TD/TP 5 de Calcul numérique
Méthodes directes et itératives pour résoudre l'équation de la chaleur 1D

T. Dufaud (thomas.dufaud@uvsq.fr), J. Gurhem (jgurhem@aneo.fr)

—M1 CHPS—

Résolution de l'équation de la chaleur en 1D stationnaire

L'objectif de ce TD/TP est d'appliquer un ensemble d'algorithmes étudiés en cours et TD pour la résolution d'un système linéaire obtenu par discrétisation par la méthode des différences finies de l'équation de la chaleur 1D stationnaire. Les implémentations seront faites en C avec BLAS et LAPACK. Un code à trou est proposé afin de faciliter l'implémentation des méthodes étudiées. Afin de valider vous pouvez vous reposer sur les implémentations Scilab réalisées en TD. Une analyse critique de vos résultats est demandée à partir de l'étude des complexités en temps et en espace. Pour chaque algorithme, le temps d'exécution en fonction de la taille de la matrice devra être mesuré et des courbes de performances devront être présentées. Un dépôt doit être fourni avant le début de la troisième séance durant laquelle une correction de la partie 3 sera fournie pour partir sur une base saine pour la partie 4. Un rapport est à fournir sur les parties mentionnées au retour des vancances de Noel.

Langage C, Bibliothèque BLAS et LAPACK, Gnuplot

---

1 Plan des séances

- **Séance 1 :**
  - Rappels sur Docker.
  - Mise en place de l'environnement de développement C + BLAS/LAPACK avec Docker.
  - Travail préliminaire : Etablissement d'un cas de test.
  - DS 3.
- **Séance 2 :**
  - Méthode directe et stockage bande.
  - DS4.
- **Séance 3 :**
  - Rendu de la partie 1 avant de commencer la séance (commit et push sur votre dépôt).
  - Méthode de résolution itérative : Richardson, Jacobi, Gauss-Seidel.

2 Travail préliminaire : Etablissement d'un cas de test

Soit à résoudre l'équation de la chaleur dans un milieu immobile linéaire homogène avec terme source et isotrope :

$$\begin{cases} -k\frac{\partial^2 T}{\partial x^2} = g,\ x \in]0,1[\\ T(0) = T_0 \\ T(1) = T_1 \end{cases} \tag{1}$$

Où g est un terme source, $k > 0$ le coefficient de conductivité thermique, et $T_0 < T_1$ les températures au bord du domaine considéré.

On se propose de résoudre cette équation par une méthode de différence finie centrée d'ordre 2. On discrétise le domaine 1D selon $n+2$ noeuds x_i, $i = 0, 1, 2, ...n+1$, espacés d'un pas h constant.

En chaque noeud l'équation discrète s'écrit :

$$-k\left(\frac{\partial^2 T}{\partial x^2}\right)_i = g_i \tag{2}$$

Pour la suite du TP on notera ce système :

$$Au = f,\ A \in \mathbb{R}^{n\times n},\ u,\ f \in \mathbb{R}^n \tag{3}$$

Dans la suite du TP on considère qu'il n'y a pas de source de chaleur, *i.e.* $g = 0$. La solution analytique de cette équation est :

$$T(x) = T_0 + x(T_1 - T_0) \tag{4}$$

▸ **Exercice 1.** ***Fait en TD***

1. *Approximer la dérivée seconde de T au moyen d'un schéma centré d'ordre 2.*
2. *Écrire le système linéaire de dimension n correspondant au problème 1.*
3. *Présentez le problème à résoudre et sa discrétisation en introduction du rapport.*

Afin de calculer efficacement la solution de ce problème nous souhaitons développer un code en langage C faisant appel aux bibliothèques de calcul BLAS et LAPACK.

▸ **Exercice 2.** ***Mise en place de l'environnement de développement (C + BLAS/LAPACK)***

1. *Créez un projet et le versionner avec git. Créer un dépôt public sur la plateforme de votre choix.*

2. *Vérifiez l'installation et la présence des bibliothèques CBLAS et CLAPACK (dans /usr/local/lib et /usr/local/include) (sous Ubuntu : apt-get install libblas-dev liblapack-dev). Le code fourni peut aussi être compilé utilisant Docker (voir le fichier docker/Dockerfile) depuis la racine du projet avec la commande :* docker build -f docker/Dockerfile –progress plain -t tp-cn:latest .

3. *Écrivez un makefile et un code de test de votre environnement de travail (on pourra utiliser le fichier* testenv.c*)*

Dans la suite de ce TP, vous étudierez et complèterez le code C associé à ce TP faisant appel à BLAS et LAPACK et téléchargeable sur moodle.

Dans la suite du TP il est conseillé de créer un fichier lib_poisson1D.c qui contient toutes les fonctions développés pour résoudre le problème de Poisson1D. Et de créer des fichiers séparés pour les codes de résolution.

3 Méthode directe et stockage bande

On considère dans notre étude des matrices tridiagonales. Une méthode de stockage efficace pour ce type de matrice est le stockage par bande.

Lors de l'utilisation de LAPACK et BLAS, différents stockages bande sont proposés. Nous étudions dans un premier temps le stockage General Band.

