

TP7 : le théorème du point fixe en action sous MATLAB

Cette séance de TP7 poursuit la familiarisation avec MATLAB. Elle illustre le chapitre 4 du cours (le théorème du point fixe et ses applications en algèbre linéaire). Ouvrez MATLAB pour commencer.

EXERCICE 1 (l'algorithme itératif conduisant au calcul approché du rayon spectral d'une matrice carrée réelle).

- (1) Déclarez sous MATLAB la matrice symétrique réelle (de taille $(4, 4)$) suivante :

```
>> A
A =

    0.5172    0.5473   -1.2240    0.8012
    0.5473    1.3880    1.3530   -1.1120
   -1.2240    1.3530    0.0364    2.8930
    0.8012   -1.1120    2.8930    0.0583
```

En utilisant la commande `eig(A)`, calculez les quatre valeurs propres réelles de cette matrice réelle symétrique A . Cette matrice A est-elle définie positive ? Est-elle diagonalisable ? Quel est le signe de la valeur propre de valeur absolue égale au rayon spectral $r(A)$ de cette matrice ? Quelle est la dimension (dans \mathbb{R}^4) du sous-espace propre correspondant ?

- (2) Téléchargez depuis le site

```
http://www.math.u-bordeaux1/~yger/initiationMATLAB
```

la routine `rayonspectral`. Modifiez ensuite cette routine pour réaliser une routine

```
[r,Niter] = rayonspectral1(A,X,epsilon,k);
```

qui, étant donné un nombre maximal k d'itérations autorisées, une matrice carrée (réelle ou complexe) de taille (N, N) A , un vecteur colonne X de longueur N , calcule de manière approchée, dans les cas favorables, le rayon spectral r de la matrice A , l'algorithme étant initié à X , ainsi que le nombre $Niter \leq k$ d'itérations nécessaires avant que l'erreur entre une approximation et la suivante ne vienne à passer en valeur absolue sous le seuil `epsilon`¹. Testez cet algorithme sur la matrice A de la question 1 en prenant $k = 100$, `epsilon=eps`, et pour X respectivement

1. Cela ne donne pas *a priori* une majoration par `epsilon` de l'erreur commise entre le rayon spectral et sa valeur approchée (au terme de $Niter$ itérations), mais seulement un contrôle de

```
>> X1 = [1;1;1;1];
>> X2 = [1;1;0;0];
>> X3 = [1;0;0;0];
>> X4 = rand(4,1);
```

Quelle valeur de `Niter` trouvez vous dans chacun de ces cinq cas ? Vérifiez que la valeur de `r` obtenue alors est bien en accord avec le résultat fourni à la question 1 par la commande `eig(A)` donnant les quatre valeurs propres (dans ce cas réelles) de la matrice `A`. Recommencez avec cette fois `epsilon = 10-6`.

- (3) On suppose que `A` est une matrice réelle et que le rayon spectral $r(A)$ est la valeur absolue d'une valeur propre réelle simple de la matrice `A`. Vérifiez que c'est bien le cas pour la matrice `A` donnée à la question 1. Modifiez la routine `rayonspectral1` en une routine `AppVPdom` (« Approximation du Vecteur Propre associé à la valeur propre dominante ») :

```
[VP,Niter] = AppVPdom(A,X,epsilon,k);
```

qui fournisse, avec les mêmes données que dans `rayonspectral1` en *input*,
– le vecteur $VP_{Niter+1}$ de la suite initiée à $X_1 = X/\text{norm}(X)$ et régie ensuite par l'équation récurrente

$$VP_{k+1} = \frac{A \cdot VP_k}{\text{norm}(A \cdot VP_k)}, \quad k \geq 1.$$

- le nombre d'itérations `Niter` effectué dans la boucle sachant que cette boucle s'arrête dès que, pour la première fois :

```
min (norm(VP_(Niter) -VP_(Niter-1)),
      norm (VP_(Niter)+ VP_(Niter-1))) <= epsilon
```

Testez cet algorithme avec la matrice `A` en prenant `epsilon=eps`, `k=200` et respectivement `X = X1,X2,X3,X4` comme à la question 2. Quelle valeur de `Niter` obtenez vous dans chacun de ces quatre cas ? Recommencez avec cette fois `epsilon = 10-6`. Calculez les vecteurs propres (normalisés) de la matrice `A` en utilisant :

```
>> [V,D] = eig(A);
```

```
>> V
```

