**Feuille de TP n° 8**

# Matrices et systèmes avec Scilab

Ce TP accompagne le chapitre 6 (Informatique et Algorithmique) : **Matrices avec Scilab**.
Créez le dossier `..\ECS1B_TPInfo\TP8\` et faites-en le répertoire courant de Scilab. Tous les scripts devront être sauvegardés dans ce dossier.

## I Entrainement

### 1) Construction et modification d'une matrice

• Exécuter les commandes suivantes dans la console :

```
-->u=[-1,0,1,2,3]
-->v=[1;3;-2;5;0;2]
-->A=[7,8;9,10]
-->B=[1/7,2/5,-1/8,1;0,2,1/2,1/3;3,0,1/9,2]
-->L1=[2,-1,0,3,-4]; L2=[1,-8,9,-4,7]; L3=[0,0,7,-1,2]; M=[u;L1;L2;L3]
-->C1=[1;-1]; C2=[-2;3]; C3=[0;-4]; C4=[2;0]; N=[C1,C2,C3,C4]
-->[N,A;u,11]
```

• Construire les matrices suivantes avec Scilab : $A = \begin{pmatrix} 7 & 7 & -5 \\ -1 & 0 & -2 \end{pmatrix}$, $B = \begin{pmatrix} 6 & -4 \\ 1 & 3 \end{pmatrix}$,

$$X = \begin{pmatrix} 1 \\ -1 \\ 0 \\ 2 \\ -3 \\ -10 \end{pmatrix}, \quad C = \begin{pmatrix} 2 & -1 & 0 & 0 & 1 \\ 3 & 4 & -3 & 7 & 0 \\ 1 & -1 & 2 & 1 & -3 \\ 4 & -1 & 0 & 1 & 2 \end{pmatrix} \quad \text{et} \quad M = \begin{pmatrix} 2 & -1 & 0 & 0 & 1 & 1 \\ 3 & 4 & -3 & 7 & 0 & -1 \\ 1 & -1 & 2 & 1 & -3 & 0 \\ 4 & -1 & 0 & 1 & 2 & 2 \\ 6 & -4 & 7 & 7 & -5 & -3 \\ 1 & 3 & -1 & 0 & -2 & -10 \end{pmatrix}$$

On construira la matrice $M$ par blocs à partir des matrices $A$, $B$, $C$ et $X$ (pas nécessairement avec une seule commande).

• Exécuter les commandes suivantes dans la console :

```
-->size(A), size(B), size(X), size(C), size(M)
-->len(A), len(B), len(X), len(C), len(M)
-->M(4,2), M(2,4), M(5,3), M(3,5)
```

```
-->M(2,:), M(5,:)
-->M(:,1), M(:,4)
-->M([3,2],[1,5,6])
```

• Exécuter les commandes suivantes dans la console :

```
-->linspace(1,9,20)
-->1:2:20
-->zeros(3,6)
-->ones(5,2)
```

```
-->eye(3,3), eye(5,8), eye(8,5)
-->diag([1,-2,3,-4,5])
-->rand(4,7)
```

• Nous allons maintenant modifier des coefficients de la matrice $M$. Exécuter les commandes suivantes dans la console :

```
-->M(1,1)=21, M(5,6)=-13
-->M(4,:)=[], size(M)
-->M(:,1)=[0;0;1;0;0]
-->M(:,[2,3])=[], size(M)
```

```
-->M=[[0;1;0;1;0],M], size(M)
-->M=[M;[1,2,3,4,5]]
-->M([2,3],[1,4,5])=rand(2,3)
```

• Construire en Scilab :
  - la matrice élémentaire $E_{5,12} \in \mathcal{M}_{8,20}(\mathbb{R})$.
  - la matrice de $\mathcal{M}_{11,7}(\mathbb{R})$ dont tous les coefficients sont égaux à $1$ sauf ceux se trouvant sur les lignes $2, 5, 6, 10$ et ceux sur les colonnes $3, 7$ qui valent $0$.
  - la matrice de $\mathcal{M}_{11,7}(\mathbb{R})$ dont tous les coefficients sont égaux à $1$ sauf ceux se trouvant à la fois sur les lignes $2, 5, 6, 10$ et sur les colonnes $3, 7$ qui valent $0$.
  - une matrice diagonale d'ordre $7$ dont les coefficients sont tirés aléatoirement entre $0$ et $1$.

## 2) Opérations sur les matrices

• Exécuter les commandes suivantes dans la console :

```
-->A=[1,0,1;0,1,0], B=[1,2,3;-4,-5,-6]
-->C=[2,4,0,1;1,5,-6,8;0,-2,0,4]
-->A', B', C'
-->A+B, A-B
-->3*A, A/4
-->A.*B, A./B, B.^A, A.^3
```

```
-->A*B
-->B^A
-->A^3
-->A*C, B*C
-->C*A
-->M=[1,2;-3,5]; M*M, M^2
```

Insistons bien sur la différence entre :
  - le produit et la puissance terme à terme sur des matrices de même taille (avec `.*` et $.\wedge$).
  - et le produit matriciel (avec des matrices de tailles compatibles avec le produit) et la puissance de matrices carrées (avec `*` et $\wedge$).

