
Informatique - T.P n°2

Suites numériques

On commencera par créer, dans son dossier personnel, un dossier intitulé "TP2" dans lequel on pourra enregistrer tous les programmes (au format `.sce`) ou fonctions (au format `.sci`) relatifs à cette nouvelle feuille d'exercices.

1 Calcul des termes d'une suite

Exercice 1.

(1) Deviner l'affichage du programme suivant avant de l'exécuter:

```
u=0
for i=1:6
    u=2*u+3
    disp(u)
end
```

- (2) Quelle est la suite définie dans ce programme?
- (3) Modifier le programme précédent pour qu'il affiche uniquement u_{100} .
- (4) Ecrire un programme sans boucle `for` qui affiche le terme u_n .

Exercice 2.

- (1) Dans chacun des cas suivants, écrire un programme qui demande à l'utilisateur un entier n et affiche le terme u_n :
 - (a) $u_0 = 0.1$ et, pour tout $n \geq 0$, $u_{n+1} = e^{-2u_n}$.
 - (b) $u_1 = 1$ et, pour tout $n \geq 1$, $u_{n+1} = 2nu_n + 3$.
- (2) Modifier les programmes précédents pour en faire des fonctions.

Exercice 3.

Soit (u_n) la suite définie par

$$u_1 = 0, \quad u_2 = -9, \quad \text{et} \quad u_{n+2} = 6u_{n+1} - 9u_n.$$

- (1) Écrire un programme qui demande n à l'utilisateur et affiche la valeur de u_n .
- (2) Compléter le programme précédent afin qu'il renvoie également la valeur de $S_n = \sum_{k=1}^n u_k$.

Exercice 4. (D'après EML 2017) On considère la suite (u_n) définie par

$$u_0 = 2, \quad \text{et} \quad u_{n+1} = \exp(u_n) - e \ln(u_n).$$

- (1) Montrer que (u_n) est bien définie et que, pour tout $n \geq 0$, $u_n \geq 2$.
- (2) En étudiant les variations puis le signe de la fonction $x \mapsto e^x - e \ln(x) - x$ sur $[2; +\infty[$, montrer que (u_n) est croissante puis montrer qu'elle diverge vers $+\infty$.
- (3) Écrire un programme qui, étant donné un réel $A \geq 0$ renvoie un entier naturel N tel que $u_N \geq A$.

Exercice 5. Soit (u_n) la suite définie par

$$\begin{cases} u_0 & \in \mathbb{R} \\ u_{n+1} & = u_n - u_n^2 \end{cases}$$

- (1) Que peut-on dire du sens de variation de (u_n) ?
- (2) Écrire un programme qui demande à l'utilisateur un réel u_0 et un entier n et affiche le terme u_n .
- (3) On suppose que $u_0 = \frac{3}{4}$.
 - (a) Calculer u_n pour des valeurs de plus en plus grandes de n . Que semble-t-il se produire?
 - (b) Écrire un programme qui détermine la plus petite valeur de n telle que $u_n < 10^{-3}$.
- (4) On suppose que $u_0 = 2$.
 - (a) Calculer u_n pour des valeurs de plus en plus grandes de n . Que semble-t-il se produire?
 - (b) Écrire un programme qui détermine la plus petite valeur de n telle que $u_n < -10^8$.

Exercice 6. Soit (u_n) la suite définie par

$$\begin{cases} u_0 & = 4 \\ u_{n+1} & = u_n^2 + 1 \end{cases}$$

- (1) Écrire un programme qui demande à l'utilisateur un réel x et renvoie la valeur du premier terme de la suite qui soit supérieur ou égal à x .
- (2) Modifier le programme pour qu'il renvoie également l'indice de ce terme.
- (3) Réécrire ce programme pour en faire une fonction (attention, il y aura deux valeurs de sortie!).

Exercice 7.

- (1) Écrire une fonction qui, prenant pour argument l'entier n , permet de calculer $S_n = \sum_{k=1}^n \frac{1}{k}$.
- (2) Utiliser cette fonction pour conjecturer les valeurs de

$$\lim_{n \rightarrow +\infty} S_n, \quad \lim_{n \rightarrow +\infty} \frac{S_n}{n}, \quad \lim_{n \rightarrow +\infty} \frac{S_n}{\ln(n)}.$$

- (3) Écrire une fonction qui à deux entiers n et p associe

$$\sum_{k=1}^n k^p.$$

- (4) Écrire un programme qui compare les valeurs obtenues avec la fonction précédentes et les formules classiques du cours.

Exercice 8.

On veut déterminer la limite α de la suite (u_n) définie par

$$\begin{cases} u_0 & = 0 \\ u_{n+1} & = \sqrt{a + u_n}, \end{cases}$$

où $a > 0$ est un paramètre fixé.

