Code sujet : 287

BCE
BANQUE COMMUNE D'ÉPREUVES

Conception : ESSEC BS

MATHÉMATIQUES 2 APPLIQUÉES

FILIÈRE ÉCONOMIQUE ET COMMERCIALE

VOIE GÉNÉRALE

Vendredi 24 avril 2026 de 14h à 18 h

La présentation, la lisibilité, l'orthographe, la qualité de la rédaction, la clarté et la précision des raisonnements entreront pour une part importante dans l'appréciation des copies.
Les candidats sont invités à ***encadrer*** *dans la mesure du possible les résultats de leurs calculs.*
Aucun document n'est autorisé. ***L'utilisation de toute calculatrice et de tout matériel électronique est interdite.*** *Seule l'utilisation d'une règle graduée est autorisée.*
Si au cours de l'épreuve, un candidat repère ce qui lui semble être une erreur d'énoncé, il la signalera sur sa copie et poursuivra sa composition en expliquant les raisons des initiatives qu'il sera amené à prendre.

Tourner la page SVP

Le sujet s'intéresse à un problème dit du "bandit manchot" qui est un exemple d'apprentissage par renforcement.

L'apprentissage par renforcement est un sujet largement utilisé dans le domaine de l'intelligence artificielle. Cela consiste schématiquement à apprendre, à partir d'expériences, quelles actions sont à réaliser pour optimiser une récompense quantitative au cours du temps.

Pour les scripts et fonctions Python, on supposera que les instructions suivantes ont été exécutées :

```
import numpy as np,numpy.random as rd
```

Un aide-mémoire Python et SQL se trouve à la fin de l'énoncé.

Les événements et variables aléatoires qui interviennent dans ce problème sont tous et toutes définis sur le même espace probabilisé $(\Omega, \mathcal{A}, \mathbb{P})$.

Si X est une variable aléatoire réelle sur cet espace, $\mathbb{E}(X)$ désigne son espérance lorsque celle-ci existe.

Le mot "Fin" marque la fin de l'énoncé.

Résultats généraux

On rappelle l'inégalité de Markov :

si X est une variable aléatoire à valeurs positives admettant une espérance et $a > 0$ un réel alors

$$\mathbb{P}(X \geqslant a) \leqslant \frac{\mathbb{E}(X)}{a}$$

1. Soit $k \in \mathbb{N}^*$ et $B_1, \ldots, B_k$ des événements. Montrer que $\mathbb{P}\left(\bigcup_{i=1}^{k} B_i\right) \leqslant \sum_{i=1}^{k} \mathbb{P}(B_i)$.

► Soit $\theta \in [0,1]$.

2. On définit pour tout $t \in \mathbb{R}$, $f(t) = \frac{t^2}{8} + \theta t - \ln\left(1 - \theta + \theta e^t\right)$.

   a) Montrer que f est de classe C^2 sur $\mathbb{R}$ et que pour tout t réel : $f''(t) = \frac{(1-\theta-\theta e^t)^2}{4(1-\theta+\theta e^t)^2}$.

   b) En déduire que pour tout t réel, $f(t) \geqslant 0$ puis que

$$(1-\theta)e^{-\theta t} + \theta e^{(1-\theta)t} \leqslant \exp\left(\frac{t^2}{8}\right)$$

3. a) Justifier brièvement que la fonction $t \mapsto e^t$ est convexe sur $\mathbb{R}$.

   b) En déduire que pour tout $x \in [-\theta, 1-\theta]$ et $t \in \mathbb{R}$, $e^{tx} \leqslant (\theta+x)e^{(1-\theta)t} + (1-\theta-x)e^{-\theta t}$.

4. Soit X une variable aléatoire d'espérance nulle à valeurs dans le segment $[-\theta, 1-\theta]$. En utilisant les deux questions précédentes, montrer que pour tout $t \in \mathbb{R}$, $\mathbb{E}(e^{tX})$ existe et

$$\mathbb{E}(e^{tX}) \leqslant \exp\left(\frac{t^2}{8}\right)$$

Code sujet : 287

BCE
BANQUE COMMUNE D'ÉPREUVES

Conception : ESSEC BS

MATHÉMATIQUES 2 APPLIQUÉES

FILIÈRE ÉCONOMIQUE ET COMMERCIALE

VOIE GÉNÉRALE

Vendredi 24 avril 2026 de 14h à 18h

RECTIFICATIF

Modification de la question 1

Dans la relation située après "Montrer que", il faut remplacer chaque B_k par un B_i .

Donc cela fait deux B_i dans la relation à établir et aucun B_k.

