Scilab : tableaux (matrices), vecteurs et scalaires

Dans Scilab, les données numériques sont structurées en tableaux.

Un peu de vocabulaire :

- **tableau** (et matrice) : tableau de nombres (deux dimensions ou plus)
- **vecteur** : **liste** de nombres (tableau à une seule ligne ou à une seule colonne, tableau à une dimension)
- **scalaire** : **un nombre** (tableau ou liste à un seul élément)

1. Extraction des données d'un tableau

Dans Scilab, un tableau (`array`) est une variable informatique qui permet de stocker des données numériques.

Considérons la variable `tab` contenant les informations suivantes :

```
tab = [0   0.1     0.2     0.3     0.4     0.5     0.6     0.7     0.8     0.9     1;
0   0.049   0.196   0.441   0.784   1.225   1.764   2.401   3.136   3.969   4.9;
0   0.98    1.96    2.94    3.92    4.9     5.88    6.86    7.84    8.82    9.8]
```

Le point virgule `;` représente le séparateur de lignes.

Copier-coller la définition dans la console Scilab.
Vous devez obtenir :

```
--> tab
 tab  =
0.   0.1     0.2     0.3     0.4     0.5     0.6     0.7     0.8     0.9     1.
0.   0.049   0.196   0.441   0.784   1.225   1.764   2.401   3.136   3.969   4.9
0.   0.98    1.96    2.94    3.92    4.9     5.88    6.86    7.84    8.82    9.8
```

Ce tableau (à deux dimensions) comporte 3 lignes et 11 colonnes (3x11 éléments).

```
--> size(tab)
 ans  =
3.   11.
```

Pour extraire l'élément (le scalaire) de la 2ème ligne et 4ème colonne :

```
--> tab(2, 4)
 ans  =
   0.441
```

Pour extraire la première ligne :

```
--> t = tab(1, :)
 t  =
0.   0.1   0.2   0.3   0.4   0.5   0.6   0.7   0.8   0.9   1.
```

`t` est un tableau à une dimension.
On dit aussi que `t` est un **vecteur ligne** (1x11).

```
--> size(t)
 ans  =
   1.   11.
```

Exercice 01 : Créer le vecteur ligne `x` qui contient la deuxième ligne du tableau `tab`, ainsi que le vecteur ligne `v` qui contient la troisième ligne du tableau `tab`.

```
--> x =
--> v =
```

Pour extraire la deuxième colonne (on obtient alors un **vecteur colonne** 3x1) :

```
--> tab(:, 2)
 ans  =
   0.1
   0.049
   0.98
```

Pour extraire la deuxième et la sixième colonne (on obtient alors un tableau 3x2) :

```
--> tab(:, [2 6])
ans  =
   0.1     0.5
   0.049   1.225
   0.98    4.9
```

Pour extraire les 5 premières colonnes des 2 dernières lignes (on obtient alors un tableau 2x5) :

```
--> tab(2:3, 1:5)
 ans  =
0.   0.049   0.196   0.441   0.784
0.   0.98    1.96    2.94    3.92
```

ou bien :

```
--> tab($-1:$, 1:5)
```

Le symbole `$` désigne le dernier élément d'une ligne ou d'une colonne.

Exercice 02 : Extraire les 3 dernières colonnes des 2 premières lignes du tableau `tab`.

```
--> tab(                )
```

Exercice 03 : Avec la fonction `plot()`, tracer la courbe x(t) puis v(t) (`t`, `x` et `v` désignent les vecteurs définis précédemment).

2. Opérations sur les tableaux et vecteurs

Pour créer une liste de valeurs régulièrement espacées, par exemple le vecteur ligne `[1  1.5  2  2.5  3]`, nous disposons de deux techniques :

```
--> x = 1:0.5:3    // début, intervalle, fin
```

ou :

```
--> x = linspace(1, 3, 5)    // début, fin, nombre d'éléments
```

2.1. Opération avec un scalaire

Supposons que nous voulions ajouter le scalaire 2 à chacun des éléments du vecteur `x`, c'est à dire obtenir le vecteur : `[3  3.5  4  4.5  5]`

1ère technique (à oublier rapidement)

```
x = linspace(1, 3, 5)
// vecteur vide
y = []
// boucle sur les éléments du vecteur x
for elt = x
    // ajout d'un nouvel élément dans le vecteur y
    y(1, $+1) = elt+2
end
disp(y)
```

2ème technique (à oublier rapidement)

```
x = linspace(1, 3, 5)
// boucle sur les indices du vecteur x
n = length(x)
for i=1:n
    // ajout d'un nouvel élément dans le vecteur y
    y(1, i) = x(i)+2
end
disp(y)
```

3ème technique (à privilégier !!)
Avec Scilab, on peut directement obtenir le résultat attendu de la manière suivante :

```
--> y = x+2
y  =
3.   3.5   4.   4.5   5.
```

Autres exemples :

```
--> 10*x-2
 ans  =
8.   13.   18.   23.   28.

--> x/10
 ans  =
   0.1   0.15   0.2   0.25   0.3
```

Puissance .^ (avec un point devant)

```
--> x.^2
 ans  =
1.   2.25   4.   6.25   9.
```