Comparez le vecteur propre $Y=V(:,1)$ ainsi obtenu (correspondant à la valeur propre de valeur absolue $r(A)$) et le vecteur `VP` obtenu *via*

```
>> [VP,Niter] = AppVPdom(A,X,epsilon,k);
```

- (4) Modifiez la routine `AppVPdom` construite à la question 3 en une nouvelle routine

```
function [VP,Niter,errY] = AppVPdom1(A,X,Y,epsilon,k);
```

qui, en plus des sorties `VP` et `Niter` (comme pour la routine `AppVPdom` construite à la question 3), fournit aussi la liste `errY` des erreurs successives²

cette erreur à un facteur multiplicatif près, ce contrôle étant de l'ordre de $\text{epsilon}/(1-\rho)$, où ρ désigne le rapport entre $r(A)$ et le module de la première valeur propre de module strictement inférieur à $r(A)$. Voir pour cela la preuve de la Proposition 4.2 du cours.

2. Même remarque que précédemment à propos du contrôle d'erreur : cette tolérance `epsilon` contrôle l'erreur entre VP_{Niter} et un vrai vecteur propre normalisé (pour la valeur propre $\pm r(A)$) en $\text{epsilon}/(1-\rho)$, où ρ a été défini dans la note 1 précédente.

```
min(norm(VP_k-Y), norm(VP_k+Y)), k=1,2,...,Niter
```

lorsque Y est un vecteur colonne de \mathbb{R}^4 donné en *input* en plus des données précédentes A, X, ϵ, k . En utilisant cette routine avec $\epsilon = 10^{-6}$, $Y=V(:,1)$ (vecteur propre normalisé de la matrice A correspondant à la valeur propre de valeur absolue $r(A)$) et $X=X1, X2, X3, X4$, comparez la rapidité de la convergence de la suite $(VP_k)_k$ vers $\pm Y$ dans les quatre cas de figure (suivant la valeur du vecteur initial X depuis lequel est lancé l'algorithme).

- (5) Reprendre la question 4 en remplaçant dans la routine l'utilisation de la norme euclidienne `norm` par la norme $\|\cdot\|_\infty$ (`norm(.,inf)` sous MATLAB). On notera la nouvelle routine (obtenue en modifiant à la marge la routine `AppVPdom1` de la question 4) `AppVPdom1bis`.

EXERCICE 2 (un schéma simpliste pour `Pagerank`). Cet exercice met en œuvre l'approche proposée dans la section 4.1.2 des notes de cours (« maquette » simpliste du fonctionnement de `Google`).

- (1) Construire une routine

```
function G=AdjacencePonderee(M);
```

qui, étant donnée une matrice M de taille (N,N) dont les entrées sont constituées de 0 et de 1 (considérée comme la *matrice d'adjacence* d'un certain graphe orienté (E,V)), calcule la *matrice d'adjacence pondérée* de ce même graphe orienté, c'est-à-dire la matrice G déduite de la matrice M en transformant chaque ligne de M ainsi :

- une ligne constituée entièrement de 0 reste inchangée;
- une ligne dans laquelle figure au moins un 1 est divisée par le nombre de 1 présents sur cette ligne.

- (2) Rédigez une procédure

```
function [Lequilibre,Niter]
=Pagerank(M,L0,kappa,epsilon,k);
```

qui, étant donné un graphe orienté (E,V) à N sommets, matérialisé par sa matrice d'incidence M , un nombre κ strictement entre 0 et 1, calcule de manière approchée le vecteur ligne `Lequilibre` (de longueur N), unique point fixe de l'application strictement contractante :

$$L = ((1-\kappa)/N)*\text{ones}(1,N) + \kappa*L*G$$

lorsque :

- G désigne la matrice d'adjacence pondérée du graphe orienté (E,V) (*cf.* la question 1);
- l'algorithme est initié avec le vecteur ligne $L0$ (dont les entrées sont positives et de somme 1);
- le nombre maximal d'itérations autorisées est k ;
- le nombre $Niter \leq k$ est le nombre d'itérations nécessaires (lorsque cela est possible) jusqu'à ce que, pour la première fois³, $\text{norm}(L(Niter)-L(Niter-1)) \leq \epsilon$

3. D'après l'étude faite en cours, *cf.* la preuve du Théorème 4.1 (du point fixe), l'erreur entre L_{Niter} et sa limite est alors majorée par $\epsilon/(1-\kappa)$.

