Examen du 18 mai 2016, de 9h à 12h.

Documents, calculatrices et ordinateurs ultraportables (netbooks) déconnectés du réseau autorisés.

Ce sujet comporte 2 pages. Barême donné à titre indicatif et non contractuel.

1. VALEURS SINGULIÈRES D'UNE MATRICE (11 POINTS)

Soit A une matrice non nulle carrée d'ordre n et A^* sa transposée (ou sa transconjuguée si A est à coefficients complexes), on note $B = A^*A$

(1) Montrer que B est diagonalisable de valeurs propres réelles positives.

(2) Soit v un vecteur non nul tel que $\|(B-\mu)v\| \leq \varepsilon\|v\|$, montrer que B admet une valeur propre dans l'intervalle $[\mu-\varepsilon, \mu+\varepsilon]$ (on pourra écrire v dans une base orthonormale bien choisie relativement à B).

(3) Soit $\lambda > 0$ la plus grande valeur propre de B, donner un algorithme permettant de trouver une valeur approchée $\tilde{\lambda}$ de λ avec une précision ε en utilisant les suites $v_k = w_k/\|w_k\|, w_{k+1} = Bv_k$ pour un vecteur w_0 à coefficients aléatoires.

(4) Programmer l'algorithme et l'appliquer à la matrice de Hilbert de taille 5 et 10. On pourra utiliser la commande `hilbert(5), hilbert(10)`.

(5) Quel est le cout d'une itération de l'algorithme en fonction de la taille n de la matrice ? Comparer avec le cout du calcul de B. Peut-on reécrire l'algorithme pour ne pas avoir à calculer B ?

(6) Proposer une algorithme de calcul de la valeur approchée de la plus petite valeur propre de B en utilisant B^{-1}. Tester l'algorithme avec la matrice de Hilbert de taille 5. Pour une matrice de Hilbert de taille 10, il faut choisir une tolérance relative, donc un test d'arrêt du type $\|w_{k+1} - \lambda v_k\| \leq \varepsilon|\lambda|\|v_k\|$, expliquer pourquoi.

(7) Quel est le cout de cet algorithme en fonction de n ? Est-il possible d'optimiser en utilisant la décomposition LU de B ou de A ?

(8) On se propose d'appliquer l'algorithme de la question (3) à $\tilde{\lambda}I_n - B$ où $\tilde{\lambda}$ est une valeur approchée de la plus grande valeur propre de B. Montrer que la valeur propre approchée obtenue donne une estimation de la plus petite valeur propre de B. Comparer le cout de l'algorithme avec celui de la question précédente et comparer la qualité des approximations obtenues pour la plus petite valeur propre de B.

(9) Rappeler l'expression du conditionnement en norme euclidienne d'une matrice A en fonction de ses valeurs singulières et l'expression des valeurs singulières de A en fonction des valeurs propres de B.
Donner un algorithme de calcul du conditionnement de A relativement à la norme euclidienne.

(10) Tester avec une matrice de Vandemonde (`vandermonde`) prise en 11 points équidistants de $[-1,1]$ puis en 11 points de Tchebyshev de $[-1,1]$ (i.e. les 11 racines du 11-ième polynôme de Tchebyshev).

(11) On voudrait calculer la deuxième plus grande valeur propre de B, pour cela on considère la matrice $B' = B - \tilde{\lambda}ww^*$ où w est un vecteur propre colonne normé approché de B correspondant à $\tilde{\lambda}$. Déterminer les valeurs propres de B' en faisant l'approximation que w est un vecteur propre de B. En déduire un algorithme permettant de déterminer la deuxième plus grande valeur propre de B. Peut-on écrire cet algorithme sans calculer explicitement B ?

2. ÉQUATION DIFFÉRENTIELLE (9 POINTS)

Pour résoudre de manière approchée une équation (ou un système différentiel) $y' = f(t,y)$ avec f de classe C^2, on va appliquer la méthode explicitée ci-dessous, appelée méthode de Heun de pas constant h.

Soit $y_0 = y(t_0)$ la condition initiale, $t_n = t_0 + nh$, on approche $y(t_n)$ par la suite récurrente y_n définie par :

$$\tilde{y}_{n+1} = y_n + hf(t_n, y_n), \quad y_{n+1} = y_n + \frac{h}{2}(f(t_n, y_n) + f(t_n + h, \tilde{y}_{n+1}))$$

(1) On suppose que $f(t,y)$ ne dépend pas de y, quelle est la méthode numérique d'intégration correspondante ? En déduire une majoration de l'erreur locale (ou erreur de consistance) $|y_1 - y(t_0 + h)|$ en fonction de h dans ce cas.

