GIS3 — Calcul Numérique — Épreuve de mai 2015
Durée 2 heures — Aucun document autorisé

---

1 Petits exercices et questions de cours (8 points)

Question 1 [2 pts]. Calculer une factorisation $L\,U$ de la matrice suivante (tous les nombres apparaissant lors des calculs sont des entiers).

$$A = \begin{pmatrix} 3 & -7 & 1 \\ 12 & -23 & 4 \\ 15 & -45 & 7 \end{pmatrix} .$$

Question 2 [2 pts]. Résoudre le système d'équations linéaires $A\,x = b$ suivant, en appliquant la méthode expliquée en cours. Expliquer en quelques mots la démarche avant de mener les calculs (tous les nombres apparaissant lors des calculs sont des entiers).

$$\begin{pmatrix} 2 & 0 \\ 3 & 2 \end{pmatrix} \begin{pmatrix} -2 & 1 \\ 0 & 5 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} -14 \\ -51 \end{pmatrix} .$$

Question 3 [1 pt]. La factorisation de la question précédente aurait-elle pu être produite par l'un des algorithmes étudiés en cours ? Justifier.

Question 4 [1 pt]. La matrice suivante est-elle orthogonale ? Justifier.

$$A = \begin{pmatrix} \sqrt{2}/2 & \sqrt{2}/2 \\ \sqrt{2}/2 & -\sqrt{2}/2 \end{pmatrix} .$$

Question 5 [2 pts]. On cherche trois réels α, β, γ tels que le plan $z = \alpha\,x + \beta\,y + \gamma$ passe au plus près des points suivants, au sens des moindres carrés :

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 1.0 \\ 1.1 \\ -3.2 \end{pmatrix} , \quad \begin{pmatrix} 1.2 \\ -2.1 \\ 15.1 \end{pmatrix} , \quad \begin{pmatrix} -2.2 \\ -0.3 \\ 4.4 \end{pmatrix} , \quad \begin{pmatrix} 3.1 \\ 0.7 \\ 5.0 \end{pmatrix} .$$

Associer un système d'équations linéaires surdéterminé à cette question. Décrire deux méthodes différentes utilisables pour y répondre. Citer les algorithmes du cours naturellement associés à ces deux méthodes.

2 Problème (12 points)

Les questions qui suivent s'appliquent au document préparatoire ci-joint.

Question 6 [1 pt]. Le sous-programme FORTRAN de la Figure 1 préserve les valeurs propres de la matrice A. Cela implique que la transformation (*) a une certaine propriété. Laquelle ?

Question 7 [2 pts]. Montrer que la transformation (*) a bien la propriété ci-dessus (donner les arguments clefs de la preuve, mais sans rédiger la preuve).

Question 8 [2 pts]. Le sous-programme de la Figure 1 est proche de l'algorithme de mise sous forme de Hessenberg. Quels sont les points communs et les différences des deux algorithmes ?

Question 9 [1 pt]. Pourquoi la boucle de la ligne 8 s'arrête-t-elle en $m-1$, et pas en m ?

Question 10 [1 pt]. La formule donnée en cours pour les matrices de réflexion de Householder est

$$F = I - 2\,\frac{v\,v^T}{v^T\,v}\cdot$$

Pourquoi cette division par $v^T\,v$ n'apparaît-elle pas dans le code FORTRAN ?

2.1 Complexité

On s'intéresse à la complexité du code FORTRAN en fonction de la dimension m de la matrice A. On ne compte que les opérations arithmétiques sur les flottants double précision. L'opération « racine carrée » compte pour 1.

Question 11 [2 pts]. Donner un équivalent asymptotique, sous la forme d'un monôme m^p (pour un certain p) du code FORTRAN. Justifier. Indiquer en particulier quelle(s) partie(s) du code atteignent cette complexité.

2.2 Optimisation

On souhaite maintenant optimiser le code du sous-programme de la Figure 1 pour le cas où la matrice A est tridiagonale.

Question 12 [3 pts]. Décrire les optimisations possibles de façon schématique. Pour chaque optimisation, indiquer :

1. le ou les numéros de ligne concernés, avec éventuellement un mot-clef,
2. l'optimisation proposée ; pour les multiplications de matrices, ne pas hésiter à décrire l'optimisation par un dessin plutôt que par du code FORTRAN ; ne pas chercher non plus à traiter certains cas particuliers, qui se posent pour la première ou la dernière colonne de A.

