© *J.-B. A. K. <jean-baptiste.apoung@math.u-psud.fr>*

Examen seconde session : 16 juin 2016
(Durée 3h00)

Note 1.

Dans la pratique, lors de la résolution du système linéaire $AU = B$, où $A \in \mathcal{M}_{n,n}(\mathbb{R})$ est une matrice carrée et $B \in \mathcal{M}_{n,m}(\mathbb{R})$ une matrice rectangulaire (m traduisant le fait que l'on résout un système linéaire avec m second-membres), on peut être tenté de calculer explicitement A^{-1}.

- Nous nous proposons dans ce sujet de décrire cette construction dans deux cas particuliers (voir **Thème-1** et **Thème-2**), et d'analyser la pertinence de l'utilisation explicite de A^{-1} dans un cas particulier (voir **Thème-3**).
- Il sera demandé d'exprimer les algorithmes obtenus à travers des fonctions écrites en langage de programmation C . **Cependant, l'écriture de ces fonctions en pseudo-code est aussi acceptée.**
- Pour des questions où votre avis est sollicité, il est attendu une réponse rigoureuse n'exhibant aucune ambiguïté.

Thème - 1 *Matrices tridiagonales*

Note 2.

Dans cette partie, on s'intéresse à la résolution dans $\mathbb{R}^n$, d'un système de la forme

$$\mathbf{Ax} = \mathbf{f} \quad où \quad \mathbf{A} = \begin{pmatrix} a_1 & b_1 & 0 & \cdots & 0 & 0 \\ c_2 & a_2 & b_2 & 0 & & 0 \\ 0 & c_3 & a_3 & b_3 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & & \ddots & c_{n-1} & a_{n-1} & b_{n-1} \\ 0 & 0 & \cdots & 0 & c_n & a_n \end{pmatrix} \tag{1}$$

On posera $a = [a_1, \ldots, a_n]^T$, $b = [b_1, \ldots, b_n]^T$, $c = [c_1, \ldots, c_n]^T$, où $c_1 = 0, b_n = 0$. [$(\cdot)^T$ est l'opérateur transposé].

Les questions **Q-1** et **Q-2** sont indépendantes.

Q-1 : Cas particulier où le calcul explicite de A^{-1} est possible

On suppose : $a_i = 2, \forall i = 1, \ldots, n$, et $c_i = -1, \forall i = 2, \ldots, n$, et $b_i = -1, \forall i = 1, \ldots, n-1$

On introduit alors les n vecteurs $(d^{(k)})_{k=1}^n$ de $\mathbb{R}^n$

$$d_j^{(k)} = j \quad \text{pour} \quad 1 \leq j \leq k \qquad \text{et} \quad d_j^{(k)} = 0 \quad \text{pour} \quad k < j \leq n.$$

Q-1-1 : Calculer les produits scalaires $(Ad^{(k)}, d^{(k')})$ pour $1 \leq k, k' \leq n$.

Q-1-2 : En déduire que les $(d^{(k)})_{k=1}^n)$ forment une base de $\mathbb{R}^n$.

Q-1-3 : On écrit la solution du système $Ax = f$ dans la base des $(d^{(k)})_{k=1}^n)$: $x = \sum_{k=1}^{n} \alpha_k d^{(k)}$.

Calculer les coefficients α_k en fonction de f et des $(d^{(k)})_{k=1}^n)$. Commenter.

Q-1-4 : On écrit cette fois x et f dans la base canonique $(e_k)_{k=1}^n$ de $\mathbb{R}^n$: $x = \sum_{k=1}^{n} x_k e_k, \qquad f = \sum_{k=1}^{n} f_k e_k$

Exprimer x_k en fonction des f_k. En déduire l'expression explicite suivante de la matrice A^{-1} :

$$(A^{-1})_{k,j} = j\frac{n+1-k}{n+1} \quad \text{pour} \quad j \leq k. \tag{2}$$

Q-1-5 : Est-il plus économique (en nombre d'opérations, i.e. multiplications et divisions) de résoudre le système linéaire $Ax = f$ ou de calculer x par la formule $x = A^{-1}f$? On suppose A^{-1} connu, c'est-à-dire que les coefficients $(A^{-1})_{k,j}$ sont déjà calculés.