(2) Dans toute la suite f peut dépendre de y. On pourra utiliser sans le justifier que $F : t \to f(t, y(t))$ est C^2 sur l'intervalle de définition maximal de y.
Donner une valeur approchée de l'intégrale $I = \int_{t_0}^{t_0+h} f(t, y(t))$ par la méthode des rectangles et une majoration de l'erreur $\tilde{y_1} - y(t_0 + h)$.
Donner une valeur approchée de l'intégrale I par la méthode de la question (1), puis une majoration de l'erreur locale $|y_1 - y(t_0 + h)|$.

(3) On pose $y_{n+1} = y_n + h\Phi(t_n, y_n, h))$, donner l'expression de Φ en fonction de f, justifier que Φ est lipschitzienne par rapport à y, et en déduire une majoration de l'erreur globale $|y_n - y(t_n)|$.

(4) On suppose dans la suite que $f(t,y) = ty$ et que $h \leq 0.1, t \in [0,T]$. Calculer une constante de Lispchitz pour Φ.

(5) Programmer la méthode. Déterminer une valeur approchée de $y(1)$ obtenue par cette méthode pour l'équation $y' = ty, y(0) = 1$ avec $h = 0.1, 0.01, 0.001$. Comparer avec la valeur exacte de $y(1)$, observe-t-on le comportement attendu en puissance de h ?

(6) Déterminer de même une valeur approchée y_N de $y(t_N)$ pour $t_N = 4$ et $h = 0.1, 0.01, 0.001$ (donc $N = 40, 400, 4000$), calculer l'erreur en utilisant la valeur exacte de $y(t_N)$, observe-t-on le comportement attendu en fonction de t_N ?

(7) Soit l'équation d'ordre 2 $y'' + ty' + y = 0$. L'écrire comme un système différentiel d'ordre 1 en $Y = (y, y')$ et appliquer la méthode précédente pour déterminer une valeur approchée de $y(1)$ pour la solution de condition initiale $y(0) = 1, y'(0) = 0$ ($Y(0) = (1,0)$).

Notes de correction 1 (13.5 points=.5+1.5+1.5+.5+1.5+1.5+1+2+1+1+1.5)

(1) car B est symétrique positive

(2) par l'absurde, on minore la norme euclidienne au carrée de $(B-\mu)v$ par ε^2 fois la norme au carré de v

(3) Méthode de la puissance (cf. TD/cours)

```
svd1(A,eps,N):={
  local w,v,j,At,l;
  At:=trn(A);
  w:=ranv(size(A),0..1);
  for j from 1 to N do
    v:=normalize(w);
    w:=A*v; w:=At*w;
    l:=dot(v,w);
    if (l2norm(w-l*v)<eps*abs(l)){ return v,l; }
  od;
  retourne "maxiter";
}:;
```

(4) En principe c'était A la matrice de Hilbert, et donc $B = A^*A$.

(5) Le cout de l'itération est dominée par la multiplication matrice-vecteur en $O(n^2)$ (le reste en $O(n)$). Négligeable devant le cout multiplication matrice-matrice pour calculer B en $O(n^3)$. On peut éviter ce cout en faisant la multiplication du vecteur par A puis par A^*.

(6) on prend l'inverse de la plus grande valeur propre de B^{-1}. La plus grande valeur propre de B^{-1} est grande pour `hilbert(5)` et même très grande pour `hilbert(10)` (onze chiffres), obtenir une convergence avec un test absolu est impossible si on travaille en flottants (pour $\varepsilon < 1e-5$ en tout cas, et serait beaucoup trop lent en flottants multi-précision).

(7) le cout est le même que pour la plus grande valeur propre de B une fois B^{-1} calculé ce qui nécessite $O(n^3)$ opérations (calcul de B et de B^{-1}). On peut un peu optimiser la constante dans le $O(n^3)$ en remplaçant `w:=A*v; w:=At*w;` par `w:=linsolve(p,l,u,v)` où p,l,u est la décomposition LU de B, voire en économisant la multiplication $B = A^*A$, en décomposant la résolution de $A^*Aw = v$ en plusieurs systèmes triangulaires. Mais cela reste en $O(n^3)$.

(8) La plus grande valeur propre de $\tilde{\lambda} I_n - B$ est $\tilde{\lambda}$ moins la plus petite valeur propre de B. On peut donc appliquer la méthode de la puissance directement sur $\tilde{\lambda} I_n - A^*A$ pour avoir une estimation de la plus petite valeur propre de B, ce qui coutera $O(kn^2)$ si k est le nombre d'itérations qui dépend linéairement de la précision souhaitée.
Mais pour les matrices de Hilbert, la plus petite valeur propre est trop proche de 0 pour obtenir une estimation par cette méthode, on va trouver une estimation proche de $\tilde{\lambda}$ et on va donc soustraire deux nombres très proches l'un de l'autre, en perdant tous les chiffres significatifs (sauf si on travaille en multi-précision mais le nombre d'itérations sera très grand).