▶ Soit $n \in \mathbb{N}^*$ et $X_1, ..., X_n$ des variables aléatoires indépendantes à valeurs dans $[0,1]$ et de même espérance μ. On définit $\overline{X}_n$ par $\overline{X}_n = \frac{1}{n}\sum_{k=1}^{n} X_k$.

5. **Inégalité de Hoeffding** - Soit $\varepsilon \geqslant 0$.

   a) En utilisant l'inégalité de Markov, montrer que, pour tout $t > 0$:

   $$\mathbb{P}(\overline{X}_n - \mu \geqslant \varepsilon) \leqslant \frac{\mathbb{E}\left(e^{t(\overline{X}_n - \mu)}\right)}{e^{t\varepsilon}} \text{ puis que } \mathbb{P}(\overline{X}_n - \mu \geqslant \varepsilon) \leqslant e^{-t\varepsilon} \prod_{k=1}^{n} \mathbb{E}\left(\exp\left(\frac{t}{n}(X_k - \mu)\right)\right)$$

   b) En déduire que, pour tout $t \geqslant 0$, $\mathbb{P}(\overline{X}_n - \mu \geqslant \varepsilon) \leqslant \exp\left(-t\varepsilon + \frac{t^2}{8n}\right)$, puis en choisissant convenablement t que

   $$\mathbb{P}(\overline{X}_n - \mu \geqslant \varepsilon) \leqslant \exp\left(-2n\varepsilon^2\right)$$

6. Soit $\varepsilon \geqslant 0$. En considérant les variables $1 - X_1, ..., 1 - X_n$, montrer que

   $$\mathbb{P}(\overline{X}_n - \mu \leqslant -\varepsilon) \leqslant \exp\left(-2n\varepsilon^2\right)$$

7. Soit $\alpha \in]0,1[$. Si le paramètre μ est inconnu et $X_1, ..., X_n$ suivent la même loi, montrer que $\left[\overline{X}_n - \sqrt{\frac{\ln(\frac{2}{\alpha})}{2n}}, \overline{X}_n + \sqrt{\frac{\ln(\frac{2}{\alpha})}{2n}}\right]$ est un intervalle de confiance aléatoire pour l'estimation de μ au niveau de confiance $1 - \alpha$.

Description du modèle

On modélise le problème, évoqué dans le préambule, de la manière suivante :

n et r sont des entiers naturels plus grands que 2 et $r < n$.

Les actions sont identifiées aux entiers de 1 à r et n actions sont réalisées l'une après l'autre et numérotées de 1 à n.

$p_1, ..., p_r$ sont des réels appartenant à $]0,1[$ non tous égaux et on note p^* leur maximum.

On définit, pour tout le sujet, les variables aléatoires suivantes :

- **Le choix des actions :** $(I_j)_{j \in [\![1,n]\!]}$ à valeurs dans $[\![1,r]\!]$, I_j est égale à la j-ème action réalisée.
- **Les récompenses aléatoires proposées :** $(X_{i,j})_{i \in [\![1,r]\!], j \in [\![1,n]\!]}$ sont des variables de Bernoulli indépendantes telles que pour tout $i \in [\![1,r]\!], j \in [\![1,n]\!]$, $X_{i,j}$ est de paramètre p_i. $X_{i,j}$ est la récompense aléatoire pour l'action i lorsque celle-ci est effectuée pour la j-ième fois.

  On suppose que les $(X_{i,j})_{i \in [\![1,r]\!], j \in [\![1,n]\!]}$ sont indépendantes des $(I_j)_{j \in [\![1,n]\!]}$.
- Y_j est la variable aléatoire égale à la récompense pour la j-ème action réalisée, $j \in [\![1,n]\!]$.
- Par exemple, si pour un certain $\omega \in \Omega$, les quatre premières actions réalisées sont 2, 4, 3 et 2, alors on a :

  $I_1(\omega) = 2$, $I_2(\omega) = 4$, $I_3(\omega) = 3$, $I_4(\omega) = 2$, $Y_1(\omega) = X_{2,1}(\omega)$, $Y_2(\omega) = X_{4,1}(\omega)$, $Y_3(\omega) = X_{3,1}(\omega)$, $Y_4(\omega) = X_{2,2}(\omega)$.

