DS6 – Mathématiques

Mercredi 5 Mars 2025

Durée : 3 heures

- **Aucun document ou appareil électronique n'est autorisé.**
- Le devoir comporte un exercice d'informatique, deux exerices de mathématiques et un problème.
- Une notice Python est fournie en dernière page, que vous pouvez utiliser librement.
- **Utilisez des feuilles doubles uniquement. Rédigez l'exercice d'informatique sur une copie double séparée.**
- La qualité de la présentation et de la rédaction seront prises en compte dans la note finale.

Exercice 1 (Informatique)**.** Dans cet exercice, on suppose que le module `numpy` est importé à l'aide de l'instruction

```
1 import numpy as np
```

1. Écrire une fonction `decompte` prenant en entrée un tableau Numpy de nombres et qui renvoie le nombre de zéros contenus dans le tableau. Par exemple si $T = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 3 \end{pmatrix}$, alors l'instruction `decompte(T)` doit renvoyer 6.
2. Ecrire une instruction qui permette de stocker dans la variable T la matrice de la question précédente.
3. Écrire une fonction `lignes_egales` prenant en entrée deux listes A et B contenant le même nombre p d'éléments, et qui renvoie `True` si les deux listes sont identiques (c'est-à-dire si tous leurs éléments sont égaux) et `False` sinon.

   Par exemple si $A1 = [1,2,3,4,5]$ et $B1 = [1,2,3,4,5]$ alors `lignes_egales(A1,B1)` doit renvoyer `True` tandis que si $A2 = [2,0,1,3]$ et $B2 = [1,2,3,4]$ alors `lignes_egales(A2,B2)` doit renvoyer `False`.
4. Écrire une fonction `existe_lignes_egales` prenant en entrée un tableau Numpy à n lignes et p colonnes, et qui renvoie `True` s'il existe deux lignes identiques et `False` sinon.

   Par exemple si $A = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 2 \\ 1 & 1 & 1 \end{pmatrix}$ alors `existe_lignes_egales(A)` doit renvoyer `True` tandis que si $B = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ alors `existe_lignes_egales(B)` doit renvoyer `False`.

   (Indication : on pourra utiliser la fonction écrite à la question précédente.)

Exercice 2.

1. Résoudre l'équation différentielle $y'-2y=-6$ puis résoudre le problème de Cauchy $\begin{cases} y'-2y=-6, \\ y(0)=5. \end{cases}$
2. Résoudre l'équation différentielle $y''-5y'+6y=9$ puis résoudre le problème de Cauchy $\begin{cases} y''-5y'+6y=9 \\ y(0)=\dfrac{9}{2} \\ y'(0)=8. \end{cases}$

Exercice 3. On définit la suite $(u_n)_{n\in\mathbb{N}}$ par $u_0 \in \mathbb{R}$ et $\forall\, n \geq 1, u_{n+1} = \dfrac{3}{4}u_n^2 - 2u_n + 3$. Notons f la fonction définie sur $\mathbb{R}$ par $f(x) = \dfrac{3}{4}x^2 - 2x + 3$.

1. Étudier la fonction f.
2. Étudier le signe de la fonction $g : x \mapsto f(x) - x$.
3. Calculer les limites éventuelles de la suite (u_n).
4. On suppose que $u_0 = 2$. Que dire de la suite (u_n) ?
5. On suppose que $u_0 > 2$.
   (a) Montrer que la suite est bien définie et que pour tout $n \in \mathbb{N}$, $u_n > 2$.
   (b) Étudier la monotonie de la suite (u_n).
   (c) Étudier le comportement à l'infini de la suite (u_n).
6. On suppose que $u_0 \in \left[\frac{2}{3}, 2\right[$.
   (a) Montrer que la suite est bien définie et que pour tout $n \in \mathbb{N}$, $u_n \in \left]\frac{2}{3}, 2\right[$.
   (b) Étudier la monotonie de la suite (u_n).
   (c) Étudier le comportement à l'infini de la suite (u_n).

Problème. On considère les matrices $J = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$ et $M = \begin{pmatrix} -1 & 1 & 1 \\ 1 & -1 & 1 \\ 1 & 1 & -1 \end{pmatrix}$.

Dans cet exercice, on cherche à calculer M^n pour tout $n \in \mathbb{N}$ par trois méthodes différentes.

