TP2 : Vecteurs, opérations et graphes simples

1. Définition de vecteurs élément par élément

Dans *Scilab*, les vecteurs lignes ou colonnes sont notés entre crochets. Les coefficients sont séparés par une virgule ou un espace dans le cas d'un vecteur ligne et par un point-virgule dans le cas d'un vecteur colonne. Nous verrons qu'il y a d'autres méthodes pour définir un vecteur.

Exemples :

```
-->U=[8, sqrt(25), -2/5]
```

cette instruction crée le vecteur ligne : $(8 \quad 5 \quad -0.4)$

Par contre l'instruction :

```
-->V=[8; sqrt(25); -2/5]
```

va crée le vecteur colonne : $\begin{pmatrix} 8 \\ 5 \\ -0.4 \end{pmatrix}$

Remarques :
Le vecteur le plus simple que l'on puisse créer est le vecteur vide :

```
-->w=[]
```

On peut transposer un vecteur en ajoutant une apostrophe après celui-ci. Ainsi :

```
-->z=[1;2;3]'
```

désignera le vecteur ligne : (1 2 3)

Exercice 1

(a) Créer le vecteur $\begin{pmatrix}1\\2\\3\end{pmatrix}$ de deux manières.

(b) Affecter respectivement à u et v les vecteurs lignes suivants : $(-1\ 3\ 5)$ et $(6\ 2\ 0)$. Puis affecter à une variable w le vecteur colonne ayant les mêmes coefficients que $3u - 2v$.

(c) Que renvoie l'instruction `-->length(w)` ?

2. Définition globale de vecteurs

(a) Il sera souvent très utile de créer un vecteur ligne dont les coefficients forment une subdivision d'un intervalle :

Si `a`, `b` et `h` sont des nombres réels alors `a:h:b` désigne le vecteur formé par tous les nombres `a`, `a+h`, `a+2h`,...compris entre `a` et `b`. (On dit que `h` est le *pas* de la subdivision)

Exemples :

```
-->-2:0.8:3
ans  =
   - 2.   - 1.2   - 0.4   0.4   1.2   2.   2.8
-->v=9:-0.5:6
v =
   9.   8.5   8.   7.5   7.   6.5   6.
```

Si on omet le paramètre `h` avec `a<b` alors `h=1` par défaut :

Exemples :

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

(b) **Fonction `linspace()`**
Dans le cas où l'on souhaite créer une subdivision régulière à partir d'un intervalle $[a, b]$ et d'un nombre de coefficients n prédéterminé, on utilisera la fonction *linspace*. `linspace(a,b,n)` renvoie un vecteur ligne composé de `n` coefficients, le premier étant le nombre `a` et la dernier `b`.

Exemple :

```
-->x=linspace(1,6,5)
x =
   1.    2.25   3.5   4.75   6
```

Remarque : l'instruction `linspace(a,b,n)` est équivalente à l'instruction `a:(b-a)/(n-1):b`. En effet dans la première instruction le pas vaut `(b-a)/(n-1)`.

(c) **Fonction `zeros()`**
`zeros(1,n)` créer un vecteur ligne ayant n coefficients égaux à 0.
`zeros(n,1)` créer un vecteur colonne ayant n coefficients égaux à 0.

Exemple :

```
-->x=zeros(1,5)
x =
   0.   0.   0.   0.   0.
```

(d) **Fonction `ones()`**
Son emploi est identique à celui de la fonction `zeros()` mais la fonction `ones()` permet d'initialiser le vecteur avec des coefficients tous égaux à 1.

Exemple :

```
-->x=ones(4,1)
x =
   1.
   1.
   1.
   1.
```

(e) **Fonction `rand()`**
Cette fonction permet de générer des nombres pseudo-aléatoires. Par défaut `rand()` suit une loi uniforme sur $[0,1]$ mais la commande `rand("normal")` permet par la suite de forcer `rand()` à suivre une loi normal centrée réduite. Si l'on souhaite à nouveau simuler une loi uniforme sur $[0,1]$ on utilisera la commande `rand("uniform")`.

Exemples (à tester et comprendre - prendre des notes) :

```
-->x=rand()
-->u=rand(1,3)
-->v=rand(3,1)
-->z=rand(1,5)+1
```

Remarque :
Nous verrons plus tard une autre fonction pour générer des nombres pseudo-aléatoires, la fonction `grand()` (get random number). Cette dernière offre plus de possibilités, elle permet notamment de simuler des variables aléatoires selon de nombreuses lois.

3. Manipulations de vecteurs

(a) **Accéder aux coefficients**
Pour prendre connaissance du $i^{ème}$ coefficients ($i \in \mathbb{N}^*$) d'un vecteur `u`, on écrira `u(i)`. La toute première valeur d'un vecteur `u` est donc `u(1)`. Pour *Scilab*, et contrairement à d'autres langages informatiques (*C*, *C++*, *Php*,...), `u(0)` n'a pas de sens.