• Exécuter les commandes suivantes dans la console :

```
-->A=[1,2,-1;3,6,-4;4,3,0;0,1,2;-6,8,1]
-->B=[1,2,3;-1,5,-4;7,-2,2;1,-1,2;0,2,0]
-->A<B, find(A<B)
-->A==B, find(A==B)
```

• Exécuter les commandes suivantes dans la console :

```
-->A=[1,2,3;-4,-5,-6];
-->min(A), max(A)
-->sum(A), prod(A)
```

## 3) Inversion de matrices et résolution de $AX = B$

• Exécuter les commandes suivantes dans la console :

```
-->A=[-3,-2,-5,-3;-2,0,-1,-2;-2,-3,-5,-3;-1,-4,-4,-2]
-->B=inv(A)
-->A*B, B*A
```

Que fait la commande `inv(A)` ?

• Exécuter les commandes suivantes dans la console :

```
-->C=[2,0,-1,-1,-3;4,2,-4,3,2;-2,1,-1,-3,5]
-->inv(C)
-->rank(C)
```

```
-->K=kernel(C)
-->C*K
-->rank(A)
-->kernel(A)
```

Que font les commandes `rank(C)` et `kernel(C)` ?

• Si `M` est une matrice implémentée en Scilab, que renvoie la commande

`rank(M)+size(kernel(M),'c')-size(M,'c')` ?

Pourquoi ? Tester-la avec les matrices `A` et `B`.

• Exécuter les commandes suivantes dans la console :

```
-->A=[-3,-2,-5,-3;-2,0,-1,-2;-2,-3,-5,-3;-1,-4,-4,-2], B=[-1;4;-2;-3]
-->X=inv(A)*B
-->[X0,K]=linsolve(A,-B)
```

Que fait la commande `linsolve(A,-B)` ? Cette commande n'a pas grand intérêt dans le cas où `A` est inversible. Par contre, si ce n'est pas le cas :

```
-->A=[2,0,-1,-1,-3;4,2,-4,3,2;-2,1,-1,-3,5], B=[-9;-1;7]
-->[X0,K]=linsolve(A,-B)
-->A=[-2,4,-1;-5,-3,4;8,10,-9], B=[2;3;-1]
-->[X0,K]=linsolve(A,-B)
```

# II Exercices

*Commencez par taper* `clear` *dans la console. Pour vous y retrouver plus facilement, veuillez écrire*

- *en commentaire au début de chaque fonction le numéro du TP et de l'exercice concerné.*
- *sur cette feuille de TP, le nom que vous avez donné à a fonction (extension en* `.sci`*).*

*Vous testerez dans la console Scilab toutes les fonctions que vous écrirez (sans oublier de les exécuter au préalable).*

**Exercice 1.** Implémenter en Scilab une fonction qui prend en entrée deux entiers naturels $n$ et $p$ strictement positifs et qui renvoie la matrice de taille $n \times p$ dont les coefficients sont les entiers de $1$ à $np$ (rangés par ordre croissant ligne par ligne).

**Exercice 2.** Soit $A = \begin{pmatrix} 3 & 1 & -2 \\ 0 & -1 & 0 \\ -4 & -5 & -3 \end{pmatrix}$. Écrire un programme en Scilab qui demande un entier naturel $n$ à l'utilisateur et qui calcule $A^n$ (sans utiliser la commande $\wedge$). Peut-on calculer $A^n$ si $n \in \mathbb{Z}\backslash\mathbb{N}$ ? Si oui, modifier le programme en conséquence.

**Exercice 3.** Résoudre les systèmes linéaires suivants à l'aide de Scilab :

1) $\begin{cases} x + \frac{y}{2} - \frac{z}{2} = \frac{5}{2} \\ \frac{x}{3} + \frac{5y}{3} + \frac{3z}{2} = \frac{7}{3} \\ \frac{x}{3} - \frac{4y}{3} - \frac{11z}{6} = -\frac{1}{2} \end{cases}$

2) $\begin{cases} 3y - 4z = -7 \\ 5x + 2z = 4 \\ -2x - 4y = 3 \end{cases}$

3) $\begin{cases} 2x + 5z = 2 \\ x + y + z = 3 \\ + 4y - 6z = 8 \\ -x - 3y + 2z = -7 \end{cases}$

4) $\begin{cases} x - 2y + 4z = 1 \\ 3x - y - z - 2t = 0 \\ -5x + 6z + 4t = 1 \\ 2x - 4y + 8z = -1 \end{cases}$

5) $\begin{cases} \frac{x}{5} + y - z = \frac{1}{3} \\ \frac{x}{4} - \frac{y}{2} - \frac{z}{3} = -\frac{1}{2} \end{cases}$