1 Première méthode

1. Déterminer deux réels α et β tels que $M^2 + \alpha M + \beta I_3 = 0_{\mathcal{M}_3(\mathbb{R})}$.
2. La matrice M est-elle inversible? Si oui, indiquer son inverse. *(Indication : il n'est pas nécessaire d'utiliser l'algorithme du pivot de Gauss pour répondre à cette question).*
3. Montrer que, pour tout entier $n \in \mathbb{N}$, il existe deux réels a_n et b_n tels que $M^n = a_n M + b_n I_3$.
4. On considère les suites (a_n) et (b_n) définies par :

$$\begin{cases} a_0 = 0 \\ b_0 = 1 \end{cases} \quad \text{et pour tout } n \in \mathbb{N}, \quad \begin{cases} a_{n+1} = b_n - a_n, \\ b_{n+1} = 2a_n. \end{cases}$$

   Montrer que la suite (a_n) est récurrente linéaire d'ordre 2 .
   En déduire une expression de a_n et b_n en fonction de $n \in \mathbb{N}$, puis l'expression de M^n en fonction de M et I_3 et $n \in \mathbb{N}$.

2 Deuxième méthode

1. Pour tout $n \geqslant 1$, exprimer J^n en fonction de J.
   Cette formule est-elle valable pour $n = 0$?
2. Déterminer deux réels a et b tels que $M = aI_3 + bJ$.
3. En utilisant la formule du binôme de Newton, exprimer M^n en fonction de I_3 et de J pour tout $n \in \mathbb{N}$.
4. Vérifier que cette expression est cohérente avec celle trouvée avec la première méthode.

3 Troisième méthode

1. Pour $\lambda \in \mathbb{R}$, on considère $M - \lambda I_3 = \begin{pmatrix} -1-\lambda & 1 & 1 \\ 1 & -1-\lambda & 1 \\ 1 & 1 & -1-\lambda \end{pmatrix}$.
   Montrer qu'il existe exactement deux valeurs de λ pour lesquelles $M - \lambda I_3$ est non-inversible. Préciser ces valeurs.

2. Dans cette question et la suivante, on note $X = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$.
   Vérifier que le système linéaire $MX = X$ équivaut au système linéaire $(M - I_3)X = 0_{\mathcal{M}_{3,1}(\mathbb{R})}$. Montrer que ce système admet une infinité de solutions, et déterminer l'ensemble de ses solutions.

3. De même qu'à la question précédente, montrer que le système linéaire $MX = -2X$ admet une infinité de solutions, et déterminer l'ensemble de ces solutions.

Soit $P = \begin{pmatrix} 1 & 1 & 0 \\ -2 & 1 & -1 \\ 1 & 1 & 1 \end{pmatrix}$.

4. Montrer que P est inversible et calculer son inverse.

5. Déterminer la matrice D définie par $D = P^{-1}MP$, et justifier que $M = PDP^{-1}$. Quelle propriété particulière possède la matrice D ?

6. Montrer que : $\forall\, n \in \mathbb{N}, M^n = PD^nP^{-1}$.

4 Une application :

On considère les trois suites réelles (u_n), (v_n) et (w_n) définies par :

$$u_0 = 1, v_0 = 0, w_0 = 0 \quad \text{et} \quad \forall\, n \in \mathbb{N}, \begin{cases} u_{n+1} = -u_n + v_n + w_n \\ v_{n+1} = u_n - v_n + w_n \\ w_{n+1} = u_n + v_n - w_n \end{cases}$$

1. On souhaite réaliser en Python une fonction `suites(n)` qui prend en argument un entier n, et renvoie une liste composée des valeurs de u_n, v_n et w_n. Recopier et compléter le programme suivant afin qu'il accomplisse cette tâche :

```
def suites(n):
    u = ...
    v = ...
    w = ...
    for .............. :
        nouveau_u = ..........
        nouveau_v = ..........
        nouveau_w = ..........
        u = nouveau_u
        v = nouveau_v
        w = nouveau_w
    return [u, v, w]
```

2. Pour tout $n \in \mathbb{N}$, on pose $X_n = \begin{pmatrix} u_n \\ v_n \\ w_n \end{pmatrix}$.
   Déterminer, pour tout $n \in \mathbb{N}$, une relation entre X_{n+1}, X_n et M.

3. Montrer que pour tout $n \in \mathbb{N}$, $X_n = M^n X_0$.

4. En déduire l'expression des termes généraux de $(u_n), (v_n), (w_n)$.

PYTHON
AGRO-VETO
2023

Listes

`[]` -------- Créer une liste vide
`[a]*n` ----- Créer une liste avec n fois l'élément `a`
`L.append(a)` Ajoute l'élément `a` à la fin de la liste `L`
`L1 + L2` --- Concatène les deux listes `L1` et `L2`
`len(L)` ---- Renvoie le nombre d'éléments de la liste `L`
`L.pop(k)` -- Renvoie le $k^{\text{ème}}$ élément de la liste `L` et l'enlève de `L`
`L.remove(a)` Enlève une fois la valeur `a` de la liste `L`
`max(L)` ---- Renvoie le plus grand élément de la liste `L`
`min(L)` ---- Renvoie le plus petit élément de la liste `L`
`sum(L)` ---- Renvoie la somme de tous les éléments de la liste `L`

