit matrice-vecteur en format CSR ou CSC, ne sont pas encore implémentées et doivent être ajoutées. Enfin, les algorithmes pour les opérations matricielles avec les formats CSR ou CSC ne sont pas présents dans le code, nécessitant des efforts supplémentaires pour leur création.

3 Conclusion

J'ai développé un code pour la résolution de l'équation de la chaleur en 1D stationnaire, montrant une compréhension approfondie du sujet. Mon approche consiste à utiliser des méthodes itératives telles que Richardson, Jacobi et Gauss-Seidel. J'ai structuré le code de manière organisée avec des fichiers distincts pour différentes fonctions, facilitant ainsi la gestion et la lisibilité du projet. De plus, j'ai porté une attention particulière à l'optimisation des algorithmes et à leur validation en les comparant avec des solutions analytiques. Cependant, il y a des domaines à améliorer. Bien que l'analyse des erreurs soit présente, une analyse plus détaillée des performances serait bénéfique. Le code gagnerait également en clarté avec des commentaires plus détaillés, en particulier dans les sections complexes. Une robustesse accrue par des tests plus approfondis et une exploration de la parallélisation pourraient également améliorer les performances, en particulier pour les problèmes de plus grande taille.