Programmation linéaire et convexité
TP1

17 février 2010

L'objectif de cette séance est de réviser les commandes de base du logiciel scilab liées au calcul matriciel et de les utiliser dans le cadre de la résolution de systèmes linéaires : $AX = b$, où $A \in \mathcal{M}_{n,m}(\mathbb{R})$, $b \in \mathbb{R}^n$ et $X \in \mathbb{R}^m$ désigne l'inconnue.

1 Calcul matriciel sous scilab

Scilab est un logiciel de calcul scientifique développé par l'INRIA. Vous pouvez le télécharger gratuitement en ligne à l'adresse suivante `http ://www.scilab.org`. Dans le cadre de cette séance, seules les fonctions de base du calcul matriciel seront utilisées.

Exercice 1 On considère la matrice $A \in \mathcal{M}_{3,4}(\mathbb{R})$ définie par :

$$\begin{pmatrix} 4 & 6 & -2 & 3 \\ 2 & -1 & 0 & 1 \\ -7 & 0 & 1 & 12 \end{pmatrix}.$$

1. Définir la matrice A sous scilab.
2. Multiplier les deux premières lignes par 2, puis diviser la dernière colonne par 3.
3. Créer une nouvelle matrice B définie par

$$\begin{pmatrix} 4 & 5 & 6 \\ 5 & 10 & 15 \\ 1 & 1 & 1 \end{pmatrix},$$

   en utilisant le fait que les lignes 1 et 2 sont composées des éléments successifs de deux suites arithmétiques.
4. Créer la matrice $C \in \mathcal{M}_3(\mathbb{R})$ extraite de A telle que pour $1 \leq i, j \leq 3$, $c_{ij} = a_{ij}$.
5. Différents produits matriciels :
   - Définir la matrice $D = BA$ *produit matriciel standard* de B et A ;
   - Définir la matrice $E = B \cdot C$ *produit de Hadamard* des mêmes matrices.

   Pour mémoire, le produit de Hadamard $E \in \mathcal{M}_3(\mathbb{R})$ des matrices $B = (b_{ij})_{1 \leq i,j \leq 3}$ et $C = (c_{ij})_{1 \leq i,j \leq 3}$ est défini par : $\forall\ 1 \leq i, j \leq 3$, $e_{ij} = b_{ij} c_{ij}$.
6. Calculer la somme des éléments de la matrice E et le vecteur colonne $Y \in \mathbb{R}^3$ tel que pour $1 \leq i \leq 3$, $y_i = \sum_{j=1}^{4} d_{ij}$.

Exercice 2 Quelques fonctions très utiles en algèbre linéaire.

On considère la matrice $A \in \mathcal{M}_4(\mathbb{R})$

$$\begin{pmatrix} 4 & 5 & 6 & -1 \\ 5 & 10 & 15 & 2 \\ 6 & 15 & 1 & 4 \\ -1 & 2 & 4 & -2. \end{pmatrix}.$$

1. Pourquoi A est-elle diagonalisable? A l'aide de la fonction `spec`, calculer sous scilab ses valeurs propres et donner une base de vecteurs propres.
2. Calculer de deux manières l'inverse de A. En utilisant la fonction `inv` d'une part et en utilisant le résultat de la question précédente d'autre part. Comparer les résultats obtenus.
3. Mêmes questions pour la matrice $A \in \mathcal{M}_n(\mathbb{R})$

$$\begin{pmatrix} 2 & -1 & 0 & \cdots & \cdots & 0 \\ -1 & 2 & -1 & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & -1 & 2 & -1 \\ 0 & \cdots & \cdots & 0 & -1 & 2 \end{pmatrix}.$$

Examiner les cas $n = 5, 10, 50$. Comparer les valeurs propres obtenues avec les valeurs définies pour $1 \leq k \leq n$ par

$$\lambda_k = 4\sin^2\left(\frac{k\pi}{2(n+1)}\right).$$

Exercice 3 On s'intéresse à la matrice $A \in \mathcal{M}_{3,5}(\mathbb{R})$

$$\begin{pmatrix} 1 & -1 & 2 & 1 & 2 \\ -1 & 2 & 3 & -4 & 1 \\ 0 & -1 & 1 & 0 & 0 \end{pmatrix}.$$

1. Quel est le rang de la matrice A? Utiliser la fonction `rank` de scilab afin de retrouver ce résultat.
2. Calculer une base de Ker A. Utiliser la fonction `kernel` de scilab afin de retrouver ce résultat.
3. A l'aide de la fonction `linsolve`. Calculer une solution de référence du système $AX = b$ où $b = (3, -7, 1)^T$. Cette fonction recalcule une base de Ker A. Vérifier que les noyaux donnés par `kernel` et `linsolve` sont identiques.
4. Tester la fonction `linsolve` sur les exemples suivants et donner dans chaque cas le rang de la matrice.

