Concours Blanc

Epreuve de Mathématiques - Mercredi 30 avril 2025

Durée : 3 h

L'usage d'abaques, de tables, de calculatrice et de tout instrument électronique susceptible de permettre au candidat d'accéder à des données et de les traiter par les moyens autres que ceux fournis dans le sujet est interdit.

Chaque candidate ou candidat est responsable de la vérification de son sujet d'épreuve : pagination et impression de chaque page. Ce contrôle doit être fait en début d'épreuve. En cas de doute, il convient d'alerter au plus tôt l'équipe de surveillance qui vérifiera et, éventuellement, remplacera le sujet.

Ce sujet comporte 6 pages : 5 pages numérotées de 1 à 5 et une annexe Python.

Si, au cours de l'épreuve, une candidate ou un candidat repère ce qui lui semble être une erreur d'énoncé, elle ou il le signale sur sa copie et poursuit sa composition en expliquant les raisons des initiatives prises.

Ce sujet comporte 2 problèmes indépendants.

On pourra admettre le résultat d'une question ou d'une sous-question pour passer aux questions suivantes, en le mentionnant explicitement.

Une annexe dans laquelle certaines commandes Python sont rappelées est jointe à la fin du sujet. **Pour les questions d'informatique, on considérera que les importations de modules nécessaires ont été préalablement faites.**

Problème 1 (Dynamique des populations)**.** Dans ce problème, on propose d'étudier différents modèles d'évolution de population, et d'étudier les conditions de son extinction. Les questions sont largement indépendantes ; les sous-questions sont liées.

On s'intéresse d'abord à des modèles déterministes discrets d'évolution d'une population. Dans chacun des modèles, une suite (v_n) modélise le nombre d'individus dans la population à la génération n. On dit qu'il y a extinction si $\lim\limits_{n\to+\infty} v_n = 0$.

1. Pour commencer, on propose le modèle suivant : le nombre initial d'individus est noté $v_0 \in \mathbb{R}^+$ et à chaque génération, chaque individu a un nombre de descendants $q > 0$, de telle sorte que

$$\forall\, n \in \mathbb{N}, \quad v_{n+1} = q\, v_n.$$

   Résoudre le modèle (c'est-à-dire exprimer v_n en fonction de $n \in \mathbb{N}$) et donner une condition d'extinction à l'aide des paramètres du problème q et v_0.

2. On propose un nouveau modèle. La suite (v_n) est définie par :

$$v_0 \in \mathbb{R}^+ \quad \text{et} \quad \forall\, n \in \mathbb{N}, \quad v_{n+1} = v_n + \frac{1}{2} v_n \left(\frac{S - v_n}{S}\right)$$

   où $S \in \mathbb{R}^{+\star}$ est une constante du problème.

   (a) Déterminer une fonction f sur $\mathbb{R}^+$ telle que

$$\forall\, n \in \mathbb{N}, \quad v_{n+1} = f(v_n).$$

   Dresser le tableau de variations de f sur $\mathbb{R}^+$.

(b) Compléter le code suivant pour qu'il trace les 20 premiers termes de la suite.

```
S = 30
v0 = 5
L=[v0]
v=v0
for k in ## LIGNE A COMPLETER ##
    ## LIGNE A COMPLETER ##
    ## LIGNE A COMPLETER ##
plt.plot(L)
plt.xlabel("n")
plt.ylabel("v_n")
plt.show()
```

(c) Pour le tracé, on a utilisé le module `matplotlib.pyplot` sous l'alias `plt`. Comment importer le module ?

(d) On trace sur la même figure l'évolution de v_n pour différentes valeurs de v_0. Cela donne les courbes suivantes (pour $S = 30$), représentées sur la figure 1. Conjecturer sur le comportement de la suite.

