Chapitre 4 : La boucle FOR et ses applications classiques

I Syntaxe de la boucle FOR avec Scilab

L'instruction `x=a:h:b` : elle fournit sous forme de vecteur ligne la progression arithmétique de premier terme a et de raison h, avec comme dernier terme le plus grand élément de cette progression inférieur ou égal à b (si h > 0).
Par défaut, h est égal à 1.

Exemple :

```
-->x=0:3
 x  =
 0.    1.    2.    3.
```

;

```
-->x=3.5:-1:0
 x  =
 3.5    2.5    1.5    0.5
```

Soit v un vecteur ligne. L'instruction `for k=v , instruction(s) ; end` permet de répéter la ou les instructions pour k prenant successivement comme valeurs les composantes de v.

Exemple :

```
-->for k=1:4 , disp(k^2) ;end
    1.
    4.
    9.
   16.
```

Exercices :

1. Afficher les entiers pairs de 1 à 100
2. Afficher les multiples 3 entre -321 et 48.
3. Afficher les carrés des 50 premiers entiers.
4. Effectuer successivement 30 fois de suite l'opération : mettre au carré un nombre, puis lui enlever la moitié du nombre de départ. On partira du nombre 2.

II Applications classiques à maîtriser

Méthode 1 : calcul des termes successifs d'une suite récurrente d'ordre 1

Soit f une fonction numérique et (u_n) la suite définie par son premier terme $u_0=a$ et la relation de récurrence $u_{n+1}=f(u_n)$.

Le calcul et l'affichage du (n+1)-iéme terme de la suite (d'indice n) se fait par cette séquence d'instructions :

```
u=a
for k=1:n ,  u=f(u) ; end
disp(u)
```

`De plus, le placement du disp à l'intérieur de la boucle permet d'afficher chacun des termes calculé.`

Voici un tableau qui décrit l'évolution des variables utilisées après chaque instruction. Complétez-le pour bien comprendre l'algorithme :

Variable k	Variable u
	$a=u_0$
1	$f(u)=f(u_0)=u_1$
2	
3	

…	…
n	

Ex 1 : Soit (u_n) la suite définie par $u_0=2$ et $u_{n+1}=u_n e^{-u_n}$ pour tout $n\geq 0$. Écrire des instructions permettant le calcul et l'affichage de u_{25} .

Ex 2 : Soit (u_n) la suite définie par $u_1=2$ et $u_{n+1}=\ln(1+u_n)$ pour tout $n\geq 1$. Écrire des instructions permettant le calcul et l'affichage de u_{36} .

Méthode 2 : calcul des termes successifs d'une suite récurrente d'ordre 2

Prenons un exemple : soit (u_n) la suite définie par ses deux premiers termes $u_0=3$, $u_1=1$ et pour tout $n\geq 0$, $u_{n+2}=u_{n+1}-2\mathrm{u}_n$. Essayons d'écrire un programme qui calcule et affiche u_{10} .

D'abord calculons à la main quelques termes de cette suite :

u_0	u_1	u_2	u_3	u_4

Proposer un script (déjà bien amorcé ci-dessous) et compléter le tableau pour décrire son fonctionnement.

```
x=3, y=1
for k=2 : 10, z = y-2*x ;     x=...     ;     y=...     ; end
disp(y)
```

Variable k	Variable z	Variable x	Variable y
		$3=u_0$	$1=u_1$
2			
3			
4			

Remarque : le script précédent peut se terminer par l'instruction disp(y) ou disp(z) (les deux donnent bien u_{10}), mais l'instruction finale disp(x) ne marcherait pas. Pourquoi ?

Méthode 3 : Calcul de sommes

Soit f une fonction numérique et n un entier strictement positif. On considère la somme $S_n=\sum_{k=1}^{n} f(k)$.

Pour calculer S_n , on se ramène à une situation de suite récurrente. On a en effet la relation suivante, valable pour tout $n \geq 2$: $S_n = S_{n-1} + f(n)$.

Si on pose $S_0 = 0$, la relation est vraie à partir de n=1.
On en déduit l'algorithme de calcul suivant :

```
s=0
for k=1:n, s=s+f(k);end
disp(s)
```

Ex 3 : -->s=0 ; for k=1:10, s=s+1/(2^k) ; end ; disp(s)
0.9990234
Quelle somme Scilab a-t-il calculé ? Pourquoi est-ce si proche de 1 ?

Méthode 4 : Calcul de produits

Soit f une fonction numérique et n un entier strictement positif. On cherche à calculer le produit $P_n=\prod_{k=1}^{n} f(k)$. Est-ce vraiment nouveau par rapport à la situation précédente ? Pas vraiment ...

Voici un algorithme de calcul :

```
p=1
for k=1:n, p=p*f(k);end
disp(p)
```

Vous aurez remarqué l'initialisation « neutre » pour un produit : `p=1`

Ex 4 : `-->p=1;for k=1:10, p=p*((k+1)/k);end;disp(p)`
11.
Quel produit Scilab a-t-il calculé ? Comprendre le résultat égal à 11.

TP 4 : Les scripts demandés devront être sauvegardés dans un fichier TP4 avec le numéro de l'exercice dans leur nom.