- (1) On fait l'hypothèse suivante: si $|u_{n+1} - u_n| < 0.0005$, alors u_n est une approximation de la limite de la suite à 0.001 près. Écrire un programme qui utilise cette hypothèse pour donner une valeur approchée de la limite de (u_n) à partir du paramètre a entré par l'utilisateur.
- (2) On peut montrer, mathématiquement, que cette limite vaut

$$\alpha = \frac{1 + \sqrt{1 + 4a}}{2}.$$

Compléter le programme précédent pour vérifier que l'approximation précédente est bien vérifiée.

2 Suite en tant que vecteur ligne

Les exercices précédent avaient tous pour principe le calcul d'un terme de la suite. Ainsi, on calculait les termes individuellement. On peut également définir la suite comme "liste" de valeurs, et le terme de rang n sera donc le n -ième terme de cette liste (ou $(n + 1)$ -ième si la suite commence pour $n = 0$).

On représente une liste sous SciLab sous forme de **vecteur-ligne**. On définit un vecteur-ligne en indiquant la suite des nombres qui le composent, séparés par des virgules, le tout entre crochets [].

Si le vecteur-ligne u est défini par $u=[0,3,7,12,15]$, on accède au k -ième terme via $u(k)$. Dans le cas précédent, $u(3)$ renvoie donc 7.

L'instruction `length(u)` renvoie la longueur de u (c'est à dire le nombre de coefficients qui le composent, ici `length(u)` va donc renvoyer 5).

Si un vecteur-ligne possède n coefficient, on peut le compléter en affectant directement la valeur du p -ième coefficient (avec $p > n$). Si $p > n + 1$, les coefficients intermédiaires seront automatiquement affectés de la valeur 0.

Par exemple, si on écrit `u(9)=20`, SciLab affichera ensuite
`u=0. , 3. , 7. , 12. , 15. , 0. , 0. , 0. , 20.`

On peut donc facilement remplir, à l'aide d'une boucle `for` par exemple, toutes les composantes d'une suite (définie sous forme de vecteur-ligne).

Cela dit, cette commande est chère en temps de calcul. Il vaut mieux définir dès le départ la taille du vecteur-ligne, puis modifier ensuite chaque terme. La commande `zeros(1,n)` permet par exemple de créer un vecteur ligne de n termes, pré-rempli avec des zéros.

De plus, la fonction `find()` renvoie la position (c'est à dire le rang) des éléments du vecteur-ligne qui vérifient une condition donnée. Par exemple, `find(u>10)` renverra 4. , 5. , 9. qui sont les valeurs de k telles que $u_k > 10$.

Enfin, la fonction `sum()` prenant en argument un vecteur-ligne renvoie la somme de toutes les composantes de ce vecteur-ligne, ce qui est bien pratique. La même chose existe pour le produit, avec la commande `prod()`.

Exercice 9. Écrire une commande très courte, à l'aide de la fonction `prod()`, permettant de calculer $n!$.

Exercice 10. On considère la suite (u_n) définie par

$$\begin{cases} u_1 &= 5 \\ u_{n+1} &= u_n - 2, \end{cases}$$

- (1) Écrire une suite d'instruction qui permet d'obtenir les 20 premiers termes de la suite sous forme de vecteur-ligne.
- (2) En déduire la valeur de la somme des 20 premiers termes de (u_n) .
- (3) Reprendre les mêmes questions avec $u_1 = 16$.

Exercice 11. Ecrire, sur une seule ligne de commande, les intructions qui permettent de calculer la somme des 15 premiers termes de la suite géométrique de premier terme $v_1 = 3$ et de raison $\frac{4}{3}$.

Exercice 12. Reprendre l'Exercice 6 avec la notion de vecteur.

Si a, b, c sont trois réels avec $b > 0$ et $a \leq c$, la commande $\mathbf{a:b:c}$ permet de définir (en une seule instruction) le vecteur-ligne composé des termes d'une suite arithmétique de premier terme a , de raison b et dont le dernier terme sera le plus grand terme de la suite inférieur à c .

Par exemple, $1:1:20$ renvoie le vecteur-ligne composé des 20 premiers entiers consécutifs.

Exercice 13. (Sommes)

- (1) Écrire une fonction, qui ne comporte qu'une seule instruction, prenant pour argument n et renvoyant

$$\sum_{k=0}^n (3k + 4).$$

- (2) Écrire une fonction prenant pour argument n et renvoyant $\sum_{1 \leq i \leq j \leq n} ij$.

Exercice 14.