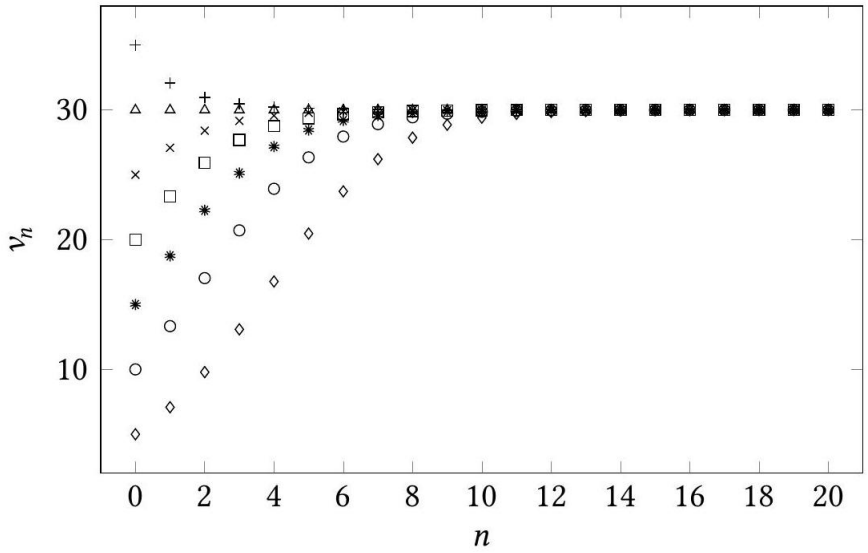


FIGURE 1 – Les valeurs de v_n pour $n \in [\![0,20]\!]$, pour différentes valeurs de v_0 (pour $S = 30$).

(e) On suppose maintenant que $v_0 \in]0, S]$. Montrer que (v_n) est croissante et majorée par S. En déduire qu'elle converge et donner sa limite.

3. On souhaite affiner le modèle en modifiant la fonction f. Désormais,

$$\forall\, n \in \mathbb{N}, \quad v_{n+1} = v_n + \frac{1}{2} v_n \left(\frac{S - v_n}{S}\right)\left(\frac{v_n - A}{S}\right)$$

où $A \in]0, S[$ est une constante fixée.

(a) Identifier la fonction f. Étudier le signe de $f(x) - x$ sur $[0, S]$. En déduire l'allure de la courbe représentative de f sur $[0, S]$. On fera attention à la position relative par rapport à la droite d'équation $y = x$ et on utilisera sans démonstration que f est une bijection strictement croissante de $[0, S]$ sur lui même.

Un code analogue au précédent donne les courbes suivantes pour les premiers termes de la suite (v_n) pour $S = 30$ et $A = 9$, représentées sur la figure 2. On suppose que $v_0 \in [0, S]$.

(b) Dans cette question $v_0 \in]0, A[$. Montrer que pour tout entier $n \in \mathbb{N}$, $v_n \in]0, A[$, puis que (v_n) est décroissante. En déduire que (v_n) converge et déterminer sa limite.

