lundi 01/03/2009

TP 4 d'analyse numérique

Le but de ce TP est d'implémenter les méthodes de la puissance (les 3 variantes) ainsi que la décomposition et la méthode QR, pour le calcul des valeurs et vecteurs propres d'une matrice carrée, $n \times n$, réelle et symétrique (les valeurs propres d'une telle matrice sont réelles).

Le fichier "`http://10.27.3.235/TP/matrices.txt`" contient les définitions des matrices A, B et C, de dimensions 3×3, 4×4 et 8×8. Il contient aussi les définitions des vecteurs $\vec{a}$, $\vec{b}$ et $\vec{c}$, de dimensions 3, 4 et 8. Vous pouvez copier ce fichier dans votre compte et l'inclure dans votre programme principal.

1. Calculer, puis afficher, les matrices symétriques $A_s = AA^t$, $B_s = BB^t$ et $C_s = CC^t$. (Vous pouvez bien sûr ré-utiliser les fonctions du TP3).

2. Ecrire en C, dans le cas général d'une matrice A, $n \times n$, une fonction qui calcule la valeur propre λ_1 qui a le module le plus grand, ainsi que le vecteur propre associé, que vous normaliserez à la fin du calcul à 1 (c'est la méthode de la puissance directe : vous supposerez que les conditions nécessaires pour appliquer cette méthode sont réunies).
Vous appellerez cette fonction "`methode_de_la_puissance_directe`", et elle devra avoir 5 arguments :
– le premier argument sera le tableau contenant la matrice A,
– le deuxième argument la dimension n,
– le troisème argument contiendra le vecteur initial (c'est un tableau en C) $\vec{x}_0$ avec lequel on démarre l'itération (et où on remet le résultat après chaque itération),
– le quatrième argument est la précision (vous arrêterez l'itération lorsque $|\lambda^{(k)} - \lambda^{(k-1)}| < \epsilon$),
– le cinquième argument sera un pointeur à un double, où vous enregistrerez la valeur propre approchée obtenue après un nombre k d'itérations.
Voici un exemple de comment définir cette fonction :

```
void methode_de_la_puissance_directe( double A[], int n, double x0[],
double eps , double *lambda)
                  { ...   }
```

Ecrire ensuite un petit programme qui calcule et affiche cette valeur propre (ainsi que son vecteur propre associé) pour les matrices A_s, B_s et C_s calculées précédemment.
Utilisez une précision de $\varepsilon = 10^{-12}$.
Exemple :

```
int n=3; double x0[n]={ ... }; double eps=1.e-10; double lambda;
methode_de_la_puissance_directe( A_s, n , x0, eps , &lambda);
```

3. Ecrire en C, dans le cas général d'une matrice A, $n \times n$, une fonction qui calcule la valeur propre λ_2 qui a le module le plus petit, ainsi que le vecteur propre associé, que vous normaliserez à la fin du calcul à 1 (c'est la méthode de la puissance inverse : vous supposerez que les conditions nécessaires pour appliquer cette méthode sont réunies).
Vous appellerez cette fonction "`methode_de_la_puissance_inverse`", et elle devra avoir 5 arguments (voir question précédente).
Ecrire ensuite un petit programme qui calcule et affiche cette valeur propre (ainsi que son vecteur propre associé) pour les matrices A_s, B_s et C_s calculées précédemment.
Utilisez une précision de $\varepsilon = 10^{-12}$.

4. Ecrire en C, une fonction qui calcule la valeur propre λ_s de la matrice $B = A - s\,I$ qui a le module le plus petit, ainsi que le vecteur propre associé, que vous normaliserez à la fin du calcul à 1 (c'est la méthode de la puissance inverse pour la matrice B, qui est carrée, $n \times n$, mais est appelée la méthode de la puissance avec translation pour la matrice A, également carrée, $n \times n$. Vous supposerez que les conditions nécessaires pour appliquer cette méthode sont réunies). Vous appellerez cette fonction "`methode_de_la_puissance_avec_translation`", et elle devra avoir 6 arguments. Par exemple :

```
void methode_de_la_puissance_avec_translation( double A[], int n,
          double s, double x0[], double eps , double *lambda_s)
             { ... }
```

Remarque : cette méthode permet d'obtenir les valeurs propres intermédiaires de A si vous choisissez s suffisamment proche d'une de ces valeurs propres intermédiaires; vous l'utiliserez plus tard, en la combinant avec la méthode QR (voir plus bas).
Testez votre fonction sur un petit programme en C que vous inventerez !

5. Ecrire en C, dans le cas général d'une matrice non singulière A une fonction qui décompose cette matrice A en un produit QR, ou Q est une matrice orthogonale et R est triangulaire supérieure (utlisez le processus de Gram-Schmidt). Toutes les matrices sont carrées, $n \times n$.
Cette fonction devra d'abord vérifier que A a un déterminant non nul.
Ecrire ensuite un petit programme qui utilise cette fonction pour décomposer chacune des matrices A, B et C données plus haut.
Vous devrez vérifier, à la fin, par exemple, que $\det A = \pm \det R$ (car la matrice orthogonale Q a toujours un déterminant égal à ± 1).

<u>Rappel :</u> Le processus de Gram-Scmidt est le suivant : On écrit A et Q comme des vecteurs $\vec{a}_i$ et $\vec{q}_i$.
$A = (\,\vec{a}_1\ \vec{a}_2\ \ldots\ \vec{a}_n\,) \qquad ; \qquad Q = (\,\vec{q}_1\ \vec{q}_2\ \ldots\ \vec{q}_n\,)$
Les $\vec{a}_i$ sont supposés donnés, et on cherche les $\vec{q}_i$. L'algorithme pour trouver les $\vec{q}_i$ (ainsi que la matrice R) est :
Pour $i = 1, \ldots, n$

1. *Calculer* $\vec{a}_i{}' = \vec{a}_i - \sum_{k=1}^{i-1} (\vec{a}_i \cdot \vec{q}_k)\, \vec{q}_k$
2. $\vec{q}_i = \vec{a}_i{}' / ||\vec{a}_i{}'||$
3. $R_{ii} = ||\vec{a}_i{}'||$
4. $R_{ij} = \vec{q}_i \cdot \vec{a}_j$ *, pour* $j = i+1, ..., n$ *(Remarque :* $R_{ij} = 0$ *si* $i > j$ *).*

6. Implémentez la méthode QR par une fonction en C que vous appellerez `methode_QR`. Elle ressemblera à ceci

```
void methode_QR( double A[], int n, double B[], int k)
   { ... }
```

Cette méthode consiste à démarrer avec une matrice connue A que nous écrivons ici A^1. On décompose $A^{(1)}$ en un produit $Q^{(1)} R^{(1)}$, puis on définit la matrice $A^{(2)} \equiv R^{(1)} Q^{(1)}$, que l'on décompose en un produit $Q^{(2)} R^{(2)}$, et ainsi de suite jusqu'à obtenir la matrice $A^{(k)} \equiv R^{(k-1)} Q^{(k-1)}$. On arrête l'itération, et on met cette matrice $A^{(k)}$ dans la matrice B. (Toutes les matrices sont carrées, $n \times n$)
Ecrire ensuite un petit programme qui utilise cette méthode sur les matrices A_s, B_s et C_s (faites $k = 30$ itérations) , et en déduire TOUTES les valeurs et vecteurs propres de ces matrices, en combinant cette méthode avec la méthode de la puissance avec translation, en utilisant une précision de $\varepsilon = 10^{-12}$.