Numpy

`import numpy as np`
`np.array()` ------------- Transforme une liste en matrice `numpy`
`np.linspace(a,b,n)` ----- Crée une matrice ligne de n valeurs uniformément réparties entre a et b (inclus)
`np.zeros([n,m])` -------- Crée la matrice nulle de taille $n \times m$
`np.eye(n)` -------------- Crée la matrice identité de taille n
`np.diag(L)` ------------- Crée la matrice diagonale dont les termes diagonaux sont les éléments de la liste `L`
`np.transpose(M)` -------- Renvoie la transposée de M
`np.dot(M,P)` ------------ Renvoie le produit matriciel MP
`np.sum(M)` -------------- Renvoie la somme de tous les éléments de M
`np.prod(M)` ------------- Renvoie le produit de tous les éléments de M
`np.max(M)` -------------- Renvoie le plus grand élément de M
`np.min(M)` -------------- Renvoie le plus petit élément de M
`np.shape(M)` ------------ Renvoie dans un couple le format de la matrice M
`np.size(M)` ------------- Renvoie le nombre d'éléments de M

Numpy.linalg

`import numpy.linalg as la`
`la.inv(M)` -------------- Renvoie l'inverse de la matrice M si elle est inversible
`la.eigvals(M)` ---------- Renvoie la liste des valeurs propres de M
`la.eig(M)` -------------- Renvoie un couple `L,P` où `L` est la liste des valeurs propres de M et `P` la matrice de passage associée
`la.matrix_rank(M)` ------- Renvoie le rang de M

Random

`import random as rd`
`rd.random()` ------- Simule une réalisation d'une variable $X \hookrightarrow \mathcal{U}([0,1])$
`rd.randint(a,b)` --- Simule une réalisation d'une variable $X \hookrightarrow \mathcal{U}(\llbracket a,b \rrbracket)$
`rd.gauss(0,1)` ----- Simule une réalisation d'une variable $X \hookrightarrow \mathcal{N}(0,1)$
`rd.choice(L)` ------ Choisit aléatoirement un élément de la liste `L`

Math

`import math as m`
`m.atan(x)` ------- Renvoie $\arctan(x)$
`m.floor(x)` ------ Renvoie $\lfloor x \rfloor$
`m.factorial(n)` -- Renvoie $n!$ si $n \in \mathbb{N}$
`m.sqrt(x)` -- Renvoie $\sqrt{x}$ si $x \geq 0$
`m.log(x)` --- Renvoie $\ln(x)$ si $x > 0$
`m.exp(x)` --- Renvoie e^x

Logique

`a == b` ---- Teste l'égalité « $a = b$ »
`a != b` ---- Teste « $a \neq b$ »
`a < b` ----- Teste « $a < b$ »
`a <= b` ---- Teste « $a \leq b$ »
`a > b` ----- Teste « $a > b$ »
`a => b` ---- Teste « $a \geq b$ »
`not A` ----- Renvoie la négation de A
`A and B` --- Renvoie « A et B »
`A or B` ---- Renvoie « A ou B »
`True` ------ Constante booléenne « Vrai »
`False` ----- Constante booléenne « Faux »

Matplotlib.pyplot

`import matplotlib.pyplot as plt`
`plt.plot(X,Y,'+-r')` ---- Génère la courbe des points définis par les listes `X` et `Y` (abscisses et ordonnées) avec les options :
- symbole : `'.'` point, `'o'` rond, `'h'` hexagone, `'+'` plus, `'x'` croix, `'*'` étoile, ...
- ligne : `'-'` trait plein, `'--'` pointillé, `'-.'` alterné, ...
- couleur : `'b'` **bleu** , `'r'` **rouge** , `'g'` **vert** , `'c'` **cyan** , `'m'` **magenta** , `'k'` **noir** , ...

`plt.bar(X,Y)` ---------- Génère l'histogramme des points définis par les listes `X` et `Y` (abscisses et ordonnées)
`plt.axis('equal')` ------ Rend le repère orthonormé
`plt.xlim(xmin,xmax)` ---- Fixe les bornes de l'axe des abscisses
`plt.ylim(ymin,ymax)` ---- Fixe les bornes de l'axe des ordonnées
`plt.show()` ------------ Affiche le graphique

Cette liste est non exhaustive. Les candidats sont libres d'utiliser les commandes de leur choix.