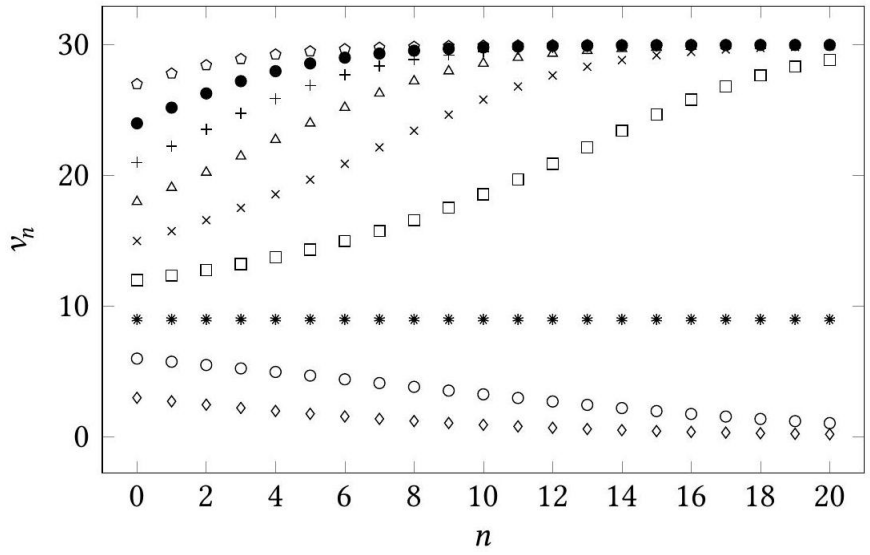


FIGURE 2 – Pour le modèle raffiné : les valeurs de v_n pour $n \in [\![0,20]\!]$, pour différentes valeurs de v_0 (pour $S = 30$ et $A = 9$).

(c) Réaliser une étude analogue lorsque $v_0 \in]A, S[$. Que se passe-t-il si $v_0 = A$?

(d) Donner une interprétation (en terme de dynamique des populations) des quantités A et S.

4. La version continue du modèle précédent est l'équation différentielle :

$$\forall\, t \in \mathbb{R}^+, \quad y'(t) = F(y(t))$$

avec

$$F(y) = \frac{y}{2}\left(\frac{S-y}{S}\right)\left(\frac{y-A}{S}\right) \text{ et une condition initiale } y(0) \geq 0.$$

(a) On appelle point d'équilibre de l'équation différentielle toute valeur x_0 telle que si $y(t) = x_0$ alors $y'(t) = 0$. Autrement dit, x_0 est un point d'équilibre signifie que $F(x_0) = 0$. Déterminer ces points d'équilibre.

(b) On dit qu'un point d'équilibre x_0 est instable si $F'(x_0) > 0$ et stable si $F'(x_0) < 0$. Déterminer les points stables et instables parmi les points d'équilibre.

(c) Si $y(0)$ est assez proche de x_0, y se comporte localement comme la solution de l'équation différentielle linéaire

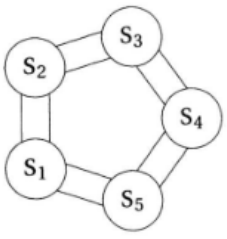
$$\forall\, t \in \mathbb{R}^+, \quad y'(t) = F'(x_0)\big(y(t) - x_0\big).$$

Résoudre cette équation différentielle (on pourra poser $a = F'(x_0)$, $c = x_0$ et remarquer qu'il s'agit d'une équation différentielle linéaire d'ordre 1 à coefficients constants).

(d) Justifier les dénominations d'équilibre stable et instable. Faire le lien avec les interprétations faites dans la question 3.

Problème 2 (Modélisation d'une rencontre aléatoire)**.**

Deux personnes P_1 et P_2 ont rendez-vous dans un complexe formé de cinq sites S_1, S_2, S_3, S_4 et S_5, disposés en pentagone et reliés par des routes, comme l'illustre le schéma ci-dessous. Ils arrivent au rendez-vous à l'heure prévue, mais suite à un malentendu, P_1 se présente au site S_1 et P_2 au site S_2.



Ils décident alors de partir à la recherche l'un de l'autre. Ils empruntent les différentes routes du complexe, avec les règles suivantes

- à partir d'un site, chacun choisit de se rendre sur l'un des deux sites voisins, les deux possibilités étant équiprobables ;
- les déplacements des deux personnes se font simultanément ;
- tous les choix de déplacement se font indépendamment les uns des autres.

Ils continuent à se déplacer ainsi jusqu'à se retrouver éventuellement sur un même site (ils ne se rencontrent pas le long des routes). Une fois retrouvés, ils ne se déplacent plus.

