TD26

Révisions de Scilab

Programmation

Exercice 25.1 (★ - Sur la série harmonique)

1. Écrire un programme qui prend comme paramètre un entier n et retourne $u_n = \sum_{k=1}^{n} \frac{1}{k} - \ln(n)$.

   Vérifier alors numériquement que la suite (u_n) est convergente, et donner une valeur approchée de sa limite.

2. Écrire une fonction qui prend comme paramètre un réel $A > 0$ et qui retourne la valeur du plus petit entier n tel que $\sum_{k=1}^{n} \frac{1}{k} > A$.

   Quel est le plus petit entier n tel que $\sum_{k=1}^{n} \frac{1}{k} > 15$?

Exercice 25.2 (★★ - Série harmonique alternée)

On considère la série harmonique alternée $\sum_{n\geq 1} \frac{(-1)^{n-1}}{n}$. On note $S = (S_n)_{n\in\mathbb{N}^*}$ la suite de ses sommes partielles.

1. Écrire une commande définissant un vecteur ligne `u` tel que pour tout $1 \leq i \leq 50$, `u(i)` soit égal à $\frac{(-1)^{i-1}}{i}$.

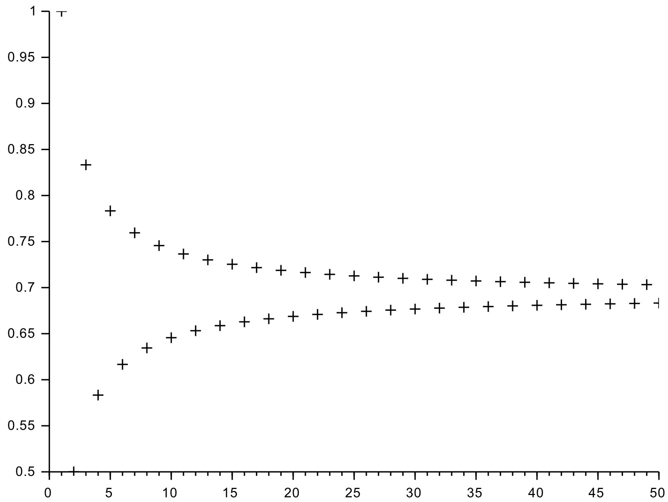
2. Écrire une commande définissant un vecteur ligne `v` tel que pour tout $1 \leq i \leq 50$, `v(i)` soit égal à $\sum_{k=1}^{i} \frac{(-1)^{k-1}}{k}$.

3. En exécutant la commande :

   ```
   plot2d([1:50],v,-1)
   ```

   on obtient le graphe ci-contre.

   Que peut-on dire des suites extraites (S_{2n}) et (S_{2n+1}) ? de la série S ?



4. On admet que $\sum_{n=1}^{+\infty} \frac{(-1)^{n-1}}{n} = \ln(2)$. Écrire un script demandant une valeur $\varepsilon > 0$ à l'utilisateur, et qui renvoie le plus petit entier naturel n pour lequel $|S_n - \ln(2)| \leq \varepsilon$.

Exercice 25.3 (★★)

Soit (u_n) la suite définie par récurrence par $u_0 = \frac{1}{2}$ et pour tout $n \in \mathbb{N}$, $u_{n+1} = u_n^2 - u_n \ln(u_n)$.

1. Écrire une fonction `suite` qui prend comme paramètre un entier n et renvoie la valeur de u_n correspondante.
2. On se propose de vérifier graphiquement que la suite (u_n) est croissante et tend vers 1.

   Écrire à cet effet un programme qui trace un graphique sur lequel se trouvent les points (n, u_n), pour $n \in [\![0, 100]\!]$.
3. Écrire une fonction nommée `plus_petit_n` qui prend comme paramètre un entier p et retourne la plus petite valeur de n pour laquelle $|1 - u_n| < 10^{-p}$.

Exercice 25.4 (★★)

Soit (u_n) la suite définie par $u_0 = u_1 = u_2 = 1$ et pour tout $n \in \mathbb{N}$, $u_{n+3} = u_{n+2} - 2u_{n+1} + u_n$.
Écrire une fonction d'entête `function y = suite_rec(n)` qui prend comme paramètre $n \in \mathbb{N}$ et retourne la valeur de u_n.

Exercice 25.5 (★★)

Soient (a_n) et (b_n) les suites définies par $a_0 = 2$, $b_0 = 3$ et :

$$\forall n \in \mathbb{N}, \quad a_{n+1} = \sqrt{a_n b_n} \quad \text{et} \quad b_{n+1} = \frac{a_n + b_n}{2}.$$

Écrire une fonction d'entête `function [a,b] = suites(n)` prenant comme paramètre un entier n et retournant les valeurs de a_n et b_n correspondantes.
Constater que ces suites convergent vers une même limite.

