9

Matrices

Les matrices sont les objets les plus importants et les plus couramment rencontrés avec Scilab. Ce type permet de représenter, de manière unifiée, les tableaux à une ou deux dimensions.

9.1. Création et modification

Il existe plusieurs méthodes pour créer des tableaux avec Scilab. La plus simple repose sur l'opérateur de concaténation `[]`. Pour créer une matrice, il vous suffit d'entrer la liste des valeurs, ligne par ligne, entre des crochets en respectant la syntaxe suivante :

- les valeurs d'une même ligne sont séparées par un espace ou une virgule (`,`) ;
- les lignes sont séparées par un point-virgule (`;`).

La saisie dans la console peut s'effectuer sur plusieurs lignes (à condition de ne pas oublier de refermer correctement la matrice).

Attention *› À la différence d'autres langages, la numérotation des cases/lignes/colonnes d'une matrice commence en Scilab à 1 (et pas à 0).*

La commande `size` permet de récupérer la taille d'une matrice (son nombre de lignes et de colonnes), le résultat de cette commande étant lui-même une matrice à une ligne et deux colonnes.

```
-->A=[1 2 3; 4 5 6; 7 8 9] // matrice 3 lignes 3 colonnes
 A  =

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

-->typeof(A) // même type que pour les nombres réels
 ans  =

 constant

-->size(A) // taille de A
 ans  =
```

```
    3.    3.

-->B=[10,11,12;15 14 13] // matrice à 2 lignes et 3 colonnes
 B  =

    10.    11.    12.
    15.    14.    13.

-->size(B) // taille de B
 ans  =

    2.    3.

-->// saisie sur plusieurs lignes

-->[1 2 3;
-->3 4 5]
 ans  =

    1.    2.    3.
    3.    4.    5.
```

Astuce *› La commande [ ] produit une matrice vide, à zéro ligne et zéro colonne. Un nombre (réel, complexe ou entier) est considéré comme une matrice à une ligne et une colonne.*

Il est possible ensuite de concaténer des matrices, toujours avec [ ], pour en fabriquer d'autres plus grandes à condition, bien sûr, qu'elles aient les bonnes dimensions.

```
-->A=[1 2 3; 4 5 6; 7 8 9];

-->B=[10,11,12;15 14 13];

-->C=[A;B]      // A au-dessus de B
 C  =

    1.     2.     3.
    4.     5.     6.
    7.     8.     9.
    10.    11.    12.
    15.    14.    13.

-->D=[B;A]      // B au-dessus de A
 D  =

    10.    11.    12.
    15.    14.    13.
    1.     2.     3.
    4.     5.     6.
    7.     8.     9.
```

```
-->E=[C,D]      // C à gauche de D
 E  =

    1.     2.     3.     10.    11.    12.
    4.     5.     6.     15.    14.    13.
    7.     8.     9.     1.     2.     3.
    10.    11.    12.    4.     5.     6.
    15.    14.    13.    7.     8.     9.

-->F=[D,C]      //D à gauche de C
 F  =

    10.    11.    12.    1.     2.     3.
    15.    14.    13.    4.     5.     6.
    1.     2.     3.     7.     8.     9.
    4.     5.     6.     10.    11.    12.
    7.     8.     9.     15.    14.    13.

-->G=[A,B]      // A à gauche de B -> erreur

    !--error 5

 Dimensions colonne/rangée incohérentes.
```

Vous pouvez aussi concaténer des matrices à l'aide de la commande `cat`.

```
-->A=[1 2 3; 4 5 6; 7 8 9]    // matrice 3 lignes 3 colonnes
 A  =

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

-->B=[10,11,12;15 14 13]    //matrice 2 lignes 3 colonnes
 B  =

    10.    11.    12.
    15.    14.    13.

-->// concaténations de matrices

-->C=[A;B]    // A au-dessus de B
 C  =

    1.     2.     3.
    4.     5.     6.
    7.     8.     9.
    10.    11.    12.
    15.    14.    13.

-->cat(1,A,B)    //=C
```

```
 ans  =

    1.     2.     3.
    4.     5.     6.
    7.     8.     9.
    10.    11.    12.
    15.    14.    13.

-->D=[B;A]    // B au-dessus de A
 D  =

    10.    11.    12.
    15.    14.    13.
    1.     2.     3.
    4.     5.     6.
    7.     8.     9.

-->cat(1,B,A)    //=D
 ans  =

    10.    11.    12.
    15.    14.    13.
    1.     2.     3.
    4.     5.     6.
    7.     8.     9.

-->E=[C,D]    // C à gauche de D
 E  =

    1.     2.     3.     10.    11.    12.
    4.     5.     6.     15.    14.    13.
    7.     8.     9.     1.     2.     3.
    10.    11.    12.    4.     5.     6.
    15.    14.    13.    7.     8.     9.

-->cat(2,C,D)    //=E
 ans  =

    1.     2.     3.     10.    11.    12.
    4.     5.     6.     15.    14.    13.
    7.     8.     9.     1.     2.     3.
    10.    11.    12.    4.     5.     6.
    15.    14.    13.    7.     8.     9.
```

Attention *› Les tailles des matrices que vous voulez concaténer doivent être compatibles, sinon vous obtiendrez une erreur (5 ou 6) lors de l'exécution, avec comme message* Dimension ligne/colonne incohérente.

