

1. Vecteurs

1.1 Crédation de vecteurs

Par défaut, le vecteur est une ligne à plusieurs colonnes

a) *Vecteur ligne par énumération des composantes :*

```
--> v=[1 3.4 5 -6] // Les valeurs sont séparées par des espaces
```

v =

1. 3.4 5. -6.

Ou encore

```
--> w =[1,5,-7.5] // Les valeurs sont séparées par des virgules (,)
```

w =

1. 5. -7.5

b) *Vecteur ligne par description :*

```
--> x = [0 : %pi/10 : %pi] // [valeur-initiale : incrément : valeur-finale]
```

x =

column 1 to 6

0. 0.3141593 0.6283185 0.9424778 1.2566371 1.5707963

column 7 to 11

1.8849556 2.1991149 2.5132741 2.8274334 3.1415927

c) *Vecteur colonne :*

```
--> xcol=x'
```

xcol =

0.
0.3141593
0.6283185
0.9424778
1.2566371
1.5707963
1.8849556
2.1991149
2.5132741
2.8274334
3.1415927

Ou encore :

```
--> xcol = [9;8;7;6;5;4;3]
```

xcol =

9.
8.
7.
6.
5.
4.
3.

d) Génération de vecteurs métriques

```
--> x = linspace(0, %pi, 11) // génère le même x que ci-dessus (11 valeurs réparties de 0 à pi
```

```
x =
```

```
column 1 to 6
```

```
0. 0.3141593 0.6283185 0.9424778 1.2566371 1.5707963
```

```
column 7 to 11
```

```
1.8849556 2.1991149 2.5132741 2.8274334 3.1415927
```

```
--> logspace(0, 2, 11) // crée un vecteur log à 11 composantes entre  $10^0$  et  $10^2$ 
```

```
ans =
```

```
column 1 to 7
```

```
1. 1.5848932 2.5118864 3.9810717 6.3095734 10. 15.848932
```

```
column 8 to 11
```

```
25.118864 39.810717 63.095734 100.
```

1.2 Adressages et indexages

```
--> x(3) // 3ème élément du vecteur x
```

```
ans =
```

```
0.6283185
```

```
--> x(2 : 4) // un bloc de composantes
```

```
ans =
```

```
0.3141593 0.6283185 0.9424778
```

```
--> x([8 3 9 1]) // une sélection de composantes (on les désigne avec un autre vecteur!)
```

```
ans =
```

```
2.1991149 0.6283185 2.5132741 0.
```

1.3 Combinaison de vecteurs

a) Accolage de deux vecteurs :

```
--> a = [1:3]
```

```
a =
```

```
1. 2. 3.
```

```
--> b=[10:10:30]
```

```
b =
```

```
10. 20. 30.
```

```
--> c = [a b]
```

```
c =
```

```
1. 2. 3. 10. 20. 30.
```

On peut faire plus compliqué :

```
--> d=[a(2:-1:1) b] // on accolé b avec une portion de a dans l'ordre renversé
```

```
d =
```

```
2. 1. 10. 20. 30.
```

Notez la différence entre () et [] :

Tableau 1 : Notations

[]	Énumération d'éléments
:	Descripteur d'éléments de vecteur/matrice
()	Ensemble d'arguments
,	Séparateur d'arguments
;	Séparateur des lignes dans les matrices Suppression de l'affichage du résultat de l'évaluation d'une instruction (sans écho)
'	Transposition de matrice
.	Force l'opérateur à s'appliquer sur chaque élément du vecteur/matrice
//	Commentaire
...	Continuation de l'instruction sur la ligne suivante

2. Matrices

2.1 Création de matrices

Une matrice $n \times p$ est un tableau à **n lignes** et **p colonnes**, dont les coefficients (les valeurs de chaque case) sont des nombres. Scilab est un langage fondé sur la manipulation des matrices. Par exemple **un nombre décimal** est considéré par Scilab comme **une matrice 1×1** .

a) *Par énumération des éléments*

```
--> m1 = [1 2 3 ; 4 5 6 ; 7 8 9]      // On sépare les lignes par des point-virgule
m1 =
  1. 2. 3.
  4. 5. 6.
  7. 8. 9.
```

On peut étendre aux matrices les autres manières de définir des vecteurs.
Par exemple :

```
--> m2 = [1:1:3 ; 11:1:13]    // Par description
m2 =
  1. 2. 3.
  11. 12. 13.
```

```
--> m3 = [1:1:3 ; logspace(0, 1, 3)] // Par description et par génération de matrices métriques
m3 =
  1. 2.      3.
  1. 3.1622777 10.
```

2.2 Transposition

L'opérateur apostrophe est utilisé pour créer un vecteur colonne est en fait l'opérateur transposition :

```
--> m2'
```

```
ans =
```

1. 11.
2. 12.
3. 13.

Opérations scalaires-matrices

Une telle opération agit sur chaque élément de la matrice :

```
--> m2*10 // De même : 4*m2 m2-10 m2/4
```

```
ans =
```

```
10. 20. 30.  
110. 120. 130.
```

Une exception :

```
--> m2^2
```

à la ligne 20 de la fonction %s_pow (C:\Program Files\scilab-6.0.0\modules\ast\macros\%s_pow.sci ligne 32)
à la ligne 3 de la fonction %s_p_s (C:\Program Files\scilab-6.0.0\modules\ast\macros\%s_p_s.sci ligne 15)

%s_pow : Dimension erronée de l'argument d'entrée n°1 : Une matrice carrée attendue.