Pour tout entier naturel n, on définit les trois événements A_n, B_n, C_n :

- A_n : "les deux personnes sont sur le même site après le n-ième déplacement" ;
- B_n : "les deux personnes sont sur des sites adjacents après le n-ième déplacement" ;
- C_n : "les deux personnes sont à deux routes de distance après le n-ième déplacement".

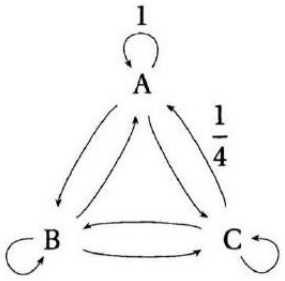
On note a_n, b_n, c_n les probabilités des événements A_n, B_n, C_n.

1. Justifier que A_n, B_n, C_n forment un système complet d'événements.
2. Déterminer les valeurs de a_0, b_0 et c_0.
3. (a) Justifier soigneusement que pour tout $n \in \mathbb{N}$,

$$P_{C_n}(A_{n+1}) = \frac{1}{4} \quad \text{et} \quad P_{A_n}(A_{n+1}) = 1.$$

Que vaut $P_{B_n}(A_{n+1})$?

(b) Déterminer toutes les probabilités conditionnelles analogues. On représentera les résultats en reproduisant et complétant le schéma ci-dessous.



4. Établir les relations suivantes pour tout entier $n \in \mathbb{N}$:

$$\begin{cases} a_{n+1} = a_n + \frac{1}{4}c_n \\ b_{n+1} = \frac{3}{4}b_n + \frac{1}{4}c_n \\ c_{n+1} = \frac{1}{4}b_n + \frac{1}{2}c_n \end{cases}.$$

5. (a) Exprimer b_{n+2} à l'aide de b_{n+1}, b_n et c_n puis exprimer c_n en fonction de b_{n+1} et b_n pour obtenir enfin une relation entre b_{n+2}, b_{n+1} et b_n.

(b) En déduire une expression de b_n en fonction de n. On fera intervenir les nombres

$$\alpha = \frac{5-\sqrt{5}}{8} \text{ et } \beta = \frac{5+\sqrt{5}}{8}.$$

(c) Montrer que pour tout $n \in \mathbb{N}$,

$$c_n = \frac{\sqrt{5}}{5}\left(\beta^n - \alpha^n\right).$$

6. À partir de la somme $a_n + b_n + c_n$, déterminer la limite, lorsque n tend vers l'infini, de la suite (a_n).
7. On note R l'évènement "les deux personnes se rencontrent à un moment".

(a) Justifier que pour, pour tout $n \in \mathbb{N}$, $A_n \subset R$.

(b) En déduire que pour tout $n \in \mathbb{N}$,

$$a_n \leq P(R) \leq 1.$$

(c) Quelle est la probabilité que les deux personnes ne se retrouvent jamais ?

8. On cherche à simuler la situation précédente en Python.

(a) À l'aide du module `random`, écrire une fonction Python `pas` qui renvoie 1 avec probabilité 1/2 et -1 avec probabilité 1/2.

(b) Écrire une fonction Python `avance` qui prend en argument la position `p` d'un des deux individus à un instant donné et qui renvoie sa position à l'instant suivant (ici `p` désigne un entier compris entre 1 et 5 - on pourra utiliser la fonction `pas`).

(c) On propose le code suivant pour simuler l'expérience :

```
12 def trajectoire():
13     #instant initial
14     n = 0
15     p1 = 1 # position initiale de la personne 1
16     p2 = 2 # position initiale de la personne 2
17     while ##LIGNE A COMPLETER##:
18         n += 1
19         p1 = avance(p1)
20         p2 = avance(p2)
21     return n
```

Compléter la ligne 17 pour que la fonction `trajectoire` renvoie l'instant `n` auquel les deux personnes se rencontrent.

(d) Justifier que le code précédent termine avec probabilité 1.

(e) Proposer une facon d'estimer la probabilité que les deux personnes se rencontrent en moins de 10 déplacements.

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.