Certaines matrices particulières peuvent être créées automatiquement, en donnant en entrée leur taille, à l'aide des fonctions suivantes :

- `zeros` produit une matrice nulle et `ones` une matrice remplie de 1 ;
- `eye` construit la matrice identité (avec des 1 sur la diagonale principale et des 0 ailleurs) ;
- `rand` remplit une matrice avec des coefficients pseudo-aléatoires uniformément répartis dans l'intervalle $[0;1[$ (voir aussi `grand`, pour générer des coefficients aléatoires selon d'autres lois, dans l'aide en ligne `help grand`).

```
-->O=zeros(2,3)    //matrice nulle
 O  =

    0.    0.    0.
    0.    0.    0.

-->U=ones(4,3)    //matrice de un
 U  =

    1.    1.    1.
    1.    1.    1.
    1.    1.    1.
    1.    1.    1.

-->Id=eye(3,3)    //matrice identité
 Id  =

    1.    0.    0.
    0.    1.    0.
    0.    0.    1.

-->M=rand(2,2)    //coefficients aléatoires
 M  =

    0.312641997    0.292226664
    0.361636101    0.566424882
```

Astuce *› Les fonctions* `zeros`, `ones`, `eye` *et* `rand` *appliquées à des matrices créent une matrice de même taille. En particulier, appliquées à un nombre, ces fonctions créent une matrice à une ligne et une colonne (donc juste un nombre !).*

```
-->A=rand(2,2)
 A  =

    0.8782165    0.5608486
    0.0683740    0.6623569

-->// créer matrice nulle de même taille que A

-->zeros(A)
```

```
 ans  =

    0.    0.
    0.    0.

-->// zeros(2) ne créé pas un vecteur à deux cases

-->zeros(2)
 ans  =

    0.
```

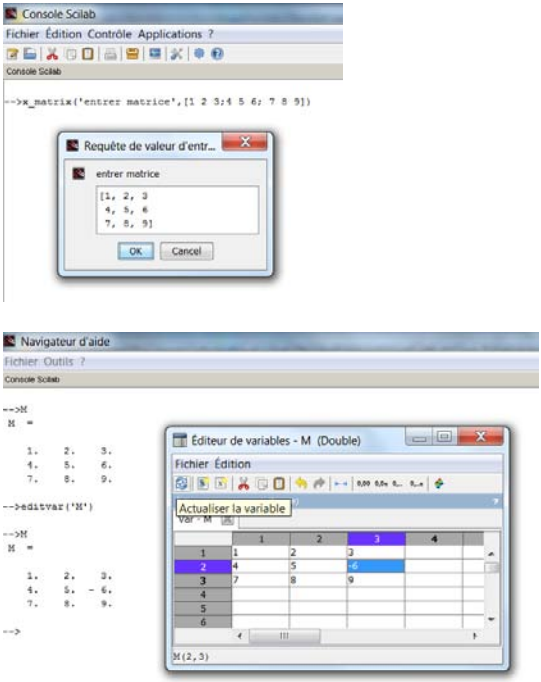
On accède aux différentes valeurs stockées dans une matrice en spécifiant entre parenthèses `()` la ligne et la colonne de la case désirée. Pour modifier une valeur d'une matrice, il suffit d'affecter la nouvelle valeur avec le `=` dans la case souhaitée.

```
-->A=[1 2 3; 4 5 6; 7 8 9]    // matrice 3 lignes 3 colonnes
 A  =

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

-->A(2,3)    // valeur sur la ligne 2 colonne 3
 ans  =

    6.

-->A(2,3)=-1    // modifie la valeur sur la ligne 2 colonne 3
 A  =

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

-->A(4,5)=10    // l'affectation augmente la taille de A
 A  =

    1.    2.    3.    0.    0.
    4.    5.  - 1.    0.    0.
    7.    8.    9.    0.    0.
    0.    0.    0.    0.    10.

-->    //case n'existant pas dans A

-->A(10,10)    // l'accès donne l'erreur 21

    !--error 21

 Index invalide.
```

Astuce *› Si on affecte, dans une matrice, une valeur dans une case qui n'existe pas (numéro de ligne/colonne supérieur à la taille de la matrice), Scilab agrandit automatiquement la taille de la matrice pour effectuer l'affectation demandée et remplit les autres cases créées avec des 0.*

Vous pouvez aussi modifier les coefficients d'une matrice par l'intermédiaire d'une interface graphique, avec la fonction `x_matrix` (voir Figure 9.1) ou en utilisant l'éditeur de variables avec la commande `editvar` (voir Figure 8.1 ou Figure 9.1).

Figure 9.1 : *Interface graphique pour modifier une matrice*



Attention *› Si vous tentez d'accéder à une valeur d'une matrice qui dépasse la taille de la matrice, vous obtiendrez une erreur (21) lors de l'exécution avec comme message* Index invalide. *C'est le cas en particulier lors d'un appel à une case/ligne/colonne de numéro 0 (ou négatif ou non entier).*

Les matrices ayant une seule ligne ou une seule colonne (qu'on appelle souvent des vecteurs) ne constituent pas un type particulier pour Scilab, cependant Scilab accepte une syntaxe simplifiée pour leur manipulation. On peut accéder aux coordonnées d'un vecteur en ne donnant qu'*un seul index au lieu de deux*. Pour calculer la taille d'un vecteur, la commande `length` qui calcule la longueur du vecteur, est plus adaptée que `size`.