Introduction à Scilab (2)

3.5 - Opérations arithmétiques sur les tableaux

Une fois un tableau défini, on peut réaliser certaines opérations arithmétiques sur l'ensemble des éléments d'un tableau ou entre tableaux de même dimensions.

Exemple : Testez les instructions suivantes :

```
M = [1 2 3;4 5 6]
M + 2 // ajoute 2 à tous les éléments de M
M - 3 // retranche 3 à tous les éléments de M
M * (-4) // multiplie tous les éléments de M par -4
M / 10 // divise tous les éléments de M par 10
```

On peut aussi effectuer des opérations arithmétiques terme à terme entre deux tableaux de même dimensions.

Exemple : Testez les instructions suivantes :

```
M = [1 2 3;4 5 6] , P = [4 7 1;0 2 8]
M + P , P + M , M - P , P - M
```

Par contre si on souhaite multiplier ou diviser terme à terme les éléments de deux tableaux, il faudra utiliser les opérateurs `.*` et `./` au lieu de `*` et `/`

Exemple : Testez les instructions suivantes :

```
M .* P , P .* M , P ./ M
1 ./ M   // les inverses des elements de M
```

⇨ il est conseillé de mettre un espace AVANT et APRÈS chaque opérateur.

De même, on peut appliquer la puissance terme à terme en utilisant l'opérateur `.^`

Exemple : Testez les instructions suivantes :

```
v1 = 0:9      // les 10 premiers entiers
v2 = v1 .^ 2 // les 10 premiers carrés
v3 = v1 .^ 3 // les 10 premiers cubes
w  = 2 .^ v1 // les 10 premieres puissances de 2
```

⇨ pour effectuer une opération entre deux tableaux, ils doivent nécessairement être de même dimensions.

Exemple : Testez les instructions suivantes :

```
u = [1 2 3 4 5 6] , v = [1 1 1 1]
u + v
u .* v
```

3.6 - Transposition

La transposition d'un tableau consiste en inversant le rôle des lignes et colonnes. Elle s'effectue avec le symbole `'` (`apostrophe` ou `quote`)

Exemple : Tapez les instructions suivantes :

```
v = 1:8 , w = [4;5;6;7] , M = [1 2 3;4 5 6]
```

puis l'instruction `v'` puis l'instruction `w'` puis l'instruction `M'`.

Exercice 1 :

en utilisant les instructions vues précédemment, écrire (si possible de la manière la plus simple) les instructions pour créer les tableaux suivants :

$$M1 = \begin{pmatrix} 1 & 3 & 5 & 7 \\ 2 & 4 & 6 & 8 \\ 3 & 5 & 7 & 9 \\ 4 & 6 & 8 & 10 \\ 5 & 7 & 9 & 11 \\ 6 & 8 & 10 & 12 \\ 7 & 9 & 11 & 13 \end{pmatrix} \qquad M2 = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 5 & 5 & -2 & -2 & -2 \\ 5 & 5 & -2 & -2 & -2 \end{pmatrix}$$

$$M3 = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 4 & 9 & 16 & 25 & 36 & 49 & 64 \end{pmatrix}$$

3.7 - Dimensions d'un tableau

A tout moment, il est possible de connaitre les dimensions d'un tableau avec les procédures `size` et `length`.

Exemple : Testez les instructions suivantes en observant le résultat après chacune d'elles :

```
v = 1:8 , w = [4;5;6;7] , M = [1 2 3;4 5 6] , t_vide = []
size(v) // donne les dimensions du tableau v : [nb_lig nb_col]
size(w) // donne les dimensions du tableau w
size(M) // donne les dimensions du tableau M
size(M,1) // donne le nombre de lignes de M
size(M,2) // donne le nombre de colonnes de M
length(v) // donne le nombre de valeurs du tableau v
length(w) // donne le nombre de valeurs du tableau w
length(M) // donne le nombre de valeurs du tableau M
length(t_vide) // donne le nombre de valeurs du tableau t_vide
```

⇨ On peut définir un tableau vide `[]` avec zéro valeur.

