Banque « Agro-Véto »
A - 0715

MATHÉMATIQUES

Modélisation Mathématique et Informatique

Durée : 3 heures 30 minutes

Chaque 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 doit alerter au plus tôt le chef de centre qui contrôlera et éventuellement remplacera le sujet.

Si, au cours de l'épreuve, un candidat repère ce qui lui semble être une erreur d'énoncé, il le signale sur sa copie et poursuit sa composition en expliquant les raisons des initiatives qu'il a été amené à prendre.

L'usage d'une calculatrice est autorisé pour cette épreuve.

Le deuxième problème demande de choisir l'un des langages de programmation Python, SciLab ou MatLab. On écrira le langage choisi au tout début de la copie.

Les trois problèmes peuvent se traiter indépendamment.

Problème 1 : Évolution de populations structurées en âge.

Dans une population, tous les individus n'ont pas le même potentiel de reproduction, ni la même probabilité de survie d'une année à l'autre. De même, l'âge est un facteur important. Nous choisissons alors une unité de temps $u \in \mathbf{R}^+$ et nous décidons de découper la population en p classes d'âges de même amplitude (à l'exception de la dernière classe), $p \in \mathbf{N} - \{0, 1\}$.

Une classe d'âge C_i, $i \in [\![1, p-1]\!]$, est donc caractérisée par la donnée d'un âge minimum $(i-1)u$, et la donnée d'un âge maximum $iu - 1$. La classe d'âge C_p est caractérisée par la seule donnée d'un âge minimum $(p-1)u$.

Par exemple, dans le cadre d'une population d'êtres humains, nous choisissons une unité de temps u de 15 années et nous découpons la population en 5 classes d'âge ($p = 5$). La classe C_1 inclut les êtres humains d'âge compris entre 0 et 14 ans, la classe C_2 les êtres humains d'âge compris entre 15 et 29 ans, la classe C_3 les êtres humains d'âge compris entre 30 et 44 ans, la classe C_4 les êtres humains d'âge compris entre 45 et 59 ans, la classe C_5 incluant les êtres humains âgés d'au moins 60 ans.

Pour $s \in \mathbf{N}$, nous décidons de représenter la population après s unités de temps par la matrice $N_s \in \mathcal{M}_{p,1}(\mathbf{R})$ définie par :

$$N_s = \begin{pmatrix} n_{s,1} \\ n_{s,2} \\ \vdots \\ n_{s,p} \end{pmatrix},$$

où pour tout $i \in [\![1, p]\!]$, $n_{s,i}$ désigne l'effectif de la i^{e} classe d'âge de la population au début de la s^{e} unité de temps.

N_0 représente donc l'état initial de la population, N_1 la population après 1 unité de temps, N_2 la population après 2 unités de temps, *etc.*

1. Pour $s \in \mathbf{N}$, exprimer l'effectif total de la population après s unités de temps.

Pour tout $i \in [\![1, p]\!]$, F_i désigne le nombre de naissances par unité de temps pour un individu de la i^e classe d'âge.

Pour tout $i \in [\![1, p-1]\!]$, P_i désigne la probabilité de survie pour un individu entre la classe d'âge C_i et la classe d'âge C_{i+1}.

On note P_p la probabilité de survie au sein de la p^e et dernière classe d'âge. Les survivants restent alors au sein de cette classe d'âge.

2. Soit $s \in \mathbf{N}$.

2.1 Exprimer $n_{s+1,1}$ en fonction de $n_{s,1}, n_{s,2}, \ldots, n_{s,p}$ et de $F_1, F_2, \ldots, F_p$.

Dans notre modèle, on suppose que les effectifs sont suffisamment importants, et on fera les approximations $n_{s+1,i} = n_{s,i-1} \times P_{i-1}$ pour $2 \leq i \leq p-1$ et $n_{s+1,p} = n_{s,p-1} \times P_{p-1} + n_{s,p} \times P_p$.

2.2 Quel théorème justifie l'approximation $n_{s+1,i} = n_{s,i-1} \times P_{i-1}$ (pour $2 \leq i \leq p-1$) dans le cas où la population est très grande ? Expliquer.

2.3 Proposer une matrice $M \in \mathcal{M}_p(\mathbf{R})$ telle que : $N_{s+1} = MN_s$.

3. Étude d'un exemple.

