L2 M-MA
2024-2025

Programmation sous Python

Contrôle continu

Durée : 2h30.

L'utilisation de la documentation sur Moodle et Internet en général est autorisée, à condition qu'elle se limite à des recherches liées à Python.

Le devoir est INDIVIDUEL et une comparaison automatique des fichiers sera effectuée avant la correction. Aucune communication (mails, forum, IA...) n'est autorisée.

Les questions où la mention "sur feuille" apparaît sont à faire sur la copie fournie. Toute réponse doit être justifiée. Les correcteurs tiendront compte de la qualité de la rédaction et de la présentation.

Les quatre exercices sont indépendants.

À l'issue de l'épreuve, vous déposerez votre fichier `Nom_Prénom.py` *ou* `Nom_Prénom.ipynb` *dans une section prévue à cet effet sur Moodle (tout autre format de rendu sera sanctionné !).*

Exercice 1. L'objectif est de déduire par simulation une approximation de l'aire du domaine $\mathcal{D}$ délimité par

$$\mathcal{C} := \left\{(x,y) \in \mathbb{R}^2, \quad x^2 + \left(y - \sqrt[3]{x^2}\right)^2 = 1\right\}.$$

1. *(sur feuille)* Montrer que
$$\begin{cases} x = \cos(t) \\ y = \sin(t) + \sqrt[3]{\cos^2(t)}, \quad t \in [0, 2\pi] \end{cases}$$
est une paramétrisation de $\mathcal{C}$.

   Indication : On fera attention au fait qu'une inclusion de l'image dans $\mathcal{C}$ seule ne permet pas de conclure !

2. *(sur feuille)* Montrer que $\mathcal{D}$ est inclus dans un rectangle de longueur et largeur à déterminer.

   On simule ici un tir à l'arc. On suppose que le tireur à l'arc, qui n'est pas un expert, répartit ses flèches uniformément dans un rectangle contenant $\mathcal{C}$: le domaine $\mathcal{D}$ est la cible.

3. Écrire une fonction `tracercible()` qui représente graphiquement le rectangle ainsi que $\mathcal{C}$ (en couleur bleue) avec des labels appropriés dans un répère orthonormé quadrillé avec des noms aux axes et des titres adéquats.

4. Écrire une fonction `simulerfleches(n)` qui simule n tirs et qui affiche sur le graphique uniquement les flèches ayant atteint la cible.

5. En déduire une approximation de l'aire du domaine $\mathcal{D}$.

Exercice 2. Un joueur dispose de k dés (à six faces) avec $k \geq 5$. Au premier lancer, il met de côté le chiffre le plus souvent obtenu. Par exemple, s'il obtient $(3, 4, 5, 2, 3)$, il met de côté les deux "3". S'il a obtenu 5 dés identiques, il s'arrête, sinon, il relance les dés restants et met de côté les dés identiques à ceux mis de côté au premier lancer (dans le premier exemple, si à son second lancer, il obtient $(2, 6, 3)$ alors, il met le 3 de côté). Et ainsi de suite...
Le joueur se lance le défi suivant : réussir à obtenir k chiffres identiques en trois coups au plus. Le but de cet exercice est d'estimer la probabilité de réussir ce défi.

1. Écrire une fonction `premierlancer(k)` pour simuler le lancer de k dés (à six faces) et retournant une liste $[d, n]$ où d est le nombre apparaissant le plus souvent et n, le nombre d'apparitions de ce nombre.

   Par exemple, si le lancer donne $(3, 4, 5, 2, 3)$*, alors la fonction doit retourner* $[3, 2]$*. En cas d'égalité, vous devrez faire un choix. On pourra par exemple choisir le plus petit : dans ce cas, si le résultat du lancer est* $(2, 5, 5, 2, 3, 2, 5)$*, alors la fonction retourne* $[2, 3]$.

2. Écrire une fonction `lancersuivant(k,d,n)` prenant en entrée trois entiers k, d et n, simulant le lancer de $k-n$ dés et renvoyant le nombre de d obtenus. Par exemple, si $(k,d,n)=(5,3,2)$, alors la fonction simule le lancer de $3=5-2$ dés et renvoie le nombre de 3 obtenus.

