TP 2 – Boucles « for »

Introduction

Tout programme un peu évolué utilise l'une des trois « structures de base » suivantes :

- Les deux structures de boucles que sont les boucles « for » et les boucles « while »,
- La structure conditionnelle « If then » ou « If then else ».

Ainsi tout langage de programmation permet de définir une boucle « for », une boucle « while » ou un « if then else ». Dans ce TP, nous allons voir comment programmer des boucles « for » avec Scilab.

Tous les programmes de ce TP devront être enregistrés dans un répertoire nommé TP02, idéalement directement sur votre clef usb. Il faudra à la fin de ce TP sauvegarder l'ensemble de ce répertoire soit sur une clef usb, soit sur votre drive.

Vous ne terminerez probablement pas ce TP durant cette séance. Il faudra le terminer à la prochaine séance.

1 Un exemple pour bien comprendre

On considère la suite (u_n) définie par $u_0 = 3$ et $\forall n \in \mathbb{N},\ u_{n+1} = 2\,u_n - 1$.

Exercice 1

1. Calculez **de tête** u_5.
2. Recopiez le programme ci-dessous dans SciNotes et enregistrez-le sous le nom « Exercice_1 ». Exécutez-le et vérifiez qu'il affiche bien u_5.

Exercice_1.sce

```
1  u=3
2  u=2*u-1
3  u=2*u-1
4  u=2*u-1
5  u=2*u-1
6  u=2*u-1
7  disp(u)
8
```

3. A quoi sert la dernière ligne de ce programme ?

   ……

   Que se passe-t-il si on la supprime ?

   ……

4. Complétez la dernière ligne pour que le programme affiche le message « u5 vaut : » avant d'afficher la valeur de u_5.
5. Complétez ce programme pour qu'il affiche toutes les valeurs de u_1 à u_5, en affichant un message d'explication uniquement pour u_5.
6. Corrigez ce programme pour qu'il affiche toutes les valeurs jusqu'à u_7.

2 Le même exemple avec une boucle for

On a vu dans l'exercice 1 que pour calculer et afficher toutes les valeurs jusqu'à u_5, on répète 5 fois les mêmes instructions.
Il y a un moyen beaucoup plus simple pour faire ça, une **boucle « for »** :

```
1 u=3
2 for n=1:5
3     u=2*u-1
4     disp(u)
5 end
6 disp("u5 vaut :", u)
7
```

Examinons pas à pas ce que fait Scilab en exécutant ce programme :

Ligne 1 : Scilab crée une variable **u** et lui affecte la valeur 3. Si **u** était déjà créée, il l'efface et lui affecte la valeur 3.
Ligne 2 : Scilab détecte l'instruction **for**, il crée une variable **n** et lui affecte la valeur 1 (si **n** était déjà créée, il l'efface et lui affecte la valeur 1).
Ligne 3 et 4 : Scilab exécute ces deux lignes et revient à la ligne 2.

Ligne 2 : Scilab change la valeur de **n** et lui affecte la valeur 2.
Ligne 3 et 4 : Scilab exécute ces deux lignes et revient à la ligne 2.

Ligne 2 : Scilab change la valeur de **n** et lui affecte la valeur 3.
Ligne 3 et 4 : Scilab exécute ces deux lignes et revient à la ligne 2.

Ligne 2 : Scilab change la valeur de **n** et lui affecte la valeur 4.
Ligne 3 et 4 : Scilab exécute ces deux lignes et revient à la ligne 2.

Ligne 2 : Scilab change la valeur de **n** et lui affecte la valeur 5.
Ligne 3 et 4 : Scilab exécute ces deux lignes et revient à la ligne 2.

Arrivé à ce stade, Scilab a terminé la boucle **for** *car la variable* **n** *a pris sa dernière valeur. Il peut donc maintenant passer à la ligne suivante. On dit que Scilab sort de la boucle.*

Ligne 6 : Scilab affiche dans la console ce qu'on lui demande d'afficher.

La variable **n** est appelé le ***compteur*** de la boule for. Elle est automatiquement créée par Scilab. On voit qu'elle augmente de 1 à chaque passage dans la boucle. On dit qu'elle est ***incrémentée*** de 1.
On peut choisir n'importe quelle variable comme compteur, mais il ne faut pas que cette variable soit utilisée ailleurs pour une autre partie du programme.

Par exemple, le programme suivant ne va pas avoir du tout le même résultat :

```
1 u=3
2 for u=1:5
3     u=2*u-1
4     disp(u)
5 end
6 disp("u5 vaut :", u)
```

En effet, dans le programme ci-dessus, on a utilisé la variable **u** comme compteur ! Ainsi, à chaque passage à la ligne 2, Scilab efface la valeur de **u** et lui affecte la valeur 1 puis 2, puis 3, etc. Ce programme « tourne », dans le sens où Scilab ne renvoie pas d'erreur et l'exécute sans problème, mais il ne fait pas du tout ce que l'on voulait faire.