6) $\begin{cases} -5x - y + 3z + 3t + 6u = -4 \\ 3x + 5y + z - 2u = 2 \\ 2x + 8y + 5z + 6t - 7u = 5 \\ -4x - 2y + 3z + 6t - 3u = 1 \\ 2x + 6y + z - 3t + 7u = -3 \end{cases}$

7) $\begin{cases} -ix - y + z = 2 - i \\ + (1+i)y + 2z = 1 - i \\ x - iy + (1-i)z = 2 \\ x + (1-2i)y - iz = i \end{cases}$

8) $\begin{cases} ix + 2y - z = -1 \\ 3x - 4iy = i \\ ix + 7y + 3iz = 2 \end{cases}$

Comparer avec les résultats théoriques que l'on a déterminés dans la feuille d'exercices n° 16.

*Pour préparer la section suivante, veuillez sauvegarder les commandes qui implémentent en Scilab les matrices et vecteurs associés à ces systèmes (par exemple en faisant des copier-coller dans un fichier* `.txt`*).*

# III Algorithme du pivot de Gauss

Voici une fonction en Scilab qui prend en argument une matrice $A$ et renvoie une matrice échelonnée obtenue via l'algorithme de Gauss appliqué à la matrice $A$ :

```
function A=PivotGauss(A)
  [n,p]=size(A);
  for  j=1:min(n-1,p)
       //On recherche le premier pivot non nul pour la colonne j :
       pivot=0; i0=j-1;
       while  (pivot==0)&(i0<n)
              i0=i0+1;
              pivot=A(i0,j);
       end
       //Ce premier pivot non nul (s'il existe) est en position i0.
       if  pivot<>0  then
            //On échange les lignes i0 et j :
            [                                ]
            //On annule les coefficients de la colonne j
            //dans les lignes suivant la ligne j :
            for  k=(j+1):n
                 [                                ]
            end
       end
  end
endfunction;
```

1) Compléter cette fonction Scilab et implémenter-la dans un script nommé `PivotGauss.sci`. Tester-la avec les matrices associées aux systèmes de l'exercice 3.
2) Modifier la fonction afin que :
   - elle prenne en argument deux matrices $A$ et $B$ (ayant la même nombre de lignes),
   - elle effectue les mêmes opérations élémentaires sur les lignes de $B$ que sur les lignes de $A$,
   - elle renvoie une matrice échelonnée $A_0$ et une matrice $B_0$ de telle sorte que, si $B$ est un vecteur colonne, alors le système $AX = B$ est équivalent au système $A_0X = B_0$.

   Nommer `PivotGauss2.sci` cette fonction. Tester-la fonction avec les matrices et vecteurs associés aux systèmes de l'exercice 3.

# IV Algorithme de Gauss-Jordan

Si `A` est une matrice carrée implémentée en Scilab, alors la commande

```
n=size(A,'r'); [A0,B0]=PivotGauss2(A,eye(n,n))
```

transforme `A` en matrice échelonnée `A0` via l'algorithme du Pivot de Gauss et réalise les mêmes opérations sur la matrice identité pour la transforme en la matrice `B0`. Tout est prêt pour implémenter l'algorithme de Gauss-Jordan permettant d'inverser une matrice :

```
function A=GaussJordan(A)
  //On commencer par mettre A sous forme échelonnée
  //et pas faire les mêmes opérations élémentaires sur B.
  n=size(A,'r')
  [A,C]=PivotGauss2(A,eye(n,n));
  //On calcule le produit des termes diagonaux de A.
  d=prod(diag(A));
  //A est inversible si et seulement si ce produit est non nul.
  if  d==0  then
      disp('La matrice n"est pas inversible')
  else
      B=C;
      for  i=0:(n-2)
           a=A(n-i,n-i);
           //On divise la ligne n-i par a pour obtenir
           //un coefficient diagonal unitaire.
           A(n-i,:)=A(n-i,:)/a; B(n-i,:)=B(n-i,:)/a;
           for  k=1:(n-i-1)
                //On annule tous les coefficients de la ligne k
                //sauf celui sur la diagonale qui vaut 1.
                B(k,:)=
                A(k,:)=
           end
      end
      B(1,:)=B(1,:)/A(1,1); A(1,:)=A(1,:)/A(1,1);
  end
endfunction;
```

Compléter cette fonction Scilab et implémenter-la dans un script nommé `GaussJordan.sci`. Tester-la avec les matrices suivantes :

$$\begin{pmatrix} -4 & 2 & 3 \\ -5 & 4 & 2 \\ -2 & 3 & 0 \end{pmatrix}, \quad \begin{pmatrix} -1 & -3 & 2 \\ -2 & 1 & 0 \\ 8 & 3 & -4 \end{pmatrix} \quad \text{et} \quad \begin{pmatrix} -1 & 1 & 3 & -1 \\ 4 & 3 & 4 & -3 \\ -1 & -2 & -3 & 1 \\ 3 & 2 & 3 & -2 \end{pmatrix}.$$

Comparer avec la fonction `inv()`.