3. Simulez le défi proposé par le joueur. Il s'agira d'écrire une fonction `defi(k)` prenant en entrée le nombre k de dés et renvoyant `True` si le défi est réussi en trois coups au plus et `False` sinon.

4. Écrivez une fonction `probadefi(k,N)` prenant en entrée k et N et renvoyant la proportion de défis réussis après N simulations.

5. Donnez votre estimation de la probabilité de réussir le défi pour $N=10^4$ simulations lorsque $k=5$.

Exercice 3. Soit $n\in\mathbb{N}$. Pour approximer l'intégrale d'une fonction f sur un intervalle, on cherche une fonction polynomiale de degré au plus n qui coïncide avec f en $n+1$ points : une subdivision régulière de l'intervalle. Concrètement, sur l'intervalle $[s,t]$ pour $s<t$, on cherche une fonction P_n telle que

$$\forall x\in[s,t],\quad P_n(x)=\sum_{i=0}^{n}a_ix^i$$

et

$$\forall i\in\{0,\dots,n\},\quad P_n\left(s+\frac{(t-s)}{n}i\right)=f\left(s+\frac{(t-s)}{n}i\right).$$

En posant pour tout $i\in\{0,\dots,n\}$,

$$y_i:=s+\frac{(t-s)}{n}i,$$

ces égalités donnent lieu à $n+1$ équations que l'on peut résumer par le système

$$\begin{pmatrix}y_0^n & y_0^{n-1} & \cdots & 1\\ y_1^n & y_1^{n-1} & \cdots & 1\\ \vdots & \vdots & \vdots & \vdots\\ y_n^n & y_n^{n-1} & \cdots & 1\end{pmatrix}\begin{pmatrix}a_n\\ a_{n-1}\\ \vdots\\ a_0\end{pmatrix}=\begin{pmatrix}f(y_0)\\ f(y_1)\\ \vdots\\ f(y_n)\end{pmatrix}.$$

1. *(sur feuille)* Justifier brièvement que pour tout $(n+1)$-uplet $(y_0,\dots,y_n)$ de réels distincts, la matrice

$$\begin{pmatrix}y_0^n & y_0^{n-1} & \cdots & 1\\ y_1^n & y_1^{n-1} & \cdots & 1\\ \vdots & \vdots & \vdots & \vdots\\ y_n^n & y_n^{n-1} & \cdots & 1\end{pmatrix}$$

est inversible.

2. Écrire une fonction `P(n,f, s, t)` qui prend en paramètres un entier naturel n, une fonction f et les bornes d'un segment $[s,t]$, et qui renvoie le $(n+1)-$uplet $(a_n,\dots,a_0)$ au format `numpy.array`, donné par la relation ci-dessus.

3. Écrire une fonction `IP(n,f, s, t)` qui prend en paramètres un entier naturel n et une fonction f ainsi que les bornes d'un intervalle $[s,t]$ et qui renvoie l'intégrale de la fonction polynomiale obtenue par `P(n,f, s, t)`.

4. On peut en déduire la stratégie suivante pour approcher une intégrale : on construit une discrétisation d'un intervalle $[a,b]$ en $n+1$ points de la forme $a=x_0<x_1<\ldots<x_{n-1}<x_n=b$, puis on considère que l'on peut approcher l'intégrale $\displaystyle\int_a^b f(x)\,\mathrm{d}x$ par la somme $\displaystyle\sum_{i=1}^{n}\int_{x_{i-1}}^{x_i}P_{k_i}(x)\,\mathrm{d}x$ où les fonctions P_{k_i} sont précisément les fonctions polynomiales de degré au plus k_i (entier naturel donné) évoquées précédemment, correspondant aux intervalles $[s=x_{i-1},t=x_i]$.