Nous considérons une population de drosophiles, la durée de vie d'une drosophile est inférieure à 30 jours.

Nous choisissons une unité de temps u de 10 jours et nous découpons la population en 3 classes d'âge.

Après étude statistique, nous estimons que :

$$F_1 = 0\,,\ F_2 = 13\,,\ F_3 = 12\,,\ P_1 = \frac{1}{4}\,,\ P_2 = \frac{1}{2}, P_3 = 0.$$

3.1 Montrer que pour tout entier naturel s : $N_{s+1} = \begin{pmatrix} 0 & 13 & 12 \\ \frac{1}{4} & 0 & 0 \\ 0 & \frac{1}{2} & 0 \end{pmatrix} N_s$.

Dans la suite, A désigne la matrice $\begin{pmatrix} 0 & 13 & 12 \\ \frac{1}{4} & 0 & 0 \\ 0 & \frac{1}{2} & 0 \end{pmatrix}$.

3.2 Montrer que pour tout entier naturel s : $N_s = A^s N_0$.

3.3 Réduction de la matrice A.

3.3.1 Démontrer que A possède comme valeurs propres les nombres 2 et $-\frac{1}{2}$, et déterminer la troisième valeur propre de A.

Les valeurs propres de A seront notées λ_1, λ_2, λ_3 telles que :

$$\lambda_1 > |\lambda_2| \geq |\lambda_3|.$$

La matrice A est-elle diagonalisable ? La réponse sera justifiée.

3.3.2 Déterminer une base de chaque sous-espace propre de A.

Pour tout entier i compris entre 1 et 3, V_i désigne une matrice de $\mathcal{M}_{3,1}(\mathbf{R})$ dont l'élément situé en 3e ligne est égal à 1 et appartenant au sous-espace propre de la matrice A associé à la valeur propre λ_i.

Montrer que la famille $\mathcal{V} = (V_1, V_2, V_3)$ est une base de $\mathcal{M}_{3,1}(\mathbf{R})$.

3.4 Comportement asymptotique de la population.

Notons (c_1, c_2, c_3) les composantes de N_0 dans la base $\mathcal{V}$.

3.4.1 Montrer que : $\forall s \in \mathbf{N}$, $N_s = (\lambda_1)^s\, c_1\, V_1 + (\lambda_2)^s\, c_2\, V_2 + (\lambda_3)^s\, c_3\, V_3$.

3.4.2 En déduire que pour tout $s \in \mathbf{N}$ on peut écrire

$$N_s = (\lambda_1)^s\, (c_1\, V_1 + \varepsilon_s)\,,$$

où tous les coefficients de la matrice ε_s ont pour limite 0 lorsque s tend vers $+\infty$.

3.4.3 Montrer que les différents rapports $\dfrac{n_{s,1}}{n_{s,3}}$ et $\dfrac{n_{s,2}}{n_{s,3}}$ ont une limite finie lorsque s tend vers $+\infty$ et la calculer.

Ces limites indépendantes notamment de N_0 montrent que la répartition de la population tend vers une répartition constante et indépendante de la population initiale.

Problème 2 : Calcul numérique d'éléments propres.

On souhaite dans ce problème écrire un programme permettant de calculer, sous certaines conditions, un vecteur propre d'une matrice. Les différents programmes demandés devront être écrits, au choix, en *Python*, en *SciLab* ou en *MatLab*. Quelques fonctions utiles sont rappelées en annexe à la fin du sujet. *On rappelle que chaque programme doit être commenté par une phrase détaillant le raisonnement qui a conduit à son élaboration.*

Notons $\mathcal{M}_{m,n}(\mathbf{R})$ l'ensemble des matrices de taille $m \times n$ à coefficients réels, et $\mathcal{M}_p(\mathbf{R}) = \mathcal{M}_{p,p}(\mathbf{R})$ l'ensemble des matrices carrées. À chaque matrice $\mathrm{M} = (m_{ij}) \in \mathcal{M}_{m,n}(\mathbf{R})$ on associe un nombre réel positif $\|\mathrm{M}\|$, appelé *norme* de M, défini par

$$\|\mathrm{M}\| = \max_{i,j} |m_{ij}|.$$

Remarquons que $\|\mathrm{M}\|$ est toujours strictement positif, sauf lorsque la matrice M est nulle.