Ex 1 :Donner les valeurs dans les variables u et s après les instructions suivantes :
u=3 ; s=3 ; for k=1:3 , u=u+k ; s=s+u ; end

Ex 2 : Soit (u_n) la suite définie par $u_1=0.5$ et $u_{n+1}=u_n^2-u_n$ pour tout $n\geq 1$. Ecrire un programme qui calcule et affiche le terme u_n de la suite, l'entier n étant entré par l'utilisateur. Exécuter le programme pour $n=10, n=11, n=100, n=101, n=10000, n=10001$. Que peut-on conjecturer ?

Ex 3 : Soit (u_n) la suite définie par $u_0=a>0$ et $u_{n+1}=\sqrt{(n+1)u_n}$ pour tout $n\geq 0$.
Ecrire un programme qui demande à l'utilisateur la valeur du premier terme a, un entier n, puis qui calcule et affiche u_n .

Ex 4 : Soit (u_n) la suite (de Fibonacci) définie par ses deux premiers termes $u_0=0$, $u_1=1$ et pour tout $n\geq 0$, $u_{n+2}=u_{n+1}+u_n$. Ecrire une FONCTION (que l'on pourra appeler « fibo ») qui calcule u_n et l'insérer dans un programme qui affiche u_n ainsi que $\frac{u_{n+1}}{u_n}$, l'entier n étant entré par l'utilisateur. Tester ces calculs pour de « grandes » valeurs de n. Que peut-on conjecturer sur les limites de (u_n) et de $(\frac{u_{n+1}}{u_n})$ lorsque $n\rightarrow+\infty$?

Ex 5 : Deux suites imbriquées
Soit (a_n) et (b_n) les suites définies par $a_0=1$, $b_0=9$ et pour tout entier n :

$$a_{n+1}=\frac{a_n+b_n}{2} \quad ; \quad b_{n+1}=\sqrt{a_n b_n}$$

Ecrire un programme qui calcule et affiche a_n et b_n lorsque l'entier n est entré par l'utilisateur.

Ex 6 : Ecrire une FONCTION qui pour tout entier naturel n, renvoie n ! (c'est un produit !)

Ex 7 : Ecrire la somme S_n (contenu de la variable s) calculée par la séquence d'instructions suivante :

u=1 ; s=0; for k=1 : n, u=2*u ; s=s+u ;end

Ex 8 : Ecrire en fonction de n le contenu de la variable x après les instructions suivantes :

x=2 ; for k=1 : n, x=x*x ;end

Ex 9 : Programmer et tester le calcul de $S_n=\frac{6}{\pi^2}\sum_{k=1}^{n}\frac{1}{k^2}$ pour de « grandes » valeurs de n. Que peut-on conjecturer ?

Ex 10 : Faire calculer $S=\sum_{k=0}^{6}u_k$ avec $u_0=2$ et $u_{n+1}=\sqrt{u_n}$ pour tout $n\geq 0$.

Ex 11 : Ecrire un programme qui calcule $P_n=\prod_{k=1}^{n}\left(1-\frac{k}{n+1}\right)$ pour un entier $n\geq 1$ saisi par l'utilisateur. Donner les valeurs de P_5, P_{10}, P_{100} .

Exprimer P_n en fonction de n (sans le symbole $\prod$). Quelle limite pour la suite (P_n) peut-on conjecturer ?

Ex 12 : Que fait le programme suivant ?

```
n=input('donner un entier n')
m=0 ; e=0 ;
for i=1: n , u=input('donner un nombre') ; m=m+u ; e=e+u^2 ; end
m=m/n
e=sqrt(e/n-m^2)
disp(m,'m=') ; disp(e,'e=')
```

Ex 13 : Sommes doubles

Ecrire un script qui saisit un entier $n\geq 2$ puis qui calcule et affiche les résultats des sommes doubles suivantes :

$$\sum_{1\leq i<j\leq n}\frac{1}{i+j} \quad \text{et} \quad \sum_{1\leq i\leq j\leq n}\frac{1}{i+j}$$

Ex 14 : Somme de Riemann, calcul approché d'une intégrale par la méthode des rectangles.

Le but de l'exercice est de calculer une valeur approchée du nombre a suivant :

$$a=\left(\int_{-5}^{5}\exp(-x^2)\,dx\right)^2 .$$

Dans un même script :

1) Créer une fonction f correspondant à $f(x)=\exp(-x^2)$.
2) La faire tracer sur [-5,5].
3) On considère $S_n=\frac{10}{n}\sum_{k=1}^{n}f\left(-5+\frac{10}{n}k\right)$. Comprendre que cette quantité est une somme d'aire de n rectangles. De quelle intégrale S_n est-elle une valeur approchée ? En déduire des instructions permettant d'afficher une valeur approchée de a (l'utilisateur donnera le nombre de rectangles n). Tester avec différentes valeurs de n (de plus en plus grandes). Qu'observe-t-on ?

Ex 15 : On considère l'équation cos(x)=x sur [-2π ; 2π].

1. Montrer que cette équation admet une unique solution a.
2. On considère alors la suite $u_{n+1}=\cos(u_n)$ et $u_0=0,5$. On admet que cette suite converge. En utilisant un script scilab simulant $(u_n)_n$, donner une estimation de la valeur de a.