4 - Fonctions

Scilab fournit un certain nombre de fonctions notamment les principales fonctions mathématiques (voir la partie *Elementary functions* de l'aide en ligne).
La plupart des fonctions peuvent s'appliquer aussi bien à une seule valeur qu'à un tableau de valeurs.

Exemple : Testez les instructions suivantes :

```
sqrt(4) // racine carrée de 4
k = 0:12
sqrt(k) // les valeurs [sqrt(0),sqrt(1),sqrt(2),...,sqrt(12)]
t = k * %pi / 12
cos(t) // les valeurs [cos(0),cos(pi/12),cos(2*pi/12),...,cos(pi)]
```

On peut à partir des opérateurs et fonctions de Scilab, créer ses propres fonctions.

Exemple : Pour calculer les valeurs de la fonction $f(x) = \dfrac{1}{1+x^2}$ pour les valeurs $x = 0$, $x = 0,5$, $x = 1$, $\ldots x = 9,5$ et $x = 10$, tapez les instructions suivantes :

```
x = 0:0.5:10
y = 1 ./ (1 + x .^ 2) // espaces avant et après ./
```

On aimerait pouvoir définir la fonction $f(x)$ puis l'utiliser à l'aide de l'instruction `y=f(x)`.

Scilab permet à l'utilisateur de créer ses propres fonctions à l'aide de l'instruction `deff`, et éventuellement de les sauvegarder sous forme de fichier pour réutiliser ultérieurement.
Exemple : Avec l'éditeur de texte, créez le fichier suivant en le nommant `ex_fct1.sce`

```
// definition de la fonction f
deff("y = f(x)" , "y = 1 ./ (1 + x .^ 2)");

//----- utilisation de la fonction f -----

x = 0:0.5:10 // calcul de y = f(x) pour x=0 , x=0,5 ,
y = f(x)     // x=1  ... x=9,5  et  x=10
disp(y)

t = linspace(-1,2,1000) // calcul de f(t) pour 1000 valeurs de t
z = f(t)                // équiréparties entre -1 et 2
disp(z)
```

puis exécutez le script `ex_fct1.sce`.
⇨ il est important d'utiliser les opérations terme à terme (`.*`, `./`, `.^`) pour pouvoir utiliser la fonction avec un tableau de valeurs.
⇨ une fois la fonction définie, on peut l'utiliser avec n'importe quelle variable (pas nécessairement avec des variables ayant les mêmes noms que dans la définition de la fonction).
⇨ dans le cas d'un tableau avec un grand nombre de valeurs, Scilab demande à l'utilisateur de continuer ou non l'affichage du tableau dans la console.

Une ou plusieurs fonctions peuvent être définies dans un fichier-script puis être utilisées dans un autre fichier-script Scilab.
Exemple : Avec l'éditeur de texte, créez le fichier suivant en le nommant `mes_fonctions.sci`

```
// definition de la fonction f1
deff("y = f1(x)" , "y = (1) ./ (1 + x .^ 2)");

// definition de la fonction f2
deff("y = f2(t)" , "y = (t+1) .^ 2");
```

puis créez le fichier suivant en le nommant `ex_fct2.sce`

```
// chargement du contenu du fichier nommé mes_fonctions.sci
exec("mes_fonctions.sci", -1);

// calcul de f1(x) pour x=0  x=0,5  x=1  ... x=9,5  et  x=10
x = 0:0.5:10
y = f1(x)
disp(y)

// calcul de f2(t) pour 100 valeurs de t entre -10 et 10
t = linspace(-10,10,100)
z = f2(t)
```

```
disp(z)
```

puis exécutez le script `ex_fct2.sce`.

⇨ par convention, les fichiers contenant uniquement des définitions de fonctions ont un nom avec le suffixe `.sci` alors que les scripts Scilab ont un nom avec le suffixe `.sce`

Exercice 2 :

ajouter au fichier nommé `mes_fonctions.sci` la définition de la fonction $y = f_3(x) = \exp(\sqrt{x/10})$ puis écrire un script Scilab nommé `ex_fct3.sce` qui permet de calculer et afficher les deux vecteurs $u = f_3(t)\cos(t)$ et $v = f_3(t)\sin(t)$ pour 1000 valeurs de t entre 0 et 10.