La solution est l'usage du point qui force l'opération sur chaque élément :

```
--> m2.^2
```

```
ans =
```

```
1. 4. 9.  
121. 144. 169.
```

2.3 Opérations entre matrices

a) Multiplications

--> m1 // rappelons la définition de m1

m1 =

1. 2. 3.
4. 5. 6.
7. 8. 9.

--> m2 // rappelons la définition de m2

m2 =

1. 2. 3.
11. 12. 13.

--> m1 * m2

Dimensions ligne/colonne incohérentes. message d'erreur

Par contre **m1*m2'** (m2' étant la transposée de m2), est possible !

--> m2'

ans =

1. 11.
2. 12.
3. 13.

--> m1 * m2' // le produit matriciel n'est possible que lorsque les dimensions sont cohérentes
ans =

14. 74.
32. 182.
50. 290.

Donc pour faire le produit de 2 matrices, il faut que le nombre de colonne de 1^{ième} soit égale au nombre de ligne de la 2^{ème} matrice !

Multiplication élément par élément :

--> m2 .*m3 // (m2 et m3 ont la même dimension)

ans =

1. 4. 9.
11. 37.947332 130.

b) Divisions

--> m2/m3 // Division matricielle à droite

ans =

1. 0.
9.5406116 -1.5959617

--> m2\m3 \\ Division matricielle à gauche

ans =

-0.5 -0.8256584 -0.45
0. 0. 0.
0.5 0.9418861 1.15

Division élément par élément :

--> m2./m3 // Chaque élément de m2 est divisé par l'élément équivalent de m3

ans =

```
1. 1. 1.  
11. 3.7947332 1.3
```

--> m2.\m3 // Chaque élément de m3 est divisé par l'élément équivalent de m2

ans =

```
1. 1. 1.  
0.0909091 0.2635231 0.7692308
```

--> m3./m2 // Chaque élément de m3 est divisé par l'élément équivalent de m2

ans =

```
1. 1. 1.  
0.0909091 0.2635231 0.7692308
```

2.4 Matrices particulières

--> ones(3,6)

ans =

```
1. 1. 1. 1. 1.  
1. 1. 1. 1. 1.  
1. 1. 1. 1. 1.
```

--> zeros(2,7)

ans =

```
0. 0. 0. 0. 0. 0. 0.  
0. 0. 0. 0. 0. 0. 0.
```

--> eye(4,4)

ans =

```
1. 0. 0. 0.  
0. 1. 0. 0.  
0. 0. 1. 0.  
0. 0. 0. 1.
```

--> diag(2:8)

ans =

```
2. 0. 0. 0. 0. 0. 0.  
0. 3. 0. 0. 0. 0. 0.  
0. 0. 4. 0. 0. 0. 0.  
0. 0. 0. 5. 0. 0. 0.  
0. 0. 0. 0. 6. 0. 0.  
0. 0. 0. 0. 0. 7. 0.  
0. 0. 0. 0. 0. 0. 8.
```

--> diag(2:2:8)

ans =

```
2. 0. 0. 0.  
0. 4. 0. 0.  
0. 0. 6. 0.  
0. 0. 0. 8.
```

3.6 Caractéristiques des matrices

--> size(m3) // dimensions

ans =

2. 3.

--> length(m3) // donne le nombre d'éléments de la matrice

ans =

6.

2.7 Quelques fonctions de manipulation des matrices

```
--> A =[1 2 3 ; 4 5 6 ; 7 8 9 ]  
A =  
1. 2. 3.  
4. 5. 6.  
7. 8. 9.  
  
--> diag(A) // extrait la diagonale de A  
ans =  
1.  
5.  
9.  
  
--> diag(ans) // travaille dans les 2 sens !  
ans =  
1. 0. 0.  
0. 5. 0.  
0. 0. 9.  
  
--> triu(A) // extrait le triangle supérieur de A  
ans =  
1. 2. 3.  
0. 5. 6.  
0. 0. 9.  
  
--> tril(A) // extrait le triangle inférieur de A  
ans =  
1. 0. 0.  
4. 5. 0.  
7. 8. 9.
```

Il y a encore bien d'autres fonctions pour travailler les matrices, voir la liste en TP.

2.8 Matrices clairsemées

Lorsque seulement quelques éléments d'une matrice sont non-nuls, on peut la définir comme une *sparse matrix*. Sa description contient seulement les éléments non nuls.

```
--> A_Qnormal = [0 5 0 ; 6 0 0; 0 0 7] // matrice normale  
A_normal =  
0. 5. 0.  
6. 0. 0.  
0. 0. 7.  
  
--> A_sparse = sparse(A_normal) // matrice clairsemée  
A_sparse =  
( 3, 3) sparse matrix  
( 1, 2) 5.  
( 2, 1) 6.  
( 3, 3) 7.  
  
--> B=full(A_sparse) // Matrice complète  
B =  
0. 5. 0.  
6. 0. 0.  
0. 0. 7.
```

full a convertit la matrice clairsemée en matrice complète.