Par contre, le programme suivant (ou on a remplacé **n** par **k**) fonctionne tout aussi bien que le premier :

```
u=3
for k=1:5
    u=2*u-1
    disp(u)
end
disp("u5 vaut :", u)
```

Exercice 2

1. Recopiez le programme ci-dessus dans SciNotes et enregistrez-le sous le nom « Exercice_2 ». Exécutez-le et vérifiez qu'il affiche bien u_5.
2. Modifiez le programme ci-dessus pour qu'il affiche u_{20}. Vous devez obtenir $u_{20} = 2097153$.
3. Donner ci-dessous la valeur approchée de u_{500}

$$\boldsymbol{u_{500}} \approx$$

4. Peut-on calculer u_{5000} ? Pourquoi ?

.....

.....

Exercice 3

On veut pouvoir modifier facilement la valeur n jusqu'à laquelle on va calculer u_n. On modifie alors le programme comme ci-dessous :

Exercice_3.sce

```
n=
u=3
for k=1:n
    u=2*u-1
    disp(u)
end
disp("Lorsque n = ", n)
disp("u_n = ", u)
```

1. Recopiez le programme ci-dessus dans SciNotes et enregistrez-le sous le nom « Exercice_3 ». Complétez-le pour qu'il affiche u_{500} et exécutez-le. Vérifiez qu'il fonctionne bien.
2. Modifiez la valeur de n et vérifiez que le programme calcule bien à chaque fois u_n.
3. Modifiez votre programme pour qu'il n'affiche que la valeur de u_n et pas toutes les valeurs intermédiaires.

On a créé un programme qui calcule u_n pour tout valeur de n, à condition que les calculs ne dépassent pas les limites de Scilab. On peut le modifier pour que cette valeur soit entrée par l'utilisateur à chaque exécution du programme.

Exercice 4

Enregistrez le programme de l'exercice 3 sous le nom « Exercice_4 » puis modifiez-le pour qu'il demande la valeur de **n** à l'utilisateur à chaque exécution du programme. Vérifiez que votre programme fonctionne bien.

3 Calculs de suites récurrentes avec des boucles for.

Exercice 5

On note (v_n) la suie définie par : $v_0 = 9$ et $\forall n \in \mathbb{N},\ v_{n+1} = 2\sqrt{v_n} - 1$.

1. Créez un programme Scilab, que vous enregistrerez sous le nom « Exercice_5 » qui demande à l'utilisateur la valeur de n et calcule v_n.
2. A l'aide de votre programme, donnez la valeur approchée de v_{50}, v_{500} et v_{5000}.
   (vous devez trouver $v_{50} \approx 1{,}086$)

   $v_{50} \approx$ $v_{500} \approx$ $v_{5000} \approx$

3. Conjecturez le comportement asymptotique de (v_n).

   ……

   ……

Exercice 6

On considère la suite (w_n) définie par :

$$w_1 = 0{,}5 \text{ et } \forall n \in \mathbb{N},\ w_{n+1} = \frac{4w_n+8}{2w_n-3}$$

A l'aide d'un programme Scilab, vérifiez que $w_{50} \approx 4.4089601$.

Exercice 7

On considère la suite (x_n) définie par :

$$x_1 = 1{,}5 \text{ et } \forall n \in \mathbb{N},\ x_{n+1} = \frac{nx_n+1}{2(n+1)}$$

1. A l'aide d'un programme Scilab, calculez et affichez toutes les valeurs de (x_n) pour n allant de 1 à 100.
   Vous devez trouver $x_{100} \approx 0{,}01$
2. Conjecturez le sens de variation et la limite de (x_n)

4 Calcul de sommes avec une boucle « for ».

Premier exemple.

On définit, la suite (H_n) par :

$$\forall n \in \mathbb{N}^*, \qquad H_n = \sum_{k=1}^{n} \frac{1}{k}.$$

Cette suite a un nom, elle s'appelle la « série harmonique ».

On veut construire un programme qui demande n à l'utilisateur puis affiche la valeur de H_n.

Pour bien comprendre comment on va calculer H_n à l'aide d'une boucle « for », voyons comment on peut calculer H_5 sans boucle :

Harmo5.sce

```
1  H=0
2  H=H+1/1
3  H=H+1/2
4  H=H+1/3
5  H=H+1/4
6  H=H+1/5
7  disp(H)
8
```

Ce petit programme affiche la valeur de H_5. On voit qu'on répète l'instruction « H=H+1/k » pour k allant de 1 à 5. C'est justement quelque chose que l'on peut faire très simplement avec une boucle « for » :

Harmo5AvecFor.sce

```
1  H=0
2  for k=1:5
3      H=H+1/k
4  end
5  disp(H)
```

NB : dans ce programme, on a utilisé les variables H et k pour plus de clarté, mais les programmes suivants fonctionnent tout aussi bien :

Harmo5AvecFor.sce

```
1  bob=0
2  for k=1:5
3      bob=bob+1/k
4  end
5  disp(bob)
6
```

Harmo5AvecFor.sce

```
1  bob=0
2  for johnny=1:5
3      bob=bob+1/johnny
4  end
5  disp(bob)
6
```

Simplement, il est beaucoup plus difficile de comprendre ces deux programmes que le premier.

