Rapport Calcul numérique

Nom : DRISS

Prénom : Ryma

M1 CHPS

Introduction :

Ce TP se concentre sur la résolution numérique de l'équation de la chaleur en 1D. Nous discrétisons le problème pour former un système linéaire. Pour réaliser ce TP on a fait appel aux bibliothèques BLAS et LAPACK. Cette étude offre une opportunité d'appliquer les concepts théoriques et d'approfondir la compréhension de la résolution numérique des équations dérivées partielles.

Mise en place de l'environnement de développement :

1. **Création du projet et du dépôt GIT :** Un dépôt Git public a été créé sur GitHub pour le projet.
2. **Installation des bibliothèques CBLAS et LAPACK :** Les bibliothèques nécessaires ont été installées à l'aide de la commande et vérifiées dans les répertoires spécifiés.
3. **Makefile et testenv.c :** afin d'évaluer l'environnement de travail, un makefile a été créé pour automatiser le processus de compilation et de construction du projet et un code de test a été développé pour évaluer l'environnement de travail et tester les fonctionnalités des bibliothèques installées.
   ➔ Utilisation d'un dockerfile afin d'installer les bibliothèques nécessaires BLAS et LAPACK.

Afin de résoudre le problème de Poisson1d, des fichiers lib_poisson1D.c, lib_poisson1D_richardson.c a été créé, il contient toutes les fonctions développées pour résoudre ce problème.

Méthode directe et stockage bande :

Dans ce TP on réalise l'étude sur les matrices tridiagonales, on a utilisé la méthode de stockage par bande qui offre une représentation compacte pour une matrice de dimension m*n avec kl sous diagonales et ku sur diagonales dans un tableau à deux dimensions avec kl + ku + 1 ligne et n colonnes. L'accès à un élément a(i,j) se fait via la position AB(ku + 1 + i − j, j) Pour cela on a utilisé LAPACK et BLAS qui nous offre la possibilité de faire le stockage par bande.

Référence et utilisation de BLAS/LAPACK :

- ## BLAS et LAPACK :

  « Basic Linear Algebra » et « Linear Algebra Package », sont utilisés pour effectuer des opérations de base en algèbre linéaire (opérations sur des vecteurs et des matrices). Ces bibliothèques sont largement utilisées dans le domaine de la programmation scientifique et du calcul numérique, fournissant des outils pour des opérations sur les matrices et les vecteurs.

1. En C, pour utiliser BLAS et LAPACK, on déclare et alloue une matrice à l'aide de pointeur en utilisant des tableaux dynamiques en utilisant malloc, puis on libère la mémoire avec free.

2. La constante LAPACK_COL_MAJOR signifie l'ordre de stockage des éléments dans une matrice utilisée par LAPACK. Lorsque LAPACK_COL_MAJOR est défini, les éléments d'une matrice sont stockés en colonnes.

3. La dimension principale <ld> représente la dimension entre deux éléments d'une même colonne ou ligne, selon le stockage dans la mémoire. Elle est cruciale dans BLAS et LAPACK pour interpréter correctement les données stockées. L'utilisation de <ld> facilite l'accès efficace aux éléments de la matrice.

4. La fonction **<dgbmv>** de BLAS effectue une **multiplication matrice-vecteur** où la matrice est stockée de manière compacte en utilisant **le format de stockage par bande.** Elle implémente une méthode pour la multiplication de matrices bandes avec un vecteur.

5. La fonction **<dgbtrf>** de LAPACK effectue **la factorisation LU** d'une matrice stockée en format de stockage par bande. Elle décompose la matrice en un produit de deux matrices, une matrice triangulaire inférieure (L) et une matrice triangulaire supérieure (U).

6. La fonction **<dgbtrs>** de LAPACK résout un système linéaire Ax=B où A est une matrice bande. Elle utilise la factorisation LU calculée par la fonction <dgbtrf> pour résoudre un système linéaire.

7. La fonction **<dgbsv>** de LAPACK résout un système linéaire, A est une matrice bande. Elle utilise la factorisation LU avec pivotage partiel, calculée par la fonction dgbtrf, suivie de la résolution du système à l'aide de la fonction <dgbtrs>.

8. Pour calculer la norme du résidu relatif d'un système linéaire résolu avec des appels BLAS, on utilise la fonction **dnrm2** qui est une fonction BLAS qui calcule la norme euclidienne d'un vecteur.

Stockage GB et appel à DGBMV :

1. La fonction **« set_GB_operator_colMajor_poisson1D »** remplit la matrice AB avec une représentation compacte en format de stockage bande, le parcourt de la matrice se fait par colonne.

    - **if (i > 0) {AB[*kv + 1 - 1 + i * (*lab)] = -1.0;}**
    - **if (i < n - 1) {AB[*kv + 1 + 1 + i * (*lab)] = -1.0;}**
    - **AB[*kv + 1 + i * (*lab)] = 2.0;**

2. La fonction **« set_dense_RHS_DBC_1D »** est utilisé pour créer le vecteur b (RHS) d'un système linéaire en tenant compte des conditions limites de Dirichlet BC0 et BC1.

