ECE 1 - Année 2017-2018
Lycée français de Vienne
Mathématiques - F. Gaunard
http://frederic.gaunard.com

Informatique - T.P n°3

Suites, Listes et combinatoire

On commencera par créer, dans son dossier personnel, un dossier intitulé "TP3" dans lequel on pourra enregistrer tous les programmes (au format *.sce*) ou fonctions (au format *.sci*) relatifs à cette nouvelle feuille d'exercices.

1 Suite en tant que vecteur ligne

Jusqu'à présent, tous les exercices relatifs aux suites 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`.

☞ La commande `zeros(1,n)` permet de générer un vecteur-ligne de longueur `n` dont toutes les composantes sont des 0. (De manière analogue, on a aussi la commande `ones()`).

☞ On utilise donc une boucle `for` pour "remplir" les composantes selon le mode de génération de la suite.

Exercice 1. Écrire une suite d'instruction qui permette d'obtenir les 20 premiers termes de la suite (u_n) sous forme de vecteur-ligne où la suite (u_n) définie par

$$\begin{cases} u_1 &= 5 \\ u_{n+1} &= 3u_n - 2, \end{cases}$$

Exercice 2. Écrire une suite d'instruction qui permette d'obtenir les 16 premiers termes de la suite (u_n) sous forme de vecteur-ligne où la suite (u_n) définie par

$$\begin{cases} u_0 &= 1 \\ u_1 &= 2 \\ u_{n+2} &= u_{n+1} - nu_n \end{cases}$$

Exercice 3. Écrire une suite d'instruction pour obtenir un vecteur-ligne dont les 25 composantes correspondent au 25 premiers termes de la suites (u_n) définie pour $n \geq 1$ par

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

1.1 Représentation graphique

Comme pour les lignes brisées permettant de représenter graphiquement des fonctions, la commande `plot2d()` permet de représenter graphiquement les termes d'une suite. En rajoutant un troisième argument à la fonction précisant le style choisi pour chaque point, on crée une figure représentant le nuage de points.

Exercice 4. Représenter graphiquement (et séparément) les 25 premiers termes de chacune des deux suites des Exercices 1 et 3.

Exercice 5. (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,'+')
```

1.2 Instructions relatives aux vecteurs

On dispose de plusieurs instructions pratiques relatives aux vecteurs (ou listes de nombres). Parmi celles prédéfinies sur `SciLab`, on utilisera notamment:

(1) L'instruction `length(u)` renvoie la longueur de `u` (c'est à dire le nombre de coefficients qui le composent).

(2) 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.`

(3) 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 $u_k > 10$.

(4) 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()`.

(5) Si a, b, c sont trois réels avec $b > 0$ et $a \leq c$, la commande `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.

⚠ **Attention!** On utilisera la commande bien tentante `sum()` avec vigilance. Elle ne prend en argument qu'une "liste" de valeurs déjà bien définie, on évitera donc d'écrire n'importe quoi du genre `sum(n^ 2)` pour calculer la somme des carrés des entiers consécutifs.

Exercice 6. On considère la suite définie par

$$u_0 = 1/4, \qquad u_{n+1} = 1 + \frac{2}{u_n}.$$

Déterminer le nombre de termes, parmi les 50 premiers, vérifiants $0 < 2 - u_n < 10^{-3}$.

Exercice 7. (Sommes)

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

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

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

Une application particulièrement utile de l'instruction `prod()` est l'exercice suivant:

Exercice 8. Écrire une commande très courte permettant de calculer $n!$.

2 Combinatoire

Exercice 9. (1) Écrire une instruction très simple permettant de calculer $n!$.

(2) Utiliser cette instruction pour en écrire deux fonctions `y=A(n,k)` et `y=C(n,k)`, prenant toutes deux pour arguments n et k, où n est un entier naturel et k un entier compris entre 0 et n et permettant de calculer respectivement A_n^k et $\binom{n}{k}$.

(3) Vérifier avec `SciLab` que

$$\sum_{k=0}^{n}\binom{n}{k} = 2^n \quad \text{et} \quad \sum_{k=0}^{n}(-1)^k\binom{n}{k} = 0.$$

(4) Calculer avec `Scilab`:

(a) La probabilité de gagner à l'*Euromillion* (pour remplir une grille, on choisit 5 nombres entre 1 et 50, puis deux nombres complémentaires entre 1 et 10);

(b) La probabilité d'obtenir une *quinte flush* (cinq cartes d'une même couleur qui se suivent) lors de la distribution d'une main de cinq cartes (avec un jeu de 32 cartes).

Exercice 10. Un nombre $x \in \mathbb{R}$ est appelé *nombre factoriel* s'il peut s'écrire comme la factorielle d'un nombre entier, c'est à dire si il existe $n \in \mathbb{N}$ tel que $x = n!$.

(1) Écrire un programme `SciLab` qui compte le nombre de nombres factoriels inférieurs ou égaux à 10^6, 10^9, 10^{12} et $17!$.

(2) Les nombres 3629300 et 39916800 sont-ils des nombres factoriels?

Exercice 11. (Triangle de Pascal)

(1) Rappeler la formule du triangle de Pascal.

(2) Écrire alors un programme qui affiche les m premières lignes du triangles de Pascal, où m est un entier positif rentré par l'utilisateur.

3 Appendice 1: Le paradoxe des anniversaires

L'objectif de cet exercice est de visualiser l'évolution de la probabilité qu'au moins deux personnes d'un groupe fêtent leur anniversaire le même jour, en fonction du nombre d'individus dans le groupe.

Partie 1 - Préliminaires probabilistes

On suppose, pour simplifier, qu'une année n'est constituée que de 365 jours et que les naissances sont réparties uniformément tout au long de l'année.

On considère alors un groupe de n individus et on appelle Ω l'ensemble des listes ordonnées constituées des n dates d'anniversaire des membres de ce groupe (on ne s'intéresse pas à l'année de naissance mais seulement au jour et au mois). L'hypothèse précédente nous mène à considérer une probabilité uniforme P sur $(\Omega, \mathcal{P}(\Omega))$.

(1) Quel est le cardinal de Ω?
(2) On s'intéresse à l'évènement A: "au moins deux personnes du groupe fêtent leur anniversaire le même jour". Quel est l'évènement contraire de A?
(3) Calculer, en fonction de n, la probabilité de $\overline{A}$. En déduire celle de A, notée p_n.

Partie 2 - Utilisation de `SciLab`

(1) À l'aide des résultats de la partie précédente, écrire une fonction `anniversaire()` qui prend en argument un nombre entier naturel `n` et renvoie la valeur de la probabilité p_n.
(2) Écrire un programme permettant de déterminer le nombre minimum de personnes nécessaires pour que la probabilité p_n soit supérieure à 50%. Même question avec 99%.
(3) Représenter graphiquement l'évolution de la suite (p_n) pour n entre 1 et 60.

4 Appendice 2: 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 `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 $A_0B_1A_1B_2....A_n$.

(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, ..., A_9.
(c) Interpréter le graphique obtenu.