Inverse (**avec un point devant**)

```
--> 1./x
 ans  =
   1.   0.6666667   0.5   0.4   0.3333333
```

2.2. Fonctions mathématiques

Avec Scilab, on peut directement appliquer une fonction mathématique à chacun des éléments d'un vecteur ou d'un tableau :

```
--> sqrt(x)    // racine carrée
 ans  =
1.   1.2247449   1.4142136   1.5811388   1.7320508
```

```
--> 1 + sin(x/3)
 ans  =
   1.3271947   1.4794255   1.6183698   1.7401769   1.841471
```

Exercice 04

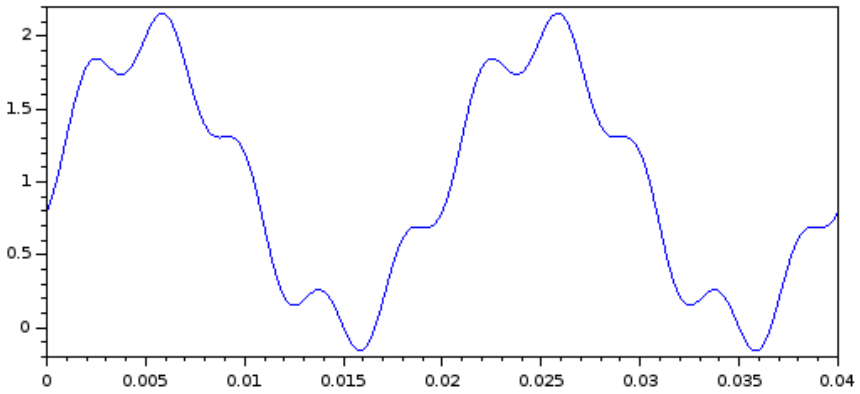
On souhaite tracer la courbe y=f(t) de la fonction :

$$y = f(t) = 1 + \sin(100.\pi.t) - \cos(500.\pi.t)/5$$

Compléter le code suivant :

```
t = linspace(0, 0.04, 1000)
y =
plot(t, y)
```

Résultat attendu :



2.3. Opération entre tableaux et vecteurs

On s'intéresse à des tableaux ou des vecteurs qui ont exactement les mêmes dimensions (même nombre de lignes et de colonnes) :

```
-->a = [1  2  4]
 a  =
   1.   2.   4.

-->b = [5  0  3]
 b  =
   5.   0.   3.
```

Opération élément par élément :

```
-->a+b
 ans  =
6.   2.   7.

-->a-b
 ans  =
-4.   2.   1.
```

```
-->a.*b     // avec un point devant
 ans  =
5.   0.   12.

-->b./a     // avec un point devant
 ans  =
5.   0.   0.75
```

Exercice 05

On veut faire la somme :

$$\sum_{i=1}^{20} i.(i+1)^3 = 1\times 2^3+2\times 3^3+3\times 4^3+4\times 5^3+...+20\times 21^3$$
$$= 1\times 8+2\times 27+3\times 64+...$$
$$= 8+54+192+...$$

On peut y arriver sans faire de boucle, en manipulant directement des vecteurs.

Compléter le code suivant :

```
i = ________   // vecteur [1 2 ... 20]
c = ________   // vecteur [8 27 64 ...]
p = ________   // vecteur [8 54 192 ...]
s = sum(p)     // fonction somme native de Scilab
disp(s)
```

Résultat attendu :

```
 863786.
```

Exercice 06

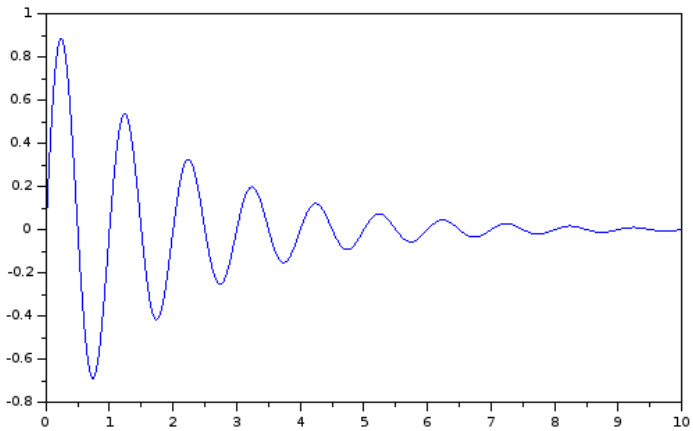
On souhaite tracer la courbe y=f(x) de la fonction :

$$y = f(x) = e^{-x/2}\sin(2\pi x)$$

Compléter le code suivant :

```
x = linspace(0, 10, 1000)
y = ____________________________
plot(x, y)
```

Résultat attendu :



2.4. Remarques diverses

- Agrégation de vecteurs

```
--> a = [1 2 4]
--> b = [5 0 3]

--> c = [a; b]   // tableau constitué des 2 vecteurs
 c  =
   1.   2.   4.
   5.   0.   3.
```

- Transposition de lignes et colonnes

```
--> a'
 ans  =
   1.
   2.
   4.

--> c'
 ans  =
   1.   5.
   2.   0.
   4.   3.
```