3. La fonction **« set_analytical_solution_DBC_1D »** calcule la fonction analytique d'une équation en une dimension avec des conditions aux limite $T_1\ et\ T_0$ suivante :

$$T(x)= T_0 + x * (T_1 - T_0)$$

EX_SOL : est le vecteur représentant la solution analytique.
DELTA_T : c'est la différence entre les condition BC1 – BC0.

4. Utilisation de la fonction BLAS : **cblas_dgbmv()**.

5. Proposer une méthode de validation : utiliser la matrice générée, vérifier que les résultats des calculs sont cohérents avec les attentes pour la résolution de l'équation de Poisson 1D, affichez la matrice générée sous une forme lisible. On peut utiliser la bibliothèque LAPACK et les fonctions BLAS.

DGBTRF, DGBTRS, DGBSV :

- Pour résoudre un système linéaire à l'aide de LAPACK, on utilise la fonction LAPACKE_dgesv, qui résout un système linéaire pour une matrice stockée en format de bande en utilisant la factorisation LU.

- L'évaluation des performances et complexité permettront de mieux comprendre l'efficacité des méthodes utilisées pour la résolution de systèmes linéaires. Les méthodes directes de LAPACK sont souvent très efficaces pour les matrices de petites tailles (la méthode de factorisation LU à une complexité de O(n^3).

- Certaines fonctions de LAPACK (et de BLAS) prennent en charge la décomposition en blocs, ce qui peut améliorer les performances.

Méthode de résolution itérative :

Dans cette partie, nous abordons la résolution itérative de l'équation de la chaleur, se concentrant sur les algorithmes adaptés aux matrices tridiagonales. Elle commence par une étude théorique suivie d'une modélisation avec Scilab. Ensuite, elle développe un code en C basé sur l'utilisation des BLAS pour implémenter ces algorithmes.

Implémentation C – Richardson :

➢ Implémentation des fonctions demandées :

- **Eig_poisson1D :** cette fonction calcule les valeurs propres en utilisant la formule suivante :

$$\lambda = 2 – (2*\cos(\pi*k /n+1))$$

- **Eigmax_poisson1D :** cette fonction retourne le maximum des valeurs propres.
- **Eigmin_poisson1D :** cette fonction retourne le minimum des valeurs propres.

- **Richardson_alpha :** elle calcule alpha optimal en utilisant la formule suivante :

$$\alpha = 2/(\lambda_{MAX} + \lambda_{MIN})$$

- **La méthode de Richardson :**

1. La fonction **richardson_alpha** implémente la méthode de Richardson pour résoudre un système linéaire **A*x=b** selon la formule suivante :

$$\mathbf{X^{k+1}} = X^k + M^{-1}(b - A * X^K)$$

- A est une matrice stockée en format bande.
- b c'est le vecteur RHS.
- Calcule le R=b - A*x pour obtenir le vecteur resvec[].
- Calcule le résidu R=b-A*x.
- Kl : le nombre d'éléments en dessous de la diagonale.
- Ku : le nombre d'éléments au-dessus de la diagonale.
- Lab : c'est la longueur du tableau qui stocke la matrice bande.

2. Pour tracer l'historique de convergence on utilise les résultats obtenus par **resvec** en utilisant gnuplot**.**

- **Méthode de jacobi :**

Est une méthode itérative utilisé pour résoudre un système linéaire, représenté par la formule suivante :

$$\mathbf{X^{k+1}} = X^k + D^{-1}(b - A * X^K)$$

➔ La fonction **extract_MB_jacobi_tridiag** extrait une sous matrice tridiagonale d'une matrice stockée en bande et la stocke dans une matrice **MB**. La sous matrice tridiagonale est extraite à partir de **AB.**

- **Méthode de gauss Seidel :**

Est une méthode itérative utilisé pour résoudre un système linéaire, représenté par la formule suivante :

$$\mathbf{X^{k+1}} = X^k + (D - E)^{-1}(b - A * X^K)$$

➔ La fonction **extract_MB_gauss_seidel_tridiag** extrait une sous matrice tridiagonale de **AB** pour l'utiliser avec la méthode de gauss Seidel pour résoudre un système linéaire**.**

➔ La fonction **richardson_MB** implémente la méthode de Richardson itérative pour résoudre un système linéaire **A*x = b**, ou **A** est une matrice tridiagonale stockée en format bande **MB**, b le vecteur RHS, **x** est le vecteur à trouver.

Autres formats de stockage :

Dans les parties précédentes, on a mis en œuvre les méthodes LU, Richardson, Jacobi, Gauss Seidel pour des matrices tridiagonales stockées en format bande (GB). Maintenant, on envisage pour appliquer d'autres formats de stockage qui sont utilisés pour représenter des matrices creuses :

- **Stockage CSR, CSC pour poisson1D :** sont des formats de stockage pour la représentation des matrices creuses, ce qui est utilisé pour la résolution d'équations telle que poisson1D. Permettent de minimiser l'utilisation de la mémoire en stockant uniquement les éléments non nuls.
    - ➔ **CSR :** Les éléments de la matrice sont parcourus ligne par ligne.
    - ➔ **CSC :** Les éléments de la matrice sont parcourus colonne par colonne.
- Les fonctions **dcsrmv** et **dcscmv** effectuent le produit matrice vecteur selon le format CSC (colonne) ou CSR (ligne).