1. Écrire une fonction `Norme(M)` qui étant donnée une matrice M de taille quelconque calcule et renvoie le nombre $\|\mathrm{M}\|$. *On interdit le recours à une quelconque fonction* `max` *prédéfinie pour cette question.*
2. Écrire une fonction `Normalise(v)` qui étant donnée une matrice colonne $v \in \mathcal{M}_{p,1}(\mathbf{R})$ non nulle renvoie une nouvelle matrice colonne $\tilde{v}$, de même forme, égale à

$$\tilde{v} = \frac{v}{\|v\|}.$$

On se donne à présent une matrice carrée $\mathrm{A} \in \mathcal{M}_p(\mathbf{R})$. Soit v_0 un élément quelconque de $\mathcal{M}_{p,1}(\mathbf{R})$. En supposant qu'aucun des termes n'est dans le noyau de A, on peut former la suite $(v_n)_{n\geq 0}$ d'éléments de $\mathcal{M}_{p,1}(\mathbf{R})$ définie par la relation de récurrence

$$v_{n+1} = \frac{\mathrm{A}v_n}{\|\mathrm{A}v_n\|}.$$

3. Écrire une fonction `PuissanceIteree(A, n)` qui étant donnée une matrice carrée A et un entier naturel n, détermine la taille p de A, choisit aléatoirement une matrice colonne $v_0 \in \mathcal{M}_{p,1}(\mathbf{R})$, puis calcule et renvoie, en supposant que tous les termes de la suite ci-dessus sont bien définis, la matrice colonne v_n.

On peut montrer que si $\mathrm{A} \in \mathcal{M}_p(\mathbf{R})$ est diagonalisable, et possède les valeurs propres $\lambda_1, \ldots, \lambda_p$ avec

$$|\lambda_1| > |\lambda_2| \geq \cdots \geq |\lambda_p|,$$

alors la suite $(v_n)_{n\geq 0}$ ci-dessus existe et ses composantes convergent vers les composantes d'un vecteur propre v de A associé à la valeur propre λ_1, sauf pour quelques choix de v_0. La probabilité, en choisissant aléatoirement v_0, de tomber sur l'une des exceptions est nulle.

On se propose d'écrire maintenant une fonction `VecteurPropre(A, e)` qui étant donnée une matrice carrée A satisfaisant les hypothèses décrites ci-dessus et un nombre $e > 0$ calcule les termes de la suite $(v_n)_{n\geq 0}$ jusqu'à ce que deux termes successifs vérifient $\|v_n - v_{n+1}\| < e$, et renvoie alors la matrice colonne v_{n+1}. Voici trois propositions de programmes.

Version A

Python	SciLab / MatLab

`def VecteurPropre(A, e) :`
`  d = A.shape`
`  v = matrix(random.rand(d[0],1))`
`  v = Normalise(v)`
`  w = Normalise(A*v)`
`  while Norme(v - w) >= e :`
`  v = w`
`  w = Normalise(A*v)`
`  return w`
	`function w=VecteurPropre(A, e)`
	`  d = size(A)`
	`  v = rand(d(1), 1)`
	`  v = Normalise(v)`
	`  w = Normalise(A*v)`
	`  while Norme(w - v) >= e`
	`  v = w`
	`  w = Normalise(A*v)`
	`  end`

Version B

Python	SciLab / MatLab

`def VecteurPropre(A, e) :`
`  d = A.shape`
`  v = matrix(random.rand(d[0],1))`
`  v = Normalise(v)`
`  w = Normalise(A*v)`
`  ecart = Norme(v - w)`
`  while ecart >= e :`
`  v = w`
`  w = Normalise(A*v)`
`  return w`
	`function w=VecteurPropre(A, e)`
	`  d = size(A)`
	`  v = rand(d(1), 1)`
	`  v = Normalise(v)`
	`  w = Normalise(A*v)`
	`  ecart = Norme(w - v)`
	`  while ecart >= e`
	`  v = w`
	`  w = Normalise(A*v)`
	`  end`