Si on applique le sous-programme de la Figure 1 sur une matrice A, on obtient une matrice ayant mêmes valeurs propres. Prenons par exemple la matrice :

```
          [2015    -764    1096    432]
          [                            ]
          [2140    -859     896    492]
     A0 = [                            ]
          [  40     -70      35    -60]
          [                            ]
          [-740     242    -688     39]
```

Appliquons 20 fois de suite le sous-programme sur A_0. On obtient la matrice :

```
          [   1035.000000          -362.8603479       -402.5779036     3446.158160]
          [                                                                       ]
          [                -9                                                     ]
          [-0.1687694811 10          254.8776031       -293.0434879     218.2029988]
     A1 = [                                                                       ]
          [               -13                                                     ]
          [0.9800593956 10          -0.1754202540      -164.9152613     93.96121940]
          [                                                                       ]
          [               -16                                                     ]
          [0.3325044274 10       -0.00006727404604    -0.1081712108    105.0376582]
```

Ces deux matrices ont $-165, 1035, 255, 105$ pour valeurs propres.

La construction de l'algorithme de la Figure 1 est inspirée de celle de l'algorithme de Householder pour la factorisation $Q\,R$ (Figure 6.5 du support de cours). En Français, on pourrait le décrire ainsi :

for $k = 1, 2, \ldots, m-1$
  Extraire un vecteur x sur la colonne k de A entre les indices k et m
  Calculer v et F comme dans l'algorithme de Householder
  Appelons Q_k la matrice obtenue en plongeant F dans une matrice identité $m \times m$
  $A := Q_k\, A\, Q_k$ (*)
end do

```
 1       SUBROUTINE EIGEN (M, A, LDA)
 2 * A est une matrice de dimension m x m
 3       INTEGER M, LDA
 4       DOUBLE PRECISION A(LDA,*), V(M), W(M)
 5       DOUBLE PRECISION DNRM2, NORM
 6       INTEGER I, J, K
 7 * Pour chaque colonne k de A (sauf la dernière)
 8       DO K = 1,M-1
 9 * v = A(k:m,k) kème colonne de A, lignes k à m
10 * Pour simplifier les formules qui suivent, v est stocké en V(k:m)
11          CALL DCOPY (M-K+1, A(K,K), 1, V(K), 1)
12 * v = v +/- norm(v,2) * e1
13          NORM = DNRM2 (M-K+1, V(K), 1)
14          V(K) = V(K) + SIGN (NORM, V(K))
15 * rendre v de longueur 1
16          NORM = DNRM2 (M-K+1, V(K), 1)
17          CALL DSCAL (M-K+1, 1D0/NORM, V(K), 1)
18 * w = Transpose(v) . A(k:m,1:m) est un vecteur ligne de dimension m
19          DO J = 1,M
20             W(J) = 0D0
21             DO I = K,M
22                W(J) = W(J) + V(I)*A(I,J)
23             END DO
24          END DO
25 * A(k:m,1:m) = A(k:m,1:m) - 2 * v . w
26 *            = (I - 2 * v . Transpose(v)) . A(k:m,1:m)
27          DO J = 1,M
28             DO I = K,M
29                A(I,J) = A(I,J) - 2D0*V(I)*W(J)
30             END DO
31          END DO
32 * w = A(1:m,k:m) . v est un vecteur colonne de dimension m
33          DO I = 1,M
34             W(I) = 0D0
35             DO J = K,M
36                W(I) = W(I) + V(J)*A(I,J)
37             END DO
38          END DO
39 * A(1:m,k:m) = A(1:m,k:m) - 2 * w . Transpose(v)
40 *            = A(1:m,k:m) . (I - 2 * v . Transpose(v))
41          DO I = 1,M
42             DO J = K,M
43                A(I,J) = A(I,J) - 2D0*W(I)*V(J)
44             END DO
45          END DO
46       END DO
47       END SUBROUTINE
```

FIGURE 1 – Le sous-programme `EIGEN`.