Q-2 : **Retour au cas où $a_i, b_i, c_i, i = 1 \ldots, n$ sont quelconques (Décomposition LU)**

On pose

$$L = \begin{pmatrix} d_1 & 0 & \cdots & \cdots & 0 \\ l_2 & d_2 & \ddots & & \vdots \\ 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & l_{n-1} & d_{n-1} & 0 \\ 0 & \cdots & 0 & l_n & d_n \end{pmatrix} \qquad U = \begin{pmatrix} 1 & r_1 & 0 & \cdots & 0 \\ 0 & 1 & r_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & 1 & r_{n-1} \\ 0 & \cdots & \cdots & 0 & 1 \end{pmatrix}$$

Q-2-1 : Montrer que $LU = A$ si et seulement si

$$\begin{array}{lll} & d_1 = a_1, & d_1\, r_1 = b_1 \\ l_k = c_k, & d_k = -l_k\, r_{k-1} + a_k, & d_k\, r_k = b_k \quad k = 2, \ldots, n-1, \\ l_n = c_n, & d_n = -l_n\, r_{n-1} + a_n. & \end{array}$$

Q-2-2 : En résolvant ces équations d'inconnues $l_k, d_k, r_k, k = 1, \ldots, n$, déduire que l'on peut stocker A à l'aide de trois vecteurs c, a, b, de taille au plus n chacun. Déduire aussi que l'on peut encore stocker dans ces trois vecteurs la décomposition LU de A. Implémenter alors en langage C un algorithme de décomposition LU de la matrice tridiagonale A dont les diagonales principales sont stockées dans c, a, b comme décrit précédemment. Fournir pour cette fin, une fonction de prototype

```
void decompLUTridiag(gsl_vector* c, gsl_vector* a, gsl_vector* b)
```

Q-2-3 : Proposer un algorithme de résolution du système $LUx = f$ et fournir une fonction C

```
void  descenteRemonteeTridiag(const gsl_vector* c, const gsl_vector* a, const gsl_vector* b,
                              const gsl_vector* f,  gsl_vector* x)
```

qui détermine la solution x du système tridiagonal $Ax = f$ dont la factorisation LU de la matrice A est stockée dans a, b, c comme décrit à la question précédente.

Thème - 2 *Matrice symétrique définie positive : algorithme de Gram-Schmidt*

Dans cette partie, A désigne une matrice de $\mathcal{M}_{n,n}(\mathbb{R})$, symétrique définie positive. On sait dans ce cas que l'application $(x, y) \longmapsto (Ax, y)$ définit un produit scalaire sur $\mathbb{R}^n$. On notera $(\cdot, \cdot)_A$ ce produit scalaire et $\| \cdot \|_A$ la norme associée. On désignera toujours par V^* la transposée au sens du produit scalaire euclidien du vecteur ou matrice V.

Q-3 : Soit $\{v_0, \dots v_{n-1}\}$ une famille de n vecteurs orthogonaux dans le produit scalaire $(\cdot,\cdot)_A$. On dit aussi que ces vecteurs sont A-conjugués. On les suppose en plus normés, toujours dans ce produit scalaire. On pose $M = [v_0 \dots v_{n-1}]$ la matrice dont la colonne i est v_i.

Soit x un vecteur quelconque de $\mathbb{R}^n$. Posons α ses composantes dans la base (v_i). On écrit alors

$$x = \sum_{i=0}^{n-1} \alpha_i v_i \equiv M\alpha.$$

Q-3-1 : Montrer que $\alpha_i = (A\,x, v_i) \quad \forall\, 0 \leq i \leq n-1$. En déduire que $\alpha = M^* Ax$, puis que $x = MM^* Ax$.

Q-3-2 : Conclure que MM^* est l'inverse de A.

Q-3-3 : En déduire que si l'on dispose de n vecteurs A-conjugués, le calcul de l'inverse de A est immédiat.