- (3) Générez un graphe orienté à 10 sommets *via* la donnée de sa matrice d'adjacence M et calculez la mesure d'équilibre $Lequilibre$ de ce graphe orienté avec le choix de $kappa=0.85$ (le choix classiquement privilégié dans l'algorithme *Pagerank*). Prendre $epsilon = 10^{-8}$, puis $epsilon=eps$, calculez aussi $Niter$ (lorsque $k=100$) et examinez la dépendance en le choix du vecteur ligne initial $L0$: on pourra prendre par exemple pour ce faire comparer les résultats
- une liste initiale « creuse » telle $L0=[1 \ 0 \ 0 \ \dots \ 0]$;
 - une liste initiale « pleine » telle $L0=L00/sum(L00)$ ($L00=rand(1,10)$).
- Affichez les résultats par exemple avec `plot(Lequilibre,'d')`.

EXERCICE 3 (algorithmes itératifs de Jacobi et de Gauß-Seidel). Téléchargez depuis le site

<http://www.math.u-bordeaux1/~yger/initiationMATLAB>

les deux routines `Jacobi.m` et `GaussSeidel.m`, respectivement basées sur les décompositions $M=D-E=diag(diag(M))-(diag(diag(M))-M)$ et $M=T-F=tril(M)-(tril(M)-M)$.

- (1) Transformez ces deux routines en des routines :

```
function [XX,Niter] = Jacobi1(M,B,X,epsilon,k);
function [XX,Niter] = GaussSeidel1(M,B,X,epsilon,k);
```

qui, étant donnés une matrice M de taille (N,N) , sans zéros sur la diagonale, et un vecteur colonne B de longueur N :

- renvoient toutes les deux le message

condition non remplie

- si la condition $r(D^{-1} \cdot E) < 1$ ou $r(T^{-1} \cdot F) < 1$ se trouve en défaut ;
- gênent, si cette condition se trouve remplie, l'algorithme itératif, soit de Jacobi, soit de Gauß-Seidel, initié sur le vecteur colonne X (aussi de longueur N), et visant à calculer de manière approchée la solution XX du système de Cramer $M \cdot XX = B$; le nombre maximal d'itérations autorisées est k (le même que celui autorisé pour calculer en préambule le rayon spectral des matrices $D^{-1} \cdot E$ ou $T^{-1} \cdot F$), et l'on décide que l'algorithme itératif s'arrête dès que, pour la première fois⁴,

```
norm(XX_(Niter)-XX_(Niter-1)) <=epsilon
```

- (2) Construisez une routine

```
[XX,Niter] = ExempleJacobi(N,B,X,epsilon,k);
```

qui, étant donné un entier strictement positif N , un vecteur B de taille $(N,1)$:

- génère la matrice creuse $M(N)$ de taille (N,N) dont la diagonale est constituée de 3, la sur-diagonale et la sous-diagonale de -1, toutes les autres entrées de la matrice étant nulles ;

4. D'après l'étude faite en cours, *cf.* la preuve du Théorème 4.1 (du point fixe), l'erreur entre XX_{Niter} et sa limite (à savoir la solution du système de Cramer que l'on tente d'approcher) est alors majorée par $epsilon/(1-r(A))$, où $A = D^{-1} \cdot E$ ou $A = T^{-1} \cdot F$ suivant le cas (Jacobi ou Gauß-Seidel).

- résout par la méthode de Jacobi (initiée au vecteur X de taille $(N,1)$, avec k itérations autorisées au plus et un seuil d'erreur `epsilon` comme à la question 1) le système de Cramer $M(N) * XX = B$, où la sortie `Niter` $\leq k$ désigne toujours le nombre d'itérations réellement effectué lors de l'algorithme de Jacobi.

Faites la même chose en remplaçant l'algorithme de Jacobi par celui de Gauß-Seidel pour construire une fonction similaire :

```
[XX,Niter] = ExempleGaussSeidel(N,B,X,epsilon,k);
```

(l'algorithme de Jacobi ayant cette fois été remplacé par l'algorithme de Gauß-Seidel).