$$A = \begin{pmatrix} 1 & -4 & 2 & 1 & 2 & 3 & 6 & 7 \\ 8 & 21 & 3 & -1 & 3 & 1 & 1 & -2 \\ 0 & -1 & -1 & 0 & 0 & 0 & 4 & 5 \\ 11 & -13 & 2 & 1 & 22 & 3 & 6 & 7 \end{pmatrix}, \; b = \begin{pmatrix} 2 \\ 4 \\ -3 \\ 7 \end{pmatrix}$$

$$A = \begin{pmatrix} 1 & -3 & 2 & 1 & 21 & 3 \\ 0 & 1 & 2 & 0 & 3 & 0 \\ -2 & -7 & -1 & 0 & 5 & -6 \\ 4 & -1 & 14 & 1 & -5 & 1 \end{pmatrix}, \; b = \begin{pmatrix} 2 \\ 4 \\ -3 \\ 7 \end{pmatrix}.$$

2 Systèmes linéaires et pivot de Gauss-Jordan

Bien qu'il existe, comme on vient de le voir, des outils génériques pour traiter la résolution de problèmes linéaires sous scilab, on va s'intéresser à l'implémentation dans cet environnement de la méthode du pivot. Il s'agit d'éviter de recourir aux boites noires que constituent les fonctions préprogrammées. L'objectif est d'illustrer la simplicité de l'implémentation sous scilab des opérations associées à la méthode.

Exercice 4 On s'intéresse au système de 5 équations linéaires en 7 inconnues $(x_i)_{1\leq i\leq 7}$

$$\left\{\begin{array}{rcrcrcrcrcrcrcl} 4x_1 & + & 2x_2 & + & x_3 & + & 4x_4 & + & 5x_5 & + & 6x_6 & + & 7x_7 & = & 1 \\ x_1 & + & 4x_2 & - & 6x_3 & + & 2x_4 & + & 2x_5 & & & - & 2x_7 & = & 2 \\ x_1 & - & 2x_2 & + & 3x_3 & + & 3x_4 & - & 5x_5 & + & 6x_6 & + & x_7 & = & -10 \\ x_1 & - & x_2 & + & x_3 & - & x_4 & + & 2x_5 & - & 3x_6 & - & x_7 & = & -2 \\ x_1 & + & x_2 & + & & & 8x_4 & + & 2x_5 & + & 3x_6 & + & 4x_7 & = & 3. \end{array}\right. \quad (1)$$

1. Reformuler le problème sous forme matricielle $AX = b$.
2. Définir sous scilab la matrice $M \in \mathcal{M}_{5,8}(\mathbb{R})$ obtenue par adjonction du vecteur colonne b à la matrice A ; ce qui signifie que pour $1 \leq i \leq 5$, $M_{i8} = b_i$.

   C'est à la matrice M que chaque itération de la méthode du pivot va être appliquée. Le second membre b de (1) doit en effet également être modifié. Dans cet exercice, les premières étapes de la méthode sont détaillées. On adopte dans la suite la notation suivante : $M = (L_1, \cdots, L_5)^T$ où, pour $1 \leq i \leq 5$, L_i désigne le vecteur ligne associé à la i-ème ligne de M.

3. *Première itération* du pivot. Effectuer sous scilab la série d'opérations suivante
   - $L_1 \leftarrow L_1/a_{11}$;
   - $L_i \leftarrow L_i - a_{i1}L_1$, pour $2 \leq i \leq 5$.
4. *Deuxième itération* du pivot. Effectuer sous scilab la série d'opérations suivante
   - $L_2 \leftarrow L_2/a_{22}$;
   - $L_i \leftarrow L_i - a_{i2}L_2$, pour $3 \leq i \leq 5$.
5. Réaliser sous scilab les itérations suivantes jusqu'à obtenir une forme satisfaisante. Expliciter la forme générique des solutions du système (1).
6. Appliquer à cet exemple la fonction `linsolve` introduite dans l'exercice 3.

Exercice 5 Fonction équivalente

1. Ecrire une fonction scilab qui généralise l'algorithme décrit dans l'exercice précédent. L'algorithme prend pour paramètre une matrice $M \in \mathcal{M}_{n,m}(\mathbb{R})$ et renvoie une matrice $N \in \mathcal{M}_{n,m}(\mathbb{R})$ triangulaire supérieure. On rappelle ci-dessous la syntaxe scilab pour définir une fonction :

   ```
   function N= PIVOT(M)
   ...
   endfunction.
   ```

2. Appliquer cette fonction à la matrice M définie dans l'exercice 4. Vérifier que l'on obtient bien le même résultat.
3. Générer sous scilab via la fonction `rand` une matrice $Q \in \mathcal{M}_{20,30}$, dont les composants sont tous compris entre 0 et 1. Appliquer la méthode du pivot au système $QX = d$ où $d \in \mathbb{R}^{20}$ vérifie $d_i = i$ pour $1 \leq i \leq 20$.