Q-4 : On s'engage à présent à chercher n vecteurs A-conjugués, en orthonormalisant les vecteurs colonnes de A par le procédé d'orthonormalisation de Gram-Schmidt relativement au produit scalaire $(\cdot,\cdot)_A$.

Dans ce qui suit, A_i désignera la i-ème colonne de la matrice A, $i = 0, \dots, n-1$.

Q-4-1 : On pose $v_0 = A_0/\sqrt{(A_0, A_0)_A}$. Vérifier que $\|v_0\|_A = 1$.

Q-4-2 : On suppose construits $v_0, \dots v_{k-1}$, et on désire construire v_k. On écrit $v_k = A_k + \sum_{l=0}^{k-1} \gamma_l^k v_l$.
Montrer que $\gamma_l^k = -(A_k, Av_l) \quad \forall\, 0 \leq l \leq k-1$. On achève la construction de v_k en le normalisant.

Q-4-3 : Écrire une fonction C de prototype

```
gsl_matrix* appliqueGramSchmidt(const gsl_matrix* A)
```

qui retourne la matrice (M) dont les colonnes sont des vecteurs orthonormés pour $(\cdot,\cdot)_A$ et sont obtenues selon le procédé décrit précédemment.

Q-5 : Expliquer comment résoudre un système linéaire $AX = B$ où $B \in \mathcal{M}_{n,m}(\mathbb{R})$.

Q-5-1 : Écrire une fonction C de prototype

```
void inverseGramSchmidt(const gsl_matrix* A, const gsl_matrix* B, gsl_matrix* X)
```

qui résout un système linéaire avec m second membres stockés comme vecteurs colonnes dans la matrice B. Le résultat est stocké dans la matrice à m colonnes X. *Cette fonction fera appel à la fonction* ***appliqueGramSchmidt*** *précédente.*

Thème - 3 *Quantification de l'utilisation explicite de* A^{-1}

Soit $A \in \mathcal{M}_{n,n}(\mathbb{R})$ et $B \in \mathcal{M}_{n,n}(\mathbb{R})$ deux matrices symétriques définies positives telles que $\varrho(A^{-1}B) < 1$. Soit $F \in \mathbb{R}^n$ un vecteur donné et $m \in \mathbb{N}^*$. On se propose de calculer le vecteur U^m le $m+1$-ième terme de la suite $AU^{k+1} = BU^k + F, \quad U^0 = U_0$.
Pour cela on considère les deux algorithmes suivants :

Algorithme 1 (avec résolution répétée)

Initialisation :
U^0 = U_0

Iterations :
Pour k = 0 à m-1
Calculer: V = B U^k + F
Resoudre: A U^{k+1} = V
fin Pour

Algorithme 2 (avec inversion de la matrice)

Initialisation :
C = A^{-1} B
E = A^{-1} F
U^0 = U_0

Iterations :
Pour k = 0 à m-1
Calculer: U^{k+1} = C U^k + E
fin Pour

Q-6 : **Analyse de l'algorithme 1**

Q-6-1 : Quels formats de stockage sont mieux adaptés pour les matrices A et B ?

Q-6-2 : Peut-on avoir des soucis d'espace mémoire pour cet algorithme ?

Q-6-3 : Quelle est la complexité de cet algorithme lorsque les matrices A, B sont pleines ?

Q-7 : **analyse de l'algorithme 2**

Q-7-1 : Justifier la non-nécessité de calculer explicitement l'inverse A^{-1} de la matrice A dans la détermination de C et E.

Q-7-2 : Quelle est la complexité de cet algorithme lorsque les matrices A, B sont pleines ?

Q-7-3 : Sur un exemple simple de matrice de taille 3, montrer que $A^{-1}B$ peut-être pleine alors que A et B sont creuses.

Q-7-4 : Peut-on être confronté à un problème d'espace mémoire lié au stockage dans cet algorithme ?

Q-8 : Conseiller alors sur le choix de l'un de ces algorithmes.
On pourra s' appuyant sur le potentiel problème d'espace mémoire et sur le fait que pour m fixé, on peut se trouver dans les cas $m \gg n$ ou $n \gg m$ (avec $\gg$ signifiant très grand).