ISIMA1 - Programmation Numérique Vendredi 1er Février 2013
Session de Juin, durée : 2 heures
Documents autorisés : Cours polycopié, notes de cours, TD et TP

Rédigez les 2 problèmes sur feuilles séparées !
Barème indicatif : 10 et 10

Problème 1 : Résolution d'un système linéaire par décomposition LU d'une matrice tridiagonale

Les matrices tridiagonales sont très faciles à factoriser par la méthode de Gauss.
Le système $A\,x = f$ est écrit sous la forme générale ($a_1 = c_n = 0$ par convention)

$$\begin{pmatrix} b_1 & c_1 & 0 & 0 & . & 0 & 0 & 0 \\ a_2 & b_2 & c_2 & 0 & . & 0 & 0 & 0 \\ . & . & . & . & . & . & . & . \\ 0 & 0 & 0 & 0 & . & a_{n-1} & b_{n-1} & c_{n-1} \\ 0 & 0 & 0 & 0 & . & 0 & a_n & b_n \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ . \\ x_{n-1} \\ x_n \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ . \\ f_{n-1} \\ f_n \end{pmatrix}$$

la matrice peut se décomposer sous la forme $L\,U$

$$L = \begin{pmatrix} b_1^* & 0 & 0 & 0 & . & 0 & 0 & 0 \\ a_2 & b_2^* & 0 & 0 & . & 0 & 0 & 0 \\ . & . & . & . & . & . & . & . \\ 0 & 0 & 0 & 0 & . & a_{n-1} & b_{n-1}^* & 0 \\ 0 & 0 & 0 & 0 & . & 0 & a_n & b_n^* \end{pmatrix},\quad U = \begin{pmatrix} 1 & c_1^* & 0 & 0 & . & 0 & 0 & 0 \\ 0 & 1 & c_2^* & 0 & . & 0 & 0 & 0 \\ . & . & . & . & . & . & . & . \\ 0 & 0 & 0 & 0 & . & 0 & 1 & c_{n-1}^* \\ 0 & 0 & 0 & 0 & . & 0 & 0 & 1 \end{pmatrix}$$

En identifiant les coefficients, nous obtenons les récurrences suivantes pour le calcul des coefficients b^* et c^* :

$$\begin{cases} b_1^* = b_1 \\ c_1^* = c_1 / b_1 \end{cases},\quad \begin{cases} b_k^* = b_k - a_k . c_{k-1}^* \\ c_k^* = c_k / b_k^* \end{cases},\quad k = 2, \dots, n$$

Les coefficients b_k^* sont supposés non nuls.
Le système $A\,x = f$ peut être résolu en deux étapes : $L\,y = f$ et $y = U\,x$.

$$L\,y = f : \begin{cases} y_1 = f_1 / b_1^* \\ y_k = (f_k - a_k . y_{k-1}) / b_k^*, \quad k = 2, \dots, n \end{cases},\quad U\,x = y : \begin{cases} x_n = y_n \\ x_k = y_k - c_k^* . x_{k+1}, \quad k = n-1, \dots, 1 \end{cases}$$

Une matrice tridiagonale sera représentée par trois vecteurs (a, b et c).

1- En considérant la matrice tridiagonale et le vecteur f sont connus, écrire une subroutine ***calc_LU*** calculant les matrices U et L (b^* et c^*).
2- En considérant les matrices U et L déjà calculées, écrire une subroutine qui calcule le vecteur x.
3- Ecrire le programme principal qui lit la matrice tridiagonale (3 vecteurs) et le vecteur f dans un fichier, appelle les deux subroutines définies précédemment et écrit le vecteur x dans un autre fichier.

Remarque : les subroutines ***calc_LU*** et ***calc_x*** seront placées dans le même module ***mod_LU***.

Suite du sujet au verso TSVP ⟶

Rédigez sur feuilles séparées de la première partie

Problème 2 - Algorithme Broyden-Fletcher-Goldfarb-Shanno (**BFGS**)

C'est une méthode de descente pour des problèmes de minimisation non-linéaire sans contrainte.
Pour $f : \mathbb{R}^n \to \mathbb{R}$ trouver une valeur approchée d'un minimum local x.
La direction de descente s'obtient en résolvant un système linéaire.
Les **sous-programmes** créés, en Fortran90, seront placés dans un **module.** Ils doivent pouvoir fonctionner en **allocation statique ou dynamique.** Aucune information ne leur sera passée en variable commune, utilisez des **paramètres. Optimisez** les calculs.

Algorithme BFGS : x_0, *kmax* et $\varepsilon > 0$ sont des paramètres
$k=0$; $B_0 = I_n$; $d_0 = -\nabla f(x_0)$; *erreur* = *faux*
Tant que *non erreur* et $k < kmax$ et $\|d_k\| \geq \varepsilon$

Recherche : Trouver $t_k > 0$ qui minimise la fonction $\Phi'(t) = f(x_k + t.d_k)$

$x_{k+1} = x_k + t_k d_k$

$y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$

$$B_{k+1} = B_k + \frac{y_k y_k^{\mathrm{T}}}{y_k^{\mathrm{T}} y_k} + \frac{B_k d_k d_k^{\mathrm{T}} B_k}{d_k^{\mathrm{T}} B_k d_k}$$

Nouvelle direction d_{k+1} solution de $B_{k+1} d_{k+1} = -\nabla f(x_{k+1})$ *erreur* = *vrai* si la résolution pose problème

$k = k + 1$

Fin tant que

1- Expliciter le calcul de $v v^{\mathrm{T}}$ et celui de $v^{\mathrm{T}} v$ pour un vecteur v de dimension n. Indiquer l'ordre des calculs dans l'expression $\dfrac{B_k d_k d_k^{\mathrm{T}} B_k}{d_k^{\mathrm{T}} B_k d_k}$ en plaçant des parenthèses.
2- En supposant que $\|d_k\| < \varepsilon$ arrête l'algorithme $(erreur = faux$ et $k < kmax)$ ou pour simplifier $\|d_k\| = 0$ montrer que x_k est très proche de l'optimum cherché.
3- Ecrire une subroutine qui calcule $v v^{\mathrm{T}}$.
4- En supposant la function réelle **Recherche déjà implantée**, expliciter ses paramètres.
5- Quels sont les paramètres de la subroutine **Gradf** qui calcule le gradient de la fonction ?
6- Implanter l'algorithme **BFGS**, la résolution de $Ax = b$ sera implantée en question 7.
7- Implanter la méthode de Gauss avec pivot partiel (résolution de $Ax = b$), suivant l'algorithme :

erreur = *faux*
Pour $j = 1, n-1$

Chercher le pivot p tel que $|A_{pj}| = max(|A_{ij}|, i = j$ à $n)$

Permuter les équations numéro j et p si nécessaire

Calculer équation$_k$ = équation$_k$ - équation$_j$ * A_{kj}/A_{jj} , pour $k = j+1$ à n

avec *erreur* = *vrai* si une erreur de calcul survient

Fin pour
Si (*non erreur*) alors résoudre le système triangulaire **Fin si**

Attention la variable erreur est un **booléen** qui sera déclaré de façon adéquate en Fortran (**integer interdit !**)