Programmation linéaire et convexité
TP1 correction

17 février 2010

1 Calcul matriciel sous scilab

Exercice 1

```
// Exercice 1

//Question 1
A=[4,6,-2,3;2,-1,0,1;-7,0,1,12]

//Question 2
A(1:2,:)=2*A(1:2,:)
A(:,3)=1.0/3*A(:,3)

//Question 3
B=[4:6;5:5:15;ones(1,3)];

//Question 4
C=A(:,1:$-1)

//Question 5
D=B*A; E=C.*B;

//Question 6
sum(E)
Y=sum(D,'c')
```

Exercice 2

1. A est diagonalisable car symétrique. En conséquence, il existe une matrice $P \in \mathcal{M}_4(\mathbb{R})$ telle que $A = P^T D P$, où $D \in \mathcal{M}_4(\mathbb{R})$ est la matrice diagonale contenant les valeurs propres de A. On appelle la fonction spec via la commande suivante.

   ```
   A=[4,5,6,-1;5,10,15,2;6,15,1,4;-1,2,4,-2];
   [P,D]=spec(A);
   ```

2. Une fois que l'on dispose de la décomposition de A, la méthode la plus simple consiste à inverser D. Mais on peut également utiliser la fonction `inv` qui ne s'applique qu'à des matrices carrées.

   ```
   INV1= P* (1./D)*P';
   INV2= inv(A);
   ```

3. La question n'est pas très différente. Il faut prendre soin lors de la définition de A, que l'on ne peut plus remplir à la main lorsque n est grand.

   ```
   n=10;
   A=2*diag(ones(1,n));
   ```

```
T=diag(ones(1,n-1));
A(2:$,1:$-1)=A(2:$,1:$-1)-T;
A(1:$-1,2:$)=A(1:$-1,2:$)-T
[P,D]=spec(A)
```

Puis comparaison avec la formule exacte.

```
D1=4*diag(sin(1.0/(2*(n+1))*%pi*(1:n)).^2);
D-D1
```

Exercice 3

1. La matrice est bien évidemment de rang 3.
2. Pour ces deux questions, il s'agit simplement d'expérimenter les fonctions scilab correspondantes qui renvoient respectivement

```
-2->rank(A)
 ans  =

    3.
-2->kernel(A)
 ans  =

  - 0.7571633  - 0.4948592
  - 0.2735437    0.4185377
  - 0.2735437    0.4185377
  - 0.0211559    0.5834908
    0.5259315  - 0.2535847
```

3. La fonction `linsolve` prend deux arguments et résout de manière systématique les systèmes linéaires. Et ce même lorsque la matrice A est quelconque. L'espace affine des solutions se présente donc sous la forme $x_0 \otimes \mathrm{Ker}\ A$. `linsolve` renvoie la solution de référence x_0 et une base de Ker A.

```
-2->[x0,Ker]=linsolve(A,b)
 Ker  =

  - 0.5630462  - 0.7079271
    0.3913214  - 0.3112355
    0.3913214  - 0.3112355
    0.5790035  - 0.0752598
  - 0.2036394    0.5472112
 x0  =

  - 0.5
    1.
  - 0.0
  - 1.2
  - 0.2
```

   Les noyaux sont effectivement identiques.

4. Idem. Les matrices sont de rang 4. `linspace` renvoie donc dans chaque cas une solution de référence base des noyaux de dimensions respectives 4 et 2.