- (3) Générez une matrice réelle A de taille $(20,20)$ en utilisant la routine

```
A = 2*(rand(20)-ones(20)/2);
```

Générez ensuite la matrice $A*A'$. L'algorithme de Gauss-Seidel converge-t-il lorsque la matrice M est la matrice $M=A*A'$? Pourquoi? Vérifiez le en générant un vecteur $B=rand(20,1)$ et en essayant de résoudre le système de Cramer $(A*A') * XX = B$ de manière itérative en utilisant la routine

```
[XX,Niter] = GaussSeidel1(A*A',B,zeros(20,1),10^(-8),k);
```

Comparez le résultat que vous obtenez ainsi avec celui donné par la résolution directe

```
>> (A*A')^(-1)*B
```

Comment faut-il choisir k pour que les deux résultats soient en cohérence? Calculez `eig(T^-1 * F)` pour la décomposition de Gauß-Seidel $A*A'=T-F$ et expliquez pourquoi il s'avérait nécessaire de choisir de telles valeurs de k pour appliquer l'algorithme de Gauß-Seidel.

- (4) Soient les trois matrices :

$$\begin{pmatrix} 1 & .75 & .75 \\ .75 & 1 & .75 \\ .75 & .75 & 1 \end{pmatrix} \quad \begin{pmatrix} 1 & 2 & -2 \\ 1 & 1 & 1 \\ 2 & 2 & 1 \end{pmatrix} \quad \begin{pmatrix} 2 & -1 & 1 \\ 2 & 2 & 2 \\ -1 & -1 & 2 \end{pmatrix}.$$

Quels algorithmes (de Jacobi ou de Gauß-Seidel) sont-ils convergent pour la résolution itérative des systèmes $M * XX = B$ lorsque M est l'une de ces trois matrices (étudiez les trois cas séparément)?

EXERCICE 4 (le conditionnement des matrices et les facteurs d'amplification d'erreurs relatives dans la résolution des systèmes de Cramer perturbés).

- (1) En utilisant la routine `svd` :

```
>> [U,D,V] =svd(M);
```

donnant la *décomposition en valeurs singulières* d'une matrice réelle ou complexe (cf. la section 4.4.1 du cours), écrivez une routine

```
function c=ConditionnementNorme2(M);
```

qui donne la valeur du conditionnement d'une matrice carrée inversible M relativement à la norme $\| \cdot \|_2$. En utilisant cette routine, calculez le conditionnement de la matrice

$$A := \begin{pmatrix} 10 & 7 & 8 & 7 \\ 7 & 5 & 6 & 5 \\ 8 & 6 & 10 & 9 \\ 7 & 5 & 9 & 10 \end{pmatrix}$$

utilisée dans la section 1.2.2 du cours. Comparez avec le résultat donné par `cond(A)`. En utilisant les routines

```
>> help cond
>> cond(A,1)
>> cond(A,inf)
```

calculez le conditionnement de la matrice A relativement au choix de la norme matricielle $\| \cdot \|_1$ et de la norme matricielle $\| \cdot \|_\infty$.

- (2) Perturbez la matrice A en lui ajoutant la perturbation :

$$\text{DeltaA} := \begin{pmatrix} 0 & 0 & .1 & .2 \\ .08 & .04 & 0 & 0 \\ 0 & -.02 & -.11 & 0 \\ -.01 & -.01 & 0 & -.02 \end{pmatrix}$$

Calculez les solutions XX et XXX des deux systèmes de Cramer

$$A * XX = B, \quad (A + \text{DeltaA}) * XXX = B$$

si $B = [32; 23; 33; 31]$, d'abord par la méthode directe :

```
>> XX = A^{-1} * B ;
>> XXX = (A+DeltaA)^{-1} * B ;
```

puis par les méthodes itératives :

```
>> [XX1,Niter] = GaussSeidel(A,B,zeros(4,1),10^(-8),k);
>> [XXX1,Niter] = GaussSeidel(A,B,zeros(4,1),10^(-8),k);
```

Calculez le coefficient d'amplification d'erreur relative :

```
>> (norm(XXX-XX)/norm(XXX))*(norm(A)/norm(DeltaA))
```

Que remarquez vous ? Quel est l'ordre de grandeur de ce coefficient d'amplification des erreurs relatives ?

- (3) On perturbe maintenant le vecteur B (de la question 2) par le vecteur $\text{DeltaB} = [.01; -.01; .01; -.01]$. Résoudre (de deux manières différentes, comme à la question 2 (directement ou alors de manière itérative en utilisant l'algorithme de Gauß-Seidel) le système de Cramer :

$$A * XXXX = \text{DeltaB}.$$

Calculez encore le coefficient d'amplification de l'erreur relative :

```
>> (norm(XXXX-XX)/norm(XXXX))*(norm(B)/norm(DeltaB))
```

Que remarquez vous à nouveau ? Quel est l'ordre de grandeur de ce coefficient d'amplification des erreurs relatives ?