Exercice 25.6 (★★★ - Extrait d'Edhec 2019)

Dans cet exercice, n désigne un entier naturel non nul.

1. On admet que, si a et b sont des entiers tels que $a < b$, la commande `grand(1,1,'uin',a,b)` permet à `Scilab` de simuler une variable aléatoire suivant la loi uniforme discrète sur $[\![a, b]\!]$.
   Compléter le script suivant pour que les lignes (5), (6), (7) et (8) permettent d'échanger les contenus des variables $A(j)$ et $A(p)$.

```
1  n=input('entrez une valeur pour n :')
2  A=1:n
3  p=n
4  for k=1:n
5      j=grand(1,1,'uin',1,p)
6      aux=----
7      A(j)=----
8      A(p)=----
9      p=p-1
10 end
11 disp(A)
```

2. On suppose dorénavant qu'après exécution du script précédent correctement complété, le vecteur A est rempli de façon aléatoire par les entiers de $[\![1,n]\!]$ de telle sorte que les $n!$ permutations soient équiprobables.
On considère alors les commandes `Scilab` suivantes (exécutées à la suite du script précédent) :

```
m=A(1)
c=1
for k=2:n
    if A(k)>m then m=A(k)
                   c=k
    end
end
disp(c)
```

   (a) Expliquer pourquoi, à la fin de la boucle `for`, la variable `m` contient la valeur n.

   (b) Quel est le contenu de la variable c affiché à la fin de ces commandes ?

   (c) On rappelle qu'en `Scilab`, l'instruction `find(test)` permet de trouver à quelle(s) place(s) se trouvent les éléments d'une matrice satisfaisant au test proposé.
   Compléter le script `Scilab` ci-dessous afin qu'il renvoie et affiche le contenu de la variable `c` étudiée plus haut :

```
c=find(---)
disp(c)
```

On admet que les contenus des variables `A(1)`, `A(2)`, ..., `A(n)` sont des variables aléatoires notées $A_1,\ A_2,\ \ldots,\ A_n$ et que le nombre d'affectations concernant la variable informatique c effectuées au cours du script présenté au début de la question (7), y compris la première, est aussi une variable aléatoire, notée X_n.
On suppose que ces variables aléatoires sont toutes définies sur le même espace probabilisé $(\Omega, \mathscr{A}, P)$.

3. Donner la loi de X_1.

4. (a) Montrer que $X_n(\Omega) = [\![1,n]\!]$.

   (b) Déterminer $P(X_n = 1)$ et $P(X_n = n)$. En déduire les lois de X_2 et X_3.

   (c) En considérant le système complet d'événements $((A_n = n), (A_n < n))$, montrer que:

$$\forall n \geq 2,\ \forall j \in [\![2,n]\!],\ P(X_n = j) = \frac{1}{n} P(X_{n-1} = j-1) + \frac{n-1}{n} P(X_{n-1} = j)$$

   (d) Donner la loi de X_4.

Simulation de variables aléatoires

Exercice 25.7 (★)

En utilisant `rand()`, mais sans `grand()`, écrire une fonction d'entête `function y = binomiale(n,p)` qui simule une réalisation d'une loi $\mathscr{B}(n,p)$. *Indication : on se rappellera qu'une loi binomiale correspond au nombre de succès lors d'une répétition d'épreuves de Bernoulli indépendantes.*

Exercice 25.8 (★★)

1. Compléter la fonction suivante pour qu'elle simule une réalisation d'une loi géométrique de paramètre p.

```
function y = geom(p)
      y = 1 ;
      while rand() -----
            y = -----
      end
endfunction
```

2. On dit qu'une variable suit la loi binomiale négative de paramètres n et p si elle a la même loi que $\sum_{i=1}^{n} X_i$ où les X_i sont des variables i.i.d. suivant la loi $\mathscr{G}(p)$.

   Écrire un programme d'entête `function y = bin_neg(n,p)` qui simule une réalisation d'une loi binomiale négative de paramètres n et p.

Exercice 25.9 (★★)

Une urne contient initialement des boules numérotées de 2 à n. On effectue un tirage dans cette urne, et on enlève de l'urne toutes les boules portant un numéro supérieur ou égal à celui de la boule tirée. On ajoute alors la boule numéro 1 dans l'urne, et on effectue un nouveau tirage, et on note X le numéro de la boule obtenue.
Écrire un programme qui simule la variable aléatoire X.

