Sujet 1 (complet)

Question de cours : Énoncer le lemme des coalitions

Exercice 1:

1. On considère les équations différentielles suivantes, d'inconnue $y : \mathbb{R} \to \mathbb{R}$:

$$(H) \quad y'' - 4y' + 5y = 0,$$
$$(E) \quad y'' - 4y' + 5y = 2 - e^{2x}.$$

a) Déterminer l'ensemble des solutions de l'équation (H).

b) En déduire l'ensemble des solutions de l'équation (E).
On pourra chercher une solution particulière y_0 de l'équation différentielle

$$y'' - 4y' + 5y = -e^{2x}$$

sous la forme $y_0 : x \mapsto ce^{2x}$, où c est un réel à déterminer.

2. Pour tout réel x, on note

$$C(x) = \int_0^x e^{2t}\cos(t)\,\mathrm{d}t \quad \text{et} \quad S(x) = \int_0^x e^{2t}\sin(t)\,\mathrm{d}t.$$

a) Montrer que, pour tout x réel,

$$C(x) = \frac{e^{2x}\cos(x) - 1}{2} + \frac{1}{2}S(x) \quad \text{et} \quad S(x) = \frac{e^{2x}\sin(x)}{2} - \frac{1}{2}C(x).$$

b) En déduire une primitive de $x \mapsto e^{2x}\cos(x)$ et une primitive de $x \mapsto e^{2x}\sin(x)$ sur $\mathbb{R}$.

3. Écrire une fonction en langage Python, nommée `intC`, prenant en paramètres un réel x et un entier `nb_pas`, qui renvoie une valeur approchée de l'intégrale $\int_0^x e^{2t}\cos(t)\,\mathrm{d}t$ obtenue à l'aide de la méthode des rectangles. L'intervalle $[0\,;\,x]$ doit être découpé en `nb_pas` intervalles de même longueur.

Dans toute la suite, on note E l'espace vectoriel des fonctions de classe C^∞ de $\mathbb{R}$ dans $\mathbb{R}$.
On considère les quatre fonctions suivantes de E :

$$f_1 : x \mapsto 1, \quad f_2 : x \mapsto e^{2x}, \quad f_3 : x \mapsto e^{2x}\cos(x), \quad f_4 : x \mapsto e^{2x}\sin(x).$$

On note F le sous-espace vectoriel de E engendré par la famille $\mathcal{B} = (f_1, f_2, f_3, f_4)$.

4. Montrer que $\mathcal{B}$ est une base de F.

5. On note u l'application définie sur F par $u(f) = f'$ pour tout $f \in F$.

a) Montrer que u est linéaire.

b) Calculer l'image par u de f_1, f_2, f_3 et f_4.

c) En déduire que u est un endomorphisme de F, et déterminer la matrice représentative A de l'endomorphisme u dans la base $\mathcal{B}$.

6. Résoudre l'équation $AX = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$, d'inconnue $X \in \mathcal{M}_{4,1}(\mathbb{R})$.
Quel résultat des questions précédentes retrouve-t-on ainsi ? Justifier.

7. Résoudre l'équation $\left(A^2 - 4A + 5I_4\right)X = \begin{pmatrix} 2 \\ -1 \\ 0 \\ 0 \end{pmatrix}$, d'inconnue $X \in \mathcal{M}_{4,1}(\mathbb{R})$.
Quel résultat des questions précédentes retrouve-t-on ainsi ? Justifier.

On pourra coller ci-dessous l'exercice sans préparation.

Question de cours : Donner la définition de fonction de densité de probabilité.

Exercice 2:

1. **a)** Soit f la fonction définie par

$$f : x \mapsto \frac{1}{1+x} + \frac{2}{2+x}.$$

Étudier la fonction f sur son ensemble de définition et dresser un tableau de variations complet.
En déduire le nombre de solutions de l'équation $f(x) = 1$.

b) Déterminer les solutions exactes de l'équation $f(x) = 1$.

Soit A_n la matrice de $\mathcal{M}_n(\mathbb{R})$ dont tous les coefficients diagonaux sont nuls et tels que les autres coefficients situés sur la colonne j sont égaux à j pour $j \in [\![1, n]\!]$.

2. **a)** Écrire une fonction Python d'argument n renvoyant A_n.

b) En déduire une valeur approchée des valeurs propres de A_n pour $n \in \{2, 3, 4, 10\}$.