2 Systèmes linéaires et pivôt de Gauss-Jordan

Exercice 4

1. Codes scilab.

```
//Question 1 et 2

A=[4,2,1,4,5,6,7;1,4,-6,2,2,0,-2;1,-2,3,3,...
-5,6,1;1,-1,1,-1,2,-3,-1;1,1,0,8,2,3,4]
b=[1;2;-10;-2;3]
M=[A,b]

// Question 3

// Premiere iteration
M(1,:)=M(1,:)*1.0/M(1,1)
M(2,:)=M(2,:)-M(2,1)*M(1,:)
M(3,:)=M(3,:)-M(3,1)*M(1,:)
M(4,:)=M(4,:)-M(4,1)*M(1,:)
M(5,:)=M(5,:)-M(5,1)*M(1,:)

// Question 4

//Deuxieme iteration
M(2,:)=M(2,:)*1.0/M(2,2)
M(3,:)=M(3,:)-M(3,2)*M(2,:)
M(4,:)=M(4,:)-M(4,2)*M(2,:)
M(5,:)=M(5,:)-M(5,2)*M(2,:)

// Question 5

//Troisieme Iteration
M(3,:)=M(3,:)*1.0/M(3,3)
M(4,:)=M(4,:)-M(4,3)*M(3,:)
M(5,:)=M(5,:)-M(5,3)*M(3,:)

//Quatrieme iteration
M(4,:)=M(4,:)*1.0/M(4,4)
M(5,:)=M(5,:)-M(5,4)*M(4,:)

//Cinquieme iteration
M(5,:)=M(5,:)/M(5,5)
```

2. Sorties correspondantes. Ne figure dans ce document que le premier chiffre après la virgule. Scilab travaillant avec des doubles, il y a en réalité plus de chiffres après la virgule. Le nombre d'entre eux affichés peut être modifié via la fonction `format`.

```
-1-> A   =

   4.    2.    1.    4.    5.    6.    7.
   1.    4.  - 6.    2.    2.    0.  - 2.
   1.  - 2.    3.    3.  - 5.    6.    1.
   1.  - 1.    1.  - 1.    2.  - 3.  - 1.
   1.    1.    0.    8.    2.    3.    4.
b  =

   1.
   2.
 - 10.
 - 2.
   3.
M  =
```

```
    4.    2.    1.    4.    5.    6.    7.    1.
    1.    4.  - 6.    2.    2.    0.  - 2.    2.
    1.  - 2.    3.    3.  - 5.    6.    1.  - 10.
    1.  - 1.    1.  - 1.    2.  - 3.  - 1.  - 2.
    1.    1.    0.    8.    2.    3.    4.    3.
M  =

    1.    0.5    0.2    1.    1.2    1.5    1.8    0.2
    0.    3.5  - 6.2    1.    0.8  - 1.5  - 3.8    1.8
    0.  - 2.5    2.8    2.  - 6.2    4.5  - 0.8  - 10.
    0.  - 1.5    0.8  - 2.    0.8  - 4.5  - 2.8  - 2.2
    0.    0.5  - 0.2    7.    0.8    1.5    2.2    2.8
M  =

    1.    0.5    0.2    1.    1.2    1.5    1.8    0.2
    0.    1.   - 1.8    0.3   0.2  - 0.4  - 1.1    0.5
    0.    0.   - 1.7    2.7 - 5.7    3.4  - 3.4  - 9.
    0.    0.   - 1.9  - 1.6   1.1  - 5.1  - 4.4  - 1.5
    0.    0.     0.6    6.9   0.6    1.7    2.8    2.5
M  =

    1.    0.5    0.2    1.    1.2    1.5    1.8    0.2
    0.    1.   - 1.8    0.3   0.2  - 0.4  - 1.1    0.5
    0.    0.     1.   - 1.6   3.3  - 2.     2.     5.2
    0.    0.     0.   - 4.6   7.5  - 9.   - 0.5    8.6
    0.    0.     0.     7.9 - 1.5    3.     1.5  - 0.9
M  =

[More (y or n) ?]

    1.    0.5    0.2    1.    1.2    1.5    1.8    0.2
    0.    1.   - 1.8    0.3   0.2  - 0.4  - 1.1    0.5
    0.    0.     1.   - 1.6   3.3  - 2.     2.     5.2
    0.    0.     0.     1.  - 1.6    1.9    0.1  - 1.9
    0.    0.     0.     0.    11.  - 12.    0.6    14.
M  =

    1.    0.5    0.2    1.    1.2    1.5    1.8    0.2
    0.    1.   - 1.8    0.3   0.2  - 0.4  - 1.1    0.5
    0.    0.     1.   - 1.6   3.3  - 2.     2.     5.2
    0.    0.     0.     1.  - 1.6    1.9    0.1  - 1.9
    0.    0.     0.     0.    1.   - 1.1    0.1    1.2
```

Exercice 5

1. Code de la fonction PIVOT

```
function N= PIVOT(M)
  s=size(M);
  l=s(1)
  for i=1:l
    M(i,:)=M(i,:)/M(i,i);
    for k=i+1:l
      M(k,:)=M(k,:)-M(k,i)*M(i,:);
    end
  end
```

```
  N=M
endfunction
```

2. Le résultat de la procédure d'élimination est bien entendu le même.
3. Définition des matrices

```
Q=rand(20,30)
d=1:20
M=[Q;d]
N=PIVOT(M)
```