Version C

Python	SciLab / MatLab

`def VecteurPropre(A, e) :`
`  d = A.shape`
`  v = matrix(random.rand(d[0],1))`
`  v = Normalise(v)`
`  while Norme(v - Normalise(A*v)) >= e :`
`  v = Normalise(A*v)`
`  return Normalise(A*v)`
	`function w=VecteurPropre(A, e)`
	`  d = size(A)`
	`  v = rand(d(1), 1)`
	`  v = Normalise(v)`
	`  while Norme(v - Normalise(A*v)) >= e`
	`  v = Normalise(A*v)`
	`  end`
	`  w = Normalise(A*v)`

4. Parmi ces trois programmes, indiquer lequel est (ou lesquels sont) correct(s). Pour chaque programme *incorrect* on indiquera succinctement ce qui ne va pas.

Problème 3 : Étude d'une loi de probabilité et application.

Dans ce problème, $\mathbf{N}$, $\mathbf{N}^*$, $\mathbf{R}$, $\mathbf{R}_+^*$ désignent respectivement les ensembles des entiers naturels, des entiers naturels non nuls, des nombres réels, des nombres réels strictement positifs.

Pour $\alpha \in \mathbf{R}$ et $r \in \mathbf{R}_+^*$, on définit la fonction $\ell_{\alpha,r}$ sur $\mathbf{R}$ par

$$\ell_{\alpha,r}(t) = \begin{cases} \alpha \dfrac{t^{\alpha-1}}{r^\alpha} & \text{si } t \in]0, r[, \\ 0 & \text{sinon.} \end{cases}$$

1. Montrer que $\ell_{\alpha,r}$ est une densité de probabilité sur $\mathbf{R}$ si et seulement si $\alpha > 0$.

Dans toute la suite de ce problème, on supposera cette condition réalisée.

On dit qu'une variable aléatoire réelle X suit la *loi $\mathscr{L}$ de paramètres α et r* si X admet $\ell_{\alpha,r}$ pour densité. On notera

$$X \hookrightarrow \mathscr{L}(\alpha, r)$$

cette propriété.

Dans toute la suite de ce problème, on note X une variable aléatoire de loi $\mathscr{L}(\alpha, r)$.

2. On s'intéresse tout d'abord aux propriétés élémentaires de cette loi.

2.1 Déterminer la fonction de répartition F de X.

2.2 Que vaut $P(X \leq 0)$?

2.3 Reconnaître la loi de la variable aléatoire $W = -\ln\left(\dfrac{X}{r}\right)$.

3. Soit $\lambda, \mu \in \mathbf{R}_+^*$. Soient U_1 et U_2 deux variables aléatoires réelles indépendantes qui suivent des lois exponentielles de paramètres λ et μ respectivement.

3.1 Donner une densité de $-U_2$.

3.2 On rappelle que si V_1 et V_2 sont deux variables aléatoires réelles indépendantes de densités respectives v_1 et v_2, alors $Z = V_1 + V_2$ est encore une variable aléatoire réelle à densité, dont une densité est donnée par $z : t \mapsto \int_{-\infty}^{+\infty} v_1(u) v_2(t-u)\,du$. Montrer qu'une densité de $U_1 - U_2$ est donnée par

$$g_{\lambda,\mu} : t \mapsto \begin{cases} \dfrac{\lambda\mu}{\lambda+\mu} e^{-\lambda t} & \text{si } t > 0, \\ \dfrac{\lambda\mu}{\lambda+\mu} e^{\mu t} & \text{si } t \leq 0. \end{cases}$$

3.3 Soient X et Y deux variables aléatoires réelles indépendantes, respectivement de lois $\mathscr{L}(\alpha, r)$ et $\mathscr{L}(\beta, s)$ avec $(\alpha, \beta, r, s) \in (\mathbf{R}_+^*)^4$. À l'aide de la question qui précède et de la question 2.3., donner la loi de $Z = -\ln\left(\dfrac{sX}{rY}\right)$ et en déduire que

$$P(X < Y) = \begin{cases} \dfrac{\beta}{\alpha+\beta}\left(\dfrac{s}{r}\right)^\alpha & \text{si } s < r, \\ 1 - \dfrac{\alpha}{\alpha+\beta}\left(\dfrac{r}{s}\right)^\beta & \text{si } r \leq s. \end{cases}$$