   (a) Écrire une fonction `IF(f, a, b, n,k)` qui renvoie l'approximation de l'intégrale de f sur $[a,b]$ à l'aide d'une discrétisation en n intervalles (donc en $n+1$ points $x_0,\dots,x_n$ avec $k=(k_1,\dots,k_n)$).

(b) Tester votre méthode en évaluant

$$I = \int_0^1 \frac{x\,\tan(x)}{\sqrt{1+x^2}}\,\mathrm{d}x.$$

Indication : $I \approx 0.33871$

5. Écrire une fonction `tracerIF(f, a, b, n,k)` qui propose une représentation graphique adéquate de la fonction f sur $[a, b]$, ainsi que toutes les fonctions P_{k_i} sur les n intervalles issus de la discrétisation.

Exercice 4. Nous disposons d'un nuage de n points aléatoires $(x_i, y_i)_{i=1;\ldots;n}$ tels que pour tout i, $x_i \sim \mathcal{N}(i, 2)$. Nous voulons les modéliser par une loi polynomiale :

$$y_i = P(x_i) + \varepsilon_i$$

avec $\varepsilon_i \sim \mathcal{N}(0, \sigma)$ (écart type $\sigma > 0$ donné) et

$$P : x \mapsto \sum_{j=0}^{d} a_j x^j$$

une fonction polynomiale réelle. On peut interpréter matriciellement le problème :

$$\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} 1 & x_1 & \cdots & x_1^d \\ 1 & x_2 & \cdots & x_2^d \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_n & \cdots & x_n^d \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_d \end{pmatrix} + \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{pmatrix}.$$

Soit de manière compacte :

$$y = Aa + \varepsilon$$

avec

$$y := \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}; \quad A := \begin{pmatrix} 1 & x_1 & \cdots & x_1^d \\ 1 & x_2 & \cdots & x_2^d \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_n & \cdots & x_n^d \end{pmatrix}; \quad a := \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_d \end{pmatrix}; \quad \varepsilon := \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{pmatrix}.$$

L'objectif est de trouver une fonction polynomiale dont les coefficients sont choisis de sorte qu'ils minimisent l'erreur quadratique :

$$\|\varepsilon\|_2^2 := \sum_{i=1}^{n} \varepsilon_i^2 = \sum_{i=1}^{n} \Big(y_i - P(x_i)\Big)^2 = \|y - Aa\|_2^2.$$

1. Écrire une fonction `simulernuage(a, n)` renvoyant les vecteurs $x := (x_1, \ldots, x_n)$ et $y = (y_1, \ldots, y_n)$.
Indication : pour simuler une variable aléatoire suivant la loi normale $\mathcal{N}(\mu, \sigma)$ où μ est l'espérance et σ est l'écart type (ou "déviation standard"), on pourra utiliser ***`numpy.random.normal`***

2. *(sur feuille)* Montrer que

$$\ker(A) = \{0_{\mathbb{R}^{d+1}}\} \iff \Big(n \geq d+1 \quad \text{et au moins} \quad d+1 \quad \text{des} \quad x_i \quad \text{sont distincts.}\Big)$$

On peut ensuite montrer que cela est aussi équivalent à $A^T A$ inversible.

3. Pour tout $n \geq d+1$, si au moins $d+1$ des x_i sont distincts, on peut montrer qu'il existe un unique $(d+1)$-uplet $a^\star := (a_0^\star, \ldots, a_d^\star)$ satisfaisant la minimisation de l'erreur quadratique :

$$a^\star = (A^T A)^{-1} A^T y.$$

Écrire une fonction `nuage(a, n)` simulant un nuage de $n \geq d+1$ points $(x_i, y_i)_{1 \leq i \leq n}$ tels qu'au moins $d+1$ des x_i sont distincts, définissant le polynôme $P^\star$ dont les coefficients sont donnés par $a^\star$ et renvoyant sur le même graphique le nuage de points et la courbe représentative du polynôme : les $(x_i, y_i)_{1 \leq i \leq n}$ et les $(x_i, P^\star(x_i))_{1 \leq i \leq n}$.

4. Qu'en est-il du cas $n < d+1$?