Une matrice de dimension $m \times n$ avec kl sous-diagonales et ku sur-diagonales peut être stockée de manière compacte dans un tableau à deux dimensions avec $kl + ku + 1$ lignes et n colonnes. Les colonnes de la matrice sont stockées dans les colonnes correspondantes du tableau, et les diagonales de la matrice sont stockées dans les lignes du tableau. Ce stockage ne doit être utilisé que si $kl, ku \ll \min(m, n)$. Dans LAPACK les matrices ayant un stockage de ce type ont un nom se teminant par B. Il suit que l'élément a_ij de la matrice A est stockée dans $AB(ku + 1 + i - j,\ j)$. Par exemple, pour $m = 5$, $n = 5$, $kl = 2$, $ku = 1$: Soit,

$$A = \begin{pmatrix} a_{11} & a_{12} & & & \\ a_{21} & a_{22} & a_{23} & & \\ a_{31} & a_{32} & a_{33} & a_{34} & \\ & a_{42} & a_{43} & a_{44} & a_{45} \\ & & a_{53} & a_{54} & a_{55} \end{pmatrix}.$$

sont stockage GB dans le tablea AB est tel que :

$$AB = \begin{pmatrix} * & a_{12} & a_{23} & a_{34} & a_{45} \\ a_{11} & a_{22} & a_{33} & a_{44} & a_{55} \\ a_{21} & a_{32} & a_{43} & a_{54} & * \\ a_{31} & a_{42} & a_{53} & * & * \end{pmatrix}.$$

Les éléments $*$ doivent être affectés. On choisira généralement 0.

Dans la suite de ce TP, vous concevez et validez un code C faisant appel à BLAS et LAPACK.

Références :

Matrix storage scheme: `http://www.netlib.org/lapack/lug/node121.html`
Band Storage: `http://www.netlib.org/lapack/lug/node124.html`
BLAS Documentation:`https://netlib.org/blas/`
LAPACK Documentation: `http://www.netlib.org/lapack`
The LAPACKE C Interface to LAPACK: `http://www.netlib.org/lapack/lapacke`
CLAPACK The Fortran to C version of LAPACK:`https://netlib.org/clapack/`

▸ **Exercice 3.** ***Référence et utilisation de BLAS/LAPACK***
A l'aide de la documentation de BLAS et LAPACK répondez aux questions suivantes :

1. *En C, comment doit on déclarer et allouer une matrice pour utiliser BLAS et LAPACK ?*
2. *Quelle est la signification de la constante* ***LAPACK_COL_MAJOR*** *?*
3. *À quoi correspond la dimension principale (leading dimension) généralement notée ld ?*
4. *Que fait la fonction dgbmv ? Quelle méthode implémente-t-elle ?*
5. *Que fait la fonction dgbtrf ? Quelle méthode implémente-t-elle ?*
6. *Que fait la fonction dgbtrs ? Quelle méthode implémente-t-elle ?*
7. *Que fait la fonction dgbsv ? Quelle méthode implémente-t-elle ?*
8. *Comment calculer la norme du résidu relatif avec des appels BLAS ?*

▸ **Exercice 4.** ***Stockage GB et appel à DGBMV***

1. *Écrivez le stockage GB en priorité colonne pour la matrice de Poisson 1D.*
2. *Utilisez la fonction BLAS* dgbmv *avec cette matrice.*
3. *Proposez une méthode de validation.*

▸ **Exercice 5.** ***DGBTRF, DGBTRS, DGBSV (à rendre)***

1. *Résolvez le système linéaire par une méthode directe en faisant appel à LAPACK.*
2. *Évaluez les performances. Que dire de la complexité des méthodes appelées ?*

▸ **Exercice 6.** ***LU pour les matrices tridiagonales (à rendre)***

1. *Implémentez la méthode de factorisation LU pour les matrices tridiagonales avec le format GB.*
2. *Proposez une méthode de validation.*
3. *Évaluez les performances. Comparez. Que dire de la complexité des méthodes appelées ?*

4 Méthode de résolution itérative

Nous souhaitons maintenant résoudre l'équation de la chaleur par une méthode itérative. Dans les exercices suivant nous étudions différents algorithmes et leur conception pour une matricie tridiagonale et plus particulièrement pour ce problème. Dans un premier temps on effectue une étude théorique puis un maquettage avec Scilab. Dans un second temps on développe le code C reposant sur l'appel des BLAS.

▸ **Exercice 7.** ***Implémentation C - Richardson***

1. *Implanter en C l'algorithme de Richardson avec des matrices au format GB.* ***Sauvegardez le residu dans un vecteur.***
2. *Calculer l'erreur par rapport à la solution analytique.*
3. *Analyser la convergence (tracez l'historique de convergence).*

▸ **Exercice 8.** ***Implémentation C - Jacobi***

1. *Écrire une fonction implémentant la méthode de Jacobi pour une matrice tridiagonale avec le format GB.*
2. *Calculer l'erreur par rapport à la solution analytique.*
3. *Analyser la convergence (tracez l'historique de convergence).*

▸ **Exercice 9.** ***Implémentation C -Gauss-Seidel***

1. *Écrire une fonction implémentant la méthode de Gauss-Seidel pour une matrice tridiagonale avec le format GB. On discutera du choix pour le calcul de l'application de $(D-E)^{-1}$ au résidu r^k.*
2. *Calculer l'erreur par rapport à la solution analytique.*
3. *Analyser la convergence (tracez l'historique de convergence).*

5 Autres formats de stockage

Nous avons, dans les sections précédentes, implémenté les méthodes LU, Richardson, Jacobi et Gauss-Seidel pour des matrices tridiagonales au format GB. Nous proposons ici de compléter notre bibliothèque pour appliquer ces résolutions à des matrices stockées CSR ou CSC.

▸ **Exercice 10.** ***Format CSR / CSC***

1. *Écrire le stockage CSR pour Poisson 1D.*
2. *Écrire le stockage CSC pour Poisson 1D.*
3. *Écrire les fonctions dcsrmv et dcscmv qui réalisent respectivement le produit matrice vecteur au format CSR et CSC.*
4. *Écrire les différents algorithmes pour ces formats.*