4. On applique maintenant le résultat de la question précédente à un modèle de test de dépistage d'une maladie canine H au sein d'une population $\mathcal{P}$. La présence de cette maladie chez un individu de $\mathcal{P}$ se note par la modification de la loi de concentration de deux bactéries A et B présentes dans l'estomac. Plus précisément, ces concentrations respectives X et Y, en $\text{UFC}\,\text{ml}^{-1}$ (Unité Formant Colonie par millilitre), sont deux variables aléatoires réelles indépendantes de lois respectives $\mathscr{L}(\alpha, r)$ et $\mathscr{L}(\beta, s)$, avec $(\alpha, \beta, r, s) \in (\mathbf{R}_+^*)^4$. On a statistiquement les valeurs suivantes :

- Pour les sujets *non atteints* de la maladie H :

$$\alpha = 2 \quad ; \quad \beta = 3 \quad ; \quad r = 100\,\text{UFC}\,\text{ml}^{-1} \quad ; \quad s = 50\,\text{UFC}\,\text{ml}^{-1}.$$

- Pour les sujets *atteints* de la maladie H :

$$\alpha = 4 \quad ; \quad \beta = 2 \quad ; \quad r = 400\,\text{UFC}\,\text{ml}^{-1} \quad ; \quad s = 800\,\text{UFC}\,\text{ml}^{-1}.$$

Le test T consiste à effectuer un prélèvement sanguin d'un individu C de $\mathcal{P}$. Une fois ce prélèvement mis en culture dans des conditions adéquates, les deux bactéries A et B entrent en concurrence. Au bout de quelques heures, seule celle dont la concentration était la plus forte subsiste, l'autre ayant totalement disparu. Cette procédure permet donc de savoir lequel des événements $\{X < Y\}$ ou $\{X > Y\}$ est réalisé pour C. On note $R = \{X < Y\}$. Le test T est positif si R est réalisé, négatif sinon. On note M l'événement « le sujet $C \in \mathcal{P}$ est atteint de la maladie H ». L'événement contraire d'un événement E sera systématiquement noté $\overline{E}$ dans la suite.

4.1 Donner les probabilités conditionnelles $P(R|M) = P_M(R)$ et $P(R|\overline{M}) = P_{\overline{M}}(R)$.

4.2 Un sondage a permis d'estimer $P(M) = \frac{1}{40}$. Donner la probabilité qu'un sujet testé soit atteint de la maladie H sachant que son test T est positif. Qu'en pensez-vous ?

Annexe : rappel de quelques commandes utiles en Python, SciLab et MatLab

On donne ci-dessous la description de quelques fonctions qui pourront être utiles dans le deuxième problème. En Python, on utilise la librairie *NumPy*, qui est supposée ouverte.

Interprétation	Python	SciLab / MatLab
Construction d'une nouvelle matrice de taille $m \times n$, ne contenant que des zéros	`matrix(zeros([m, n]))`	`zeros(m, n)`
Matrice identité de taille p	`matrix(eye(p))`	`eye(p, p)`
Construction d'une nouvelle matrice de taille $m \times n$, remplie avec des coefficients choisis aléatoirement dans $[0\,;1[$	`matrix(random.rand(m, n))`	`rand(m, n)`
Copie de la matrice A dans une nouvelle matrice B	`B = matrix(A)`	`B = A`

Dimensions de la matrice A
– nombre de lignes
– nombre de colonnes
	`d = A.shape`
	`d[0]`
	`d[1]`
		`d = size(A)`
		`d(1)`
		`d(2)`

Opérations matricielles (pour des matrices A et B de tailles compatibles)	`A+B, A-B, A*B`	`A+B, A-B, A*B`
Coefficient d'indices (i, j) dans la matrice M	`M[i,j]`	`M(i,j)`

Par ailleurs, on rappelle que pour une matrice M à m lignes et n colonnes, les indices vont de 0 à $m-1$ pour les lignes et de 0 à $n-1$ pour les colonnes en Python, alors qu'ils vont de 1 à m pour les lignes et de 1 à n pour les colonnes en SciLab et en MatLab.

On rappelle enfin qu'il est facultatif de mettre un `endfunction` (respectivement un `end`) à la fin d'une fonction en SciLab (respectivement en MatLab).

***** FIN *****