(9) conditionnement L^2 de A=racine carrée du rapport de la plus grande valeur singulière sur la plus petite de A^*A, donc racine carrée de l'estimation de la plus grande valeur propre de B sur la plus petite.

(10) On applique à `vandermonde(seq(-1,1,0.1)` et `vandermonde(proot(tchebyshev1(11)))` (on peut aussi écrire explicitement les racines).

(11) L'orthogonal de w est stable par B et B' et les valeurs propres sont identiques. Pour w on remplace la valeur propre λ par 0. Donc la plus grande valeur propre de B' est la seconde valeur propre de B. On peut écrire l'algorithme de la puissance sans calculer B en multipliant par A puis A^*.

Notes de correction 2 : (10.5=1+2+1.5+1+3+1+1)

(1) méthode des trapèzes, donc erreur locale sur un pas en $M_2h^3/12$ (M_2 =max$|f''|$)

(2) les rectangles à gauche donnent $\tilde{y}_1$ donc $|\tilde{y}_1 - y(t_0+h)|$ est majoré par $M_1h^2/2$ (ici M_1=max$|F'|$). On applique les trapèzes à F, on en déduit que

$$|y(t_0+h) - y(t_0) - \frac{h}{2}(f(t_0,y_0) + f(t_0+h, y(t_0+h)))| \leq M_2h^3\frac{1}{12}$$

avec ici M_2=max$|F''|$. Donc

$$\begin{aligned} |y_1 - y(t_0+h)| &\leq \frac{h}{2}|f(t_0+h,\tilde{y}_1) - f(t_0+h,y(t_0+h))| + M_2h^3\frac{1}{12} \\ &\leq \frac{h}{2}\Lambda|\tilde{y}_1 - y(t_0+h)| + M_2h^3\frac{1}{12} \\ &\leq \frac{h^3}{12}(3\Lambda M_1 + M_2) \end{aligned}$$

(3)

$$\Phi(t_n,y_n,h) = f(t_n,y_n) + f(t_n+h, y_n + hf(t_n,y_n))$$

est donc lipschitzienne en y_n :

$$\begin{aligned} |\Phi(t_n,z,h) - \Phi(t_n,y,h)| &\leq |f(t_n,z) - f(t_n,y)| + |f(t_n+h, z+hf(t_n,z)) - f(t_n+h, y+hf(t_n,y))| \\ &\leq \Lambda(|z-y| + |z+hf(t_n,z) - (y+hf(t_n,y))| \\ &\leq \Lambda(2+h\Lambda)|z-y| \end{aligned}$$

D'où une erreur globale en $h^2/12(3\Lambda M_1 + M_2)\frac{1}{C}(e^{Ct_n} - 1)$ avec C la constante de Lipschitz de Φ.

(4) On peut prendre $\Lambda = T$ donc $C = T(2+.1T)$

(5)
```
heun(f,t0,t1,N,y0):={
  local h,j,t,y1;
  h:=evalf(t1-t0)/N;
  pour j de 0 jusque N-1 faire
    t:=t0+j*h;
    y1:=y0+h*f(t,y0);
    y0:=y0+h/2*(f(t,y0)+f(t+h,y1));
  fpour;
  retourne y0;
}:;
```

`heun(f,0,1,10,1.0)` renvoie 1.64788134551. `heun(f,0,1,100,1.0)` renvoie 1.6487142374 `heun(f,0,1,100,1.0)` renvoie 1.64872120183 La solution exacte est $y'/y = t$ donc $y = Ke^{t^2/2}$ avec $K = 1$ (condition initiale 1 en $t = 0$). On compare donc avec $e^{0.5} = -1.6487212707$. On a le comportement attendu d'une erreur en h^2 (on gagne 2 décimales si on divise h par 10, erreur de `6e-8` pour `h:=1e-3`).

(6) Si $t_n = 4$ au lieu de 1, on a `heun(f,0,4,40,1.0)-exp(8)` qui vaut -242, `heun(f,0,4,400,1.0)-exp(8)` qui vaut -3, `heun(f,0,4,400,1.0)-exp(8)` qui vaut -0.03, on a toujours le bon comportement en puissances de h, mais une erreur environ 1 million de fois plus grande. La constante de Lipschitz qui passe de 2.1 à 9.6, qui modifie le terme $\frac{1}{C}(e^{Ct_n} - 1)$ de 3 à `5e15` expliquant un rapport d'erreur très grand (on aurait d'ailleurs pu s'attendre à beaucoup plus que 1 million).

(7) `f(t,y):=[[0,1],[-1,-t]]*y` et `heun(f,0,1,10,[1,0])` renvoie $[0.605257530382, -0.605257530382]$