3. Pour $n \geqslant 2$, on pose :

$$f_n : t \mapsto \sum_{k=1}^{n} \frac{k}{k+t}$$

et on considère l'équation E_n d'inconnue $\lambda \in \mathbb{R}$:

$$\sum_{k=1}^{n} \frac{k}{k+\lambda} = 1.$$

a) Montrer que l'équation E_n admet une unique solution dans chacun des intervalles $]-1, +\infty[$ et $]-k, -(k-1)[$ pour $k \in [\![2, n]\!]$.

b) On admet (pour l'instant) que toutes les valeurs propres de A_n sont les solutions de l'équation E_n. La matrice A_n est-elle diagonalisable ?

4. Pour $n \geqslant 2$, on appelle λ_n la solution de E_n comprise entre -2 et -1.

a) Justifier que pour tout entier naturel n non nul, pour tout réel t de $]-2, -1[$, $f_n(t) \leqslant f_{n+1}(t)$, et en déduire la monotonie de la suite (λ_n).

b) Montrer que la suite (λ_n) converge vers un réel ℓ.
Si l'on suppose que $\ell \in]-2, -1[$, comparer $f_n(\lambda_n)$ et $f_n(\ell)$.

c) Pour $\lambda \in]-2, -1[$, calculer $\lim\limits_{n\to+\infty} \sum\limits_{k=1}^{n} \frac{k}{k+\lambda}$ et conclure sur la valeur de ℓ.

d) Montrer que pour $k \in \mathbb{N}^*$,

$$\frac{1}{k} \leqslant 2\left(\sqrt{k} - \sqrt{k-1}\right)$$

et en déduire $\lim\limits_{n\to+\infty} \frac{1}{n} \sum\limits_{k=1}^{n} \frac{1}{k}$.

e) Déterminer un équivalent de $\frac{1}{1+\lambda_n}$ puis de $\lambda_n + 1$. On pourra utiliser un encadrement de $\sum\limits_{k=3}^{n} \frac{k}{k+\lambda_n}$.

5. On souhaite maintenant montrer ce qui a été admis en 3.(b).
Montrer, à l'aide du système traduisant la recherche des valeurs propres/vecteurs propres, que les valeurs propres de A_n sont les solutions de l'équation E_n.

On pourra coller ci-dessous l'exercice sans préparation.

Question de cours : Énoncer le théorème de Pythagore dans $\mathbb{R}^n$.

Exercice 3: Toutes les variables aléatoires de cet exercice sont définies sur un espace probabilisé $(\Omega, \mathcal{A}, P)$.
Soit X une variable aléatoire à densité, de densité f telle que :

$$f(x) = \begin{cases} \frac{2}{x^3} & \text{si } x > 1, \\ 0 & \text{sinon.} \end{cases}$$

1. Vérifier que f est une densité de probabilité.

2. Déterminer la fonction de répartition de X.

3. Soit U une variable aléatoire de loi uniforme sur $]0, 1[$.
Montrer que la variable aléatoire $V = \frac{1}{\sqrt{1-U}}$ suit la même loi que X.

4. Écrire un programme Python simulant une réalisation de la variable aléatoire X.

5. La variable aléatoire X admet-elle une espérance ? une variance ?
Si oui, les calculer, et vérifier vos réponses à l'aide du programme de la question 4.

6. Soit $(X_n)_{n\geqslant 1}$ une suite de variables aléatoires indépendantes de même loi que X. On définit, pour tout entier n non nul, la variable aléatoire T_n par :

$$T_n = \frac{\max(X_1, \ldots, X_n)}{\sqrt{n}}.$$

a) Soit $n \in \mathbb{N}^*$. Déterminer la fonction de répartition de T_n.

b) Calculer alors pour tout réel x :

$$G(x) = \lim_{n\to+\infty} P(T_n \leqslant x)$$

c) On **admet** que G est la fonction de répartition d'une variable aléatoire T à densité.
Montrer que T admet pour densité la fonction donnée par :

$$g(x) = \begin{cases} \frac{2}{x^3} e^{-\frac{1}{x^2}} & \text{si } x > 0, \\ 0 & \text{sinon.} \end{cases}$$

d) À l'aide du changement de variable $x = \frac{1}{u}$, montrer que T admet une espérance et la déterminer.