Exercice 8

En vous aidant du programme « Harmo5AvecFor » ci-dessus, écrire un programme « Harmonique » qui demande une valeur de n à l'utilisateur et renvoie la valeur de H_n, avec les messages apropriés.

Indication : vous devez trouver $H_{100} \approx 5{,}1873775$.

Exercice 9

Ecrire un programme « Exercice_9 » qui demande une valeur de $n \geq 10$ à l'utilisateur et renvoie la valeur de la somme S_n définie par :

$$\forall n \geq 10, \qquad S_n = \sum_{k=10}^{n} k^2.$$

Indication : vous devez trouver $S_{100} = 338065$

Exercice 10

Ecrire un programme « Exercice_10 » qui demande une valeur de n à l'utilisateur et renvoie la valeur de la somme T définie par :

$$\forall n \geq 0, \qquad T_n = \sum_{k=n}^{2n} (k^2 - 3k + 5).$$

Indication : vous devez trouver $T_{100} = 2313405$

5 Boucles « for » par pas de 2, 3 ou plus.

Par défaut, quand on écrit « `for k=1:10` », Scilab va faire passer la variable k de 1 à 10 par pas de 1 (en incrémentant k de 1 à chaque passage dans la boucle).

On peut très facilement lui demander d'avancer par pas de 2, 3 ou plus avec la syntaxe suivante :

Pour avancer par pas de 2 : `for k=1:2:10`
Pour avancer par pas de 3 : `for k=1:3:10`
Pour avancer par pas de 4 : `for k=1:4:10`

Bien sûr, quand on avance par pas de 2 et qu'on part de 1 on n'arrive pas à 10, on suit la progression suivante :

$$1 - 3 - 5 - 7 - 9.$$

Donc, avec l'instruction `for k=1:2:10`, Scilab sortira de la boucle dès que le compteur dépassera 10.

Exercice 11

Pour bien comprendre ce point, recopiez et exécutez le petit programme suivant et observez les valeurs affichées par Scilab dans la console :

PasDeDeux.sce

```
disp("par pas de deux")
for k=1:2:10
    disp(k)
end

disp("par pas de trois")
for k=1:3:10
    disp(k)
end

disp("par pas de quatre")
for k=1:4:10
    disp(k)
end

disp("par pas de cinq")
for k=1:5:10
    disp(k)
end
```

Exercice 12

A l'aide d'une boucle for, écrivez un programme qui demande la valeur de n à l'utilisateur et calcule la valeur de :

$$S_n = \sum_{\substack{k=1 \\ k \text{ pair}}}^{n} k^2$$

Et de :

$$T_n = \sum_{\substack{k=1 \\ k \text{ impair}}}^{n} k^2$$

6 Boucles « for » imbriquées.

Pour terminer avec les boucles for, voici un exercice qui vous montre qu'on peut imbriquer les boucles for : écrire une boucle for dans une boucle for.

Exercice 13

On considère la suite (S_n) définie par :

$$\forall n \geq 1, \qquad S_n = \sum_{k=1}^{n} \frac{1}{k+n}$$

1. Calculez à la main S_1, S_2 et S_3.
2. Recopiez et complétez le programme ci-dessous qui demande un nombre N à l'utilisateur et affiche toutes les valeurs de la suite (S_n).

Exercice_13.sce

```
1 N=input("Entrez une valeur N>=1 : ")
2 for n=1:N
3     S=0
4     for k=_____
5         S=_____
6     end
7     disp(S)
8 end
9
```

3. Exécutez votre programme avec $N = 10$. Vous devez obtenir le résultat suivant :

```
Entrez une valeur N>=1 : 10

  0.5

  0.5833333

  0.6166667

  0.6345238

  0.6456349

  0.6532107

  0.6587052

  0.6628719

  0.6661398

  0.6687714

-->
```

4. Modifiez votre programme pour qu'il affiche, pour n allant de 1 à N, la valeur de $|S_n - \ln 2|$. Puis exécutez votre programme avec $N = 5000$.
   a. Observez le ralentissement à la fin des calculs, pourquoi ce ralentissement ?
   b. Quelle conjecture peut-on faire quant à la limite suivante ?

$$\lim_{n \to +\infty} \sum_{k=1}^{n} \frac{1}{k+n}$$