Exemple :

```
-->u=-2:8
u =
   -2.  -1.  0.  1.  2.  3.  4.  5.  6.  7.  8.
-->u(3)
ans =
   0.
```

(b) **Modifier un coefficient**
De cette manière, on peut aussi modifier un coefficient :

```
-->u(3)=10
```

Remplacera le `0` du troisième coefficient par `10`

(c) **Extraire un vecteur**
Il est aussi possible d'extraire une partie d'un vecteur en indiquant, entre parenthèses, les indices de début et de fin séparés par deux points et éventuellement un pas :

Exemples (avec le vecteur `u` *précédent) :*

```
-->u(4:7)
ans =
   1.   2.   3.   4.
```

et la commande suivante renverra les coefficients de rang pair :

```
-->u(2:2:10)
ans =
  -1.   1.   3.   5.   7.
```

Toujours avec le même vecteur `u`, tester puis comprendre l'instruction suivante : `-->u(2:0.5:11)`

(d) **Augmenter la taille d'un vecteur**
On peut augmenter la taille d'un vecteur sans avoir à définir les coefficients intermédiaires :

```
-->x=1:3;
-->x(6)=8
x =
   1.   2.   3.   0.   0.   8.
```

la dernière instruction a pour effet d'augmenter la taille de `x` jusqu'à 6 en affectant le nombre 8 au sixième coefficient. Les coefficients déjà affectés ne sont pas modifiés et ceux qui sont intermédiaires sont initialisés à 0.

Exercice 2 :

(a) Créer, de deux manières différentes, un vecteur ligne `u` dont les coefficients vont de 1 à 10 avec un pas de 1.

(b) Modifier en une seule commande le vecteur `u` de telle sorte que les coefficients 3 à 7 soient nuls.

(c) Modifier en une seule commande le vecteur `u` de telle sorte que toutes les coefficients de rang un multiple de 3, soient égaux à -1.

(d) Créer un vecteur colonne `v` dont les coefficients vont de 4 à -3 en décroissant avec un pas de 0,5.

(e) On souhaite définir un vecteur colonne `x`, subdivision de l'intervalle $[0, 100]$ avec un pas de $0,0001$. Quelle commande vous semble la meilleure ? Pourquoi ?

```
-->u=0:0.0001:100
-->u=linspace(0,100,1000001)'
-->u=0:0.0001:100;
-->u=(0:0.0001:100)';
-->u=[0:0.0001:100]'
```

4. Quelques opérations supplémentaires sur les vecteurs

(a) Concaténation de vecteurs

Considérons deux vecteurs `u` et `v`.
Si `u` et `v` sont des vecteurs lignes alors l'instruction `[u,v]` les concatène en un nouveau vecteur ligne.
Si `u` et `v` sont des vecteurs colonnes alors l'instruction `[u;v]` les concatène en un nouveau vecteur colonne.

Exemple :

```
-->u=zeros(1,4);
-->v=1:3;
-->w=[u,v]
w=
   0.   0.   0.   0.   1.   2.   3.
```

(b) **Fonction `sum()`**
Cette fonction est très importante et sera très souvent utilisée. Elle permet d'additionner tous les coefficients d'un vecteur.

Exemples (à tester et comprendre) :

```
-->x=1:100;
-->sum(x)
ans =
   5050.
-->u=[1, 3+2*%i, -8] ;
-->sum(u)
ans =
   -4. + 2.i
```

(c) **Opérations coefficient par coefficient**

`+`	:	addtitionne coefficient par coefficient
`-`	:	soustrait coefficient par coefficient
`.*`	:	multiplie coefficient par coefficient (ne pas oublier le point !)
`./`	:	divise coefficient par coefficient (ne pas oublier le point !)
`.^`n	:	élève chaque coefficient à la puissance n
`f(u)`	:	applique la fonction `f` à chaque coefficient du vecteur `u`. `f` peut, par exemple, être la fonction `floor, sin, sqrt, abs, log,...`

Exemples (à tester et comprendre) :

```
-->x=-%pi:0.1:%pi
-->y=sin(x)
-->u=1:10
-->u=1 ./u   //Ne pas oublier l'espace après le 1
-->u=u.^2
```

5. Quelques exercices

Exercice 3 : norme et produit scalaire dans un espace supposé muni d'un repère orthonormal.

(a) Écrire un programme (voir TP 1) qui demande à l'utilisateur un vecteur u et qui affiche ensuite la norme de ce vecteur u.

(b) Écrire un programme qui demande à l'utilisateur un vecteur u puis un vecteur v et qui affiche ensuite le produit scalaire de u et v.

Exercice 4 : $\zeta(2)$, loi des grands nombres et constante d'Euler.

(a) Écrire un programme qui demande à l'utilisateur un nombre entier $n \geqslant 1$ et qui affiche ensuite la valeur de la somme :

$$\sum_{k=1}^{n} \frac{1}{k^2}, \quad \text{comparer cette somme à } \frac{\pi^2}{6} \text{ lorsque } n \text{ est grand.}$$

(b) i. Créer un programme qui demande à l'utilisateur un nombre entier n et qui affiche ensuite un vecteur de taille n dont les composantes sont les résultats de n lancers de dés équilibrés à 6 faces.

ii. Modifier ce programme pour qu'il affiche ensuite la moyenne de ces lancers de dés.

(c) Écrire un programme qui demande à l'utilisateur un nombre entier n et qui affiche ensuite la valeur de :

$$\left(\sum_{k=1}^{n} \frac{1}{k}\right) - \ln(n)$$

Que constatez-vous lorsque n devient grand ?

Exercice 5 : graphiques (help plot, help plot2d).

(a) On souhaite tracer le graphe de la fonction $x \mapsto \sin(x)$ sur l'intervalle $[-\pi, \pi]$. Pour cela on :

i. commence par se donner une liste d'antécédents :
```
-->x=-%pi:0.1:%pi;
```
ii. crée les images de ces antécédents :
```
-->y=sin(x);
```
iii. affiche enfin le graphe avec la commande :
```
-->plot(x,y)
```

(b) Créer un programme qui demande à l'utilisateur un nombre entier $n > 1$ et qui affiche le graphe de $x \mapsto \left(1 + \frac{1}{x}\right)^x$ sur $[1, n]$ avec un pas de 1. Que remarquez-vous (comparer avec la constante e) ?