On pourra coller ci-dessous l'exercice sans préparation.

Exercice du sujet 1

On appelle *vecteurs creux* des vecteurs de très grande dimension, dont la grande majorité des coordonnées sont nulles. Pour coder un vecteur creux, on utilise une liste de deux listes, celle de ses coordonnées non nulles et celle des indices correspondants triée par ordre croissant, suivies de la taille du vecteur.
Ainsi, le vecteur de $\mathbb{R}^9$ $(1,0,3,0,0,0,0,-1,0)$ est représenté par `[[1,3,-1],[0,2,7],9]`. Les vecteurs usuels sont, eux, codés par des vecteurs `numpy`.

1. Ecrire une fonction `coder` renvoyant la représentation creuse d'un vecteur. Écrire une fonction `decoder` faisant le travail inverse.
Par exemple : `coder([1,0,3,0,0,0,0,-1,0])` doit renvoyer `[[1,3,-1],[0,2,7],9]` ; `decoder([[1,3,-1 ],[0,2,7],9])` doit renvoyer `[1,0,3,0,0,0,0,-1,0]`

Pour de tels vecteurs, il serait trop coûteux, tant en temps qu'en mémoire, d'écrire toutes les coordonnées nulles et de calculer comme habituellement dans $\mathbb{R}^n$. C'est pourquoi on utilise le codage de la question précédente.

2. Ecrire une fonction `prod_scal(V1,V2)` prenant en argument deux vecteurs creux appartenant au même espace $\mathbb{R}^n$, codés comme dans la première question et qui calcule leur produit scalaire.
Il n'est pas autorisé de repasser par la forme usuelle en décodant `V1` et `V2`.

Exercice du sujet 2

1. On considère des listes non vides ne contenant que 1 ou 2 valeurs différentes.
Par exemple `['Marwa', 'Ambre', 'Marwa', 'Ambre', 'Ambre']`.
Écrire, en langage Python, une fonction nommé `election1(L)` qui prend en entrée une liste `L` de cette forme et qui renvoie l'élément qui est majoritaire. La fonction doit renvoyer `None` en cas d'égalité.

2. On considère maintenant des listes non-vides, mais pouvant contenir plus de 2 valeurs différentes.
Par exemple `['Olivia', 'Gita', 'Mathys', 'Olivia', 'Mathys']`.
La liste représente une urne, et on veut savoir qui a obtenu le plus de voix.
Écrire, en lagage Python, une fonction nommée `election2(L)` qui prend une liste ayant cette forme et qui renvoie la liste des personnes ayant obtenu le maximum de voix (la liste contient plusieurs personnes en cas d'égalité).
Sur la liste donnée en exemple, la fonction doit renvoyer `['Olivia', 'Mathys']` *ou* `['Mathys', 'Olivia']` *car Olivia et Mathys sont ex-aeqo avec deux voix chacun.*

Exercice du sujet 3

À partir d'une séquence de nucléotides, on souhaite obtenir l'ensemble de toutes les chaînes qui se répètent de manière **consécutive** ainsi que leur **position de départ.** (position 0 = début de la séquence.)
Exemple :

séquence : TACG ACG ATACG :

2 répétitions **successives** de la chaîne ACG (et non 3 !)

On dit que ACG est une chaine périodique de longueur 3 (3 éléments), de période 2 (nombre de répétitions successives) et de position 1.

On admet que la longueur maximale d'une chaîne dans une séquence de longueur n ne peut excéder $\frac{n}{2}$.

1. Écrire une fonction `repetitions` d'arguments `(seq,pos,long)` (pour (séquence,position,longueur)), qui rend :
- `[]` s'il n'y a pas de chaîne répétitive à cet emplacement
- la liste `[chaine,période]` sinon.

Par exemple : Avec la séquence TACG ACG ACG ATACG :
- pour `pos=1` et `long=3`, la fonction doit rendre `['ACG',3]`
- pour `pos=4` et `long=3`, la fonction doit rendre `['ACG',2]`.

2. En déduire une fonction `chaines(sequence,long)` qui rend la liste des listes `[chaine, position, periode]` de longueur `long` dans la séquence. *(On ne se préoccupera pas des éventuelles répétitions de chaine).*

Exemple : Pour la séquence de démarrage TACG ACG ATACG, on obtiendra `[['ACG', 1, 2], ['CGA', 2, 2]]`. :