- (1) Définir un vecteur-ligne \mathbf{u} de longueur 40 composé des 40 premiers termes de la suite de Fibonacci, notée ici (u_n) , à partir de $u_1 = 1$.
- (2) Définir, à partir de \mathbf{u} , un vecteur-ligne \mathbf{v} représentant les 40 premiers termes de la suite (v_k) définie pour $k \geq 0$ par $v_k = u_{k+1}$.
- (3) Que fait la commande $\mathbf{a=v./u}$?
- (4) Calculer les 20 premiers termes de la suite (ϕ_n) définie pour $n \geq 1$ par

$$\phi_n = f\left(\frac{u_{n+1}}{u_n}\right), \quad \text{où } f(x) = x^2 - x - 1.$$

- (5) On admet que (ϕ_n) converge vers une limite ϕ . Au vu des dernières questions, que peut-on conjecturer quant à la valeur exacte de ϕ ?

3 Représentation graphique des termes

Il peut être intéressant de représenter graphiquement, avec SciLab, une suite réelle (u_n) . Pour cela, on utilise la fonction `plot2d()`. Plus précisément, `plot2d(X,Y,-1)` trace une croix aux points de coordonnées (x_k, y_k) où $\mathbf{X}=[x_1, \dots, x_n]$ et $\mathbf{Y}=[y_1, \dots, y_n]$ sont des vecteurs-ligne de même longueur. En remplaçant `-1` par `-2` ou `-3`, on change le style et la taille de la croix.

Exercice 15.

- (1) Écrire une suite d'instruction pour obtenir un vecteur-ligne dont les 25 composantes correspondent au 25 premiers termes de la suite (u_n) définie pour $n \geq 1$ par

$$u_n = \frac{\ln(\sqrt{n} + 1)}{\ln((n+1)^2 - 2)}.$$

- (2) Représenter graphiquement, avec l'aide de la fonction `plot2d()` les 25 premiers termes de la suite (u_n) . Que peut-on conjecturer? Le démontrer.

Exercice 16. (D'après **ECRICOME 2015**) On s'intéresse à la suite (u_n) définie par

$$u_0 = 1, \quad \text{et} \quad u_{n+1} = 1 - \exp(-u_n).$$

Compléter le programme suivant qui permet de représenter les 100 premiers termes de la suite puis, interpréter la figure ainsi obtenue par exécution du programme.

```

U=zeros(1,100)
U(1)=1;
for n=1:99
    U(n+1)=.....
end
plot(U, '+')

```

4 Suites récurrentes $u_{n+1} = f(u_n)$: schéma en escalier

On considère la suite (u_n) définie par

$$\begin{cases} u_0 &= 10 \\ u_{n+1} &= u_n - \ln(u_n) \end{cases} .$$

On introduit la fonction f définie sur $]0; +\infty[$ par

$$f(x) = x - \ln(x).$$

- (1) Écrire, sous SciLab, les instructions permettant de définir la fonction f (qu'on appellera également \mathbf{f}).
- (2) Représenter, sur un même graphique, la courbe de f (en bleu) et la droite d'équation $y = x$ (en vert) sur l'intervalle $[0.25; 10.25]$.
- (3) On veut représenter graphiquement le processus itératif permettant de générer les termes de la suite. On part d'un point A_0 de coordonnées (u_0, u_0) (c'est donc un point de la droite $y = x$) et d'un point B_1 de coordonnées $(u_0, f(u_0))$.

Si A_n et B_n sont déjà construits (pour un certain $n \in \mathbb{N}$), on construit B_{n+1} de coordonnées $(u_n, f(u_n))$ puis A_{n+1} de coordonnées (u_{n+1}, u_{n+1}) .

On veut créer la *ligne brisée* reliant les points $A_0 B_1 A_1 B_2 \dots A_n$.

- (a) Écrire un script permettant de créer la ligne brisée susmentionnée jusqu'au point A_n , où n est rentré par l'utilisateur.
 - (b) Représenter **sur un même graphique** les deux courbes précédentes (avec les mêmes couleurs), la ligne brisée (en rouge) jusqu'à $n = 9$ et, en plus, sous forme d'une croix \times les points A_0, A_1, \dots, A_9 .
 - (c) Interpréter le graphique obtenu.
- (4) On étudie alors de façon théorique la suite, de manière à justifier le résultat précédemment observé.
 - (a) Étudier les variations de f sur $]0; +\infty[$. Montrer que l'intervalle $[1; +\infty[$ est stable sous l'action de f . En déduire que tous les termes de la suite sont bien définis ainsi qu'une minoration des termes.
 - (b) Déterminer le signe de $f(x) - x$. En déduire les variations de la suite.
 - (c) Conclure.