Exercice 25.10 (★★)

Une urne contient 18 boules rouges et 2 boules vertes. On effectue des tirages sans remise dans cette urne, jusqu'à vider l'urne, et on note alors X le rang d'apparition de la première boule verte, et Y le rang d'apparition de la seconde boule verte.

1. Compléter la fonction suivante de sorte que la commande `un_tirage(r,v)` simule un tirage dans une urne contenant r boules rouges et v boules vertes, et retourne 0 si la boule tirée est rouge et 1 si la boule tirée est verte.

```
function y = un_tirage(r,v)
      if rand()< --- then
            y = --- ;
      else
            y = --- ;
      end
endfunction
```

2. Compléter le programme suivant pour qu'il simule la variable aléatoire X.

```
function y = tirage_verte()
    y = 1 ;
    nb_rouges = 18 ;
    nb_vertes = 2 ;
    while ---
        nb_rouges = ---
        y = ---
    end
endfunction
```

3. Adapter le programme de la question précédente pour qu'il simule une réalisation de la variable Y.

4. Écrire un programme qui à l'aide de 10000 simulations de Y, donne une valeur approchée de son espérance et de sa variance.

Exercice 25.11 (★★)

On lance n fois une pièce équilibrée, et on note X le nombre de fois où l'on obtient deux résultats identiques consécutifs.
Écrire un programme qui simule la variable aléatoire X. Utiliser ensuite ce programme pour estimer la valeur de $E(X)$.

Exercice 25.12 (★★)

Si $(X_1, \dots, X_n)$ sont des variables aléatoires indépendantes suivant la loi $\mathscr{N}(0,1)$, la loi suivie par la variable aléatoire $\sum_{i=1}^{n}$ est appelée loi du χ^2 (prononcer « chi-deux ») de paramètre n.

1. Écrire une fonction function `y = chi2(n)` qui prend en paramètre un entier $n \geq 1$, et qui simule une variable suivant la loi $\chi^2(n)$.

2. Écrire une fonction qui simule la variable T_p , où $T_p = \max(Y_1, \dots, Y_p)$, où $Y_1, \dots, Y_p$ sont p variables aléatoires indépendantes suivant la loi $\chi^2(n)$.

3. Proposer une méthode pour obtenir une valeur approchée de $E(T_p)$.

Exercice 25.13 (★★ - Loi de Rayleigh)

1. Soit $\sigma > 0$. Montrer que la fonction définie sur $\mathbb{R}_+$ par $x \mapsto \dfrac{1}{\sigma} x e^{-\frac{x^2}{2\sigma}}$ est une densité de probabilité.

   La loi d'une variable aléatoire admettant une telle densité est appelée loi de Rayleigh de paramètre σ. Dans toute la suite, on note X une variable aléatoire suivant une telle loi.

2. Donner la fonction de répartition F_X de X.

3. (a) Montrer que F_X réalise une bijection de $\mathbb{R}_+^*$ dans $]0,1[$. Déterminer une expression explicite de F_X^{-1}.

   (b) Soit U une variable aléatoire suivant une loi $\mathscr{U}(]0,1[)$. Montrer que $F_X^{-1}(U)$ suit la même loi que X.

   (c) Écrire une fonction `Scilab` d'entête `rayleigh(sigma,n)` qui, étant donné un réel $\sigma > 0$ et un entier $n \in \mathbb{N}^*$, simule n réalisation d'une loi de Rayleigh de paramètre σ.

(d) Vérifier la pertinence de cette simulation en comparant l'histogramme des fréquences et la densité, puis les fonctions de répartition empirique et théorique.

4. Estimer numériquement l'espérance et l'écart-type d'une variable aléatoire suivant une loi de Rayleigh de paramètre σ. Retrouver ces résultats par le calcul.

5. (a) Si X suit une loi de Rayleigh de paramètre σ, quelle est la loi de $Y = X^2$.

   (b) À l'aide de la question précédente, proposer une nouvelle fonction `Scilab` d'entête `rayleigh2(sigma)` qui, étant donné un réel $\sigma > 0$, simule une réalisation d'une loi de Rayleigh de paramètre σ.

Exercice 25.14 (★★★ - QSP HEC 2016)

Donner la finalité du programme suivant :

```
N = 100000 ; S = 0 ;
for i=1:N
      u = rand() ;
      S = S + (4/N)*1/(1+u^2) ;
end
disp(S)
```

On pourra penser à la loi faible des grands nombres.
