Travaux Pratiques de calcul numérique n°5

Cours de calcul numérique: T. Dufaud

—Master CHPS - Première Année—

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. Les algorithmes pourront être testé sous Scilab avant d'être implantés en C. 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.

Langage C, Bibliothèque BLAS et LAPACK, Gnuplot

---

1 Travail préliminaire : Etablissement d'un cas de test

On considère 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 :

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

▸ **Exercice 1.**

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.*

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 :

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

2 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 étudierez et complèterez le code C associé à ce TP faisant appel à BLAS et LAPACK et téléchargeable sur moodle.

▸ **Exercice 2.** ***Environnement C + BLAS/LAPACK***
Tout d'abord préparez l'environnement de travail et compiler le code source fournit.

1. *Installez (ou vérifiez) les dépendances nécessaires à la compilation du code (sous Ubuntu : apt-get install libblas-dev liblapack-dev)*
2. *Compilez et exécutez le code de test (make; make run_testenv;)*
3. *Compilez et exécutez le code qui résout l'équation de la chaleur 1D par une méthode directe. (make; make run_tp2poisson1D_direct*

▸ **Exercice 3.** ***Utilisation de BLAS/LAPACK***
A l'aide de la documentation de LAPACK et en étudiant le code tp2_poisson1D_direct.c et ses dépendances 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 dgbsv ? Quelle méthode implémente-t-elle ?*
5. *en respectant le formalisme du code, modifiez tp2_poisson1D_direct.c et lib_poisson1D.c afin d'effectuer les opérations pour un stockage en priorité ligne et non colonne.*
   - *Ecrire le stockage en priorité ligne.*
   - *Faire l'appel à dgbsv.*

▸ **Exercice 4.** ***DGBMV***

1. *Utiliser la fonction BLAS dgbmv pour les deux stockage (ligne ou colonne).*
2. *Proposez une méthode de validation.*

▸ **Exercice 5.** ***LU pour les matrices tridiagonales***

1. *Implémentez la méthode de factorisation LU du TD/TP 2 pour les matrices tridiagonales.*
2. *Proposez une méthode de validation.*

Références :
Matrix storage scheme: http://www.netlib.org/lapack/lug/node121.html#19610
Band Storage: http://www.netlib.org/lapack/lug/node124.html
The LAPACKE C Interface to LAPACK: http://www.netlib.org/lapack/lapacke

3 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 6.** ***Étude des méthodes Jacobi et Gauss-Seidel***

1. *Appliquez la méthode de Jacobi, puis de Gauss-Seidel à une matrice tridiagonale.*
2. *Analysez la complexité de vos algorithmes. Quelles modifications pouvez-vous apportez pour diminuer la complexité.*
3. *Effectuez quelques itérations de ces méthodes pour le problème Poisson 1D, pour* $n = 3$.
4. *Implémentez ces méthodes sous Scilab.*
5. *Analysez la convergence (tracez l'historique de convergence).*

▸ **Exercice 7.** ***Étude du processus itératif de Richardson***
Soit l'itération de Richardson :

$$x^{k+1} = x^k + \alpha(b - Ax^k) \tag{5}$$

avec $\alpha \in \mathbb{R}^{*+}$.

1. *Écrire la matrice d'itération* G_α *de ce processus,* i.e *écrire* $G \in \mathbb{R}^{n\times n}$ *telle que :*

$$x^{k+1} = G_\alpha x^k + \alpha b \tag{6}$$

2. *Écrire un encadrement pour les valeurs propre de* G_α.
3. *Quelles sont les conditions sur* α *pour que la méthode converge ?*
4. *Quelle est la valeur optimal pour* α *?*
5. *Appliquez ces résultats théorique sur le problème de Poisson 1D sous Scilab.*
6. *Tracez les historiques de convergences pour différentes valeurs de* α.

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

1. *Implanter en C l'algorithme de Richardson.* ***Sauvegardez le residu dans un vecteur.***
2. *Ecrire la fonction qui permet de calculer la solution u du système 3. On utilisera le stockage General Band et on validera sur le problème de l'équation de la chaleur 1D.*
3. *Calculer l'erreur par rapport à la solution analytique.*

▸ **Exercice 9.** ***Implémentation C - Jacobi***
Comme nous l'avons fait en TD, appliquer la méthode de Jacobi.

1. *Implanter en C l'algorithme de Jacobi.* ***Sauvegardez le residu dans un vecteur.***

2. *Ecrire la fonction qui permet de calculer la solution u du système 3. On utilisera le stockage General Band et on validera sur le problème de l'équation de la chaleur 1D.*

3. *Calculer l'erreur par rapport à la solution analytique.*