- Pour tous $i \in [\![1,r]\!]$, $j \in [\![1,n]\!]$ et $\omega \in \Omega$, $N_{i,j}(\omega)$ est égal au nombre d'indices $k \in [\![1,j]\!]$ tels que $I_k(\omega) = i$.
- Pour $i \in [\![1,r]\!]$, $j \in [\![1,n]\!]$ et $\omega \in \Omega$, $\overline{X}_{i,j}(\omega) = \dfrac{1}{j}\sum_{k=1}^{j} X_{i,k}(\omega)$, et

$$\hat{X}_{i,j}(\omega) = \begin{cases} \dfrac{1}{N_{i,j}(\omega)} \displaystyle\sum_{k=1}^{N_{i,j}(\omega)} X_{i,k}(\omega) & \text{si } N_{i,j}(\omega) \neq 0 \\ 0 & \text{sinon.} \end{cases}$$

L'objectif est de maximiser, en moyenne, la récompense totale en définissant les variables I_j pour ce faire. Ces définitions constituent la définition d'une stratégie.

Utilisation d'une table SQL

On a réalisé l'expérience décrite dans le modèle ci-dessus et enregistré les résultats obtenus dans une table `Bandit`.

Précisément cette table est composé de n enregistrements, $n \geqslant 1000$. Elle comporte trois champs, `Numero` , `Action` et `Recompense` qui sont de type entier.

Si par exemple la dixième action réalisée est l'action 2 générant une récompense égale à 1, ceci est représenté par la ligne (10, 2, 1) dans la table.

Cette table est associée à un élément ω de Ω.

8. a) Quelle requête renvoie $I_{100}(\omega)$ lorsqu'on l'exécute ?
   b) Quelle requête renvoie la colonne des numéros des actions où on a réalisé l'action 2 et obtenu une récompense non nulle ?
   c) Ecrire une requête qui renvoie $N_{1,100}(\omega)$.
   d) Ecrire une requête qui renvoie $\hat{X}_{2,n}(\omega)$. Si l'on a $N_{2,n}(\omega) \geqslant 100$, de quel paramètre $\hat{X}_{2,n}(\omega)$ est-elle une bonne estimation ?

La stratégie ETC [1] et une majoration de son regret moyen

On note R_n la récompense totale après que les n actions sont exécutées, $R_n = \sum_{j=1}^{n} Y_j$.

9. Soit $j \in [\![1,n]\!]$.
   a) Montrer que, $\mathbb{E}(Y_j) = \sum_{i=1}^{r} \mathbb{P}([Y_j = 1] \cap [I_j = i])$.
   b) Etablir que $\mathbb{P}([Y_j = 1] \cap [I_j = i]) = \sum_{k=1}^{j} \mathbb{P}([X_{i,k} = 1] \cap [I_j = i] \cap [N_{i,j} = k])$.
   c) En déduire que $\mathbb{E}(Y_j) = \sum_{i=1}^{r} p_i \mathbb{P}(I_j = i)$ puis que $\mathbb{E}(Y_j) \leqslant p^*$.

▶ On définit **le regret** $\Delta_n = np^* - R_n$ et on pose pour tout $i \in [\![1,r]\!]$, $\delta_i = p^* - p_i$.

10. a) Soit $i \in [\![1,r]\!]$. Montrer que $\mathbb{E}(N_{i,n}) = \sum_{j=1}^{n} \mathbb{P}(I_j = i)$.

1. Explore then commit

b) En déduire que $\mathbb{E}(\Delta_n) = \sum_{i=1}^{r} \delta_i \mathbb{E}(N_{i,n})$.

11. **Stratégie "No Strategy"** - On suppose dans cette question que pour tout $j \in [\![1, n]\!]$, I_j suit la loi uniforme sur $[\![1, r]\!]$. Calculer $\mathbb{E}(\Delta_n)$ en fonction de n, r et des δ_i.

▶ **Stratégie "ETC"** - On suppose que m est un entier plus grand que 2 tel que $rm < n$.

On définit une nouvelle variable aléatoire Z_m par, pour tout $\omega \in \Omega$:
$Z_m(\omega)$ est égal au plus petit indice $k \in [\![1, r]\!]$ pour lequel on a $\hat{X}_{k,mr}(\omega) = \max_{1 \leqslant i \leqslant r} \hat{X}_{i,mr}(\omega)$.

La stratégie ETC consiste à définir les I_j comme suit pour tout $j \in [\![1, n]\!]$ et pour tout $\omega \in \Omega$:

- si $j \in [\![1, m]\!]$ alors $I_j(\omega) = 1$, si $j \in [\![m+1, 2m]\!]$ alors $I_j(\omega) = 2, \ldots$, si $j \in [\![(k-1)m+1, km]\!]$ alors $I_j(\omega) = k, \ldots$, si $j \in [\![(r-1)m+1, rm]\!]$ alors $I_j(\omega) = r$,
- si $mr < j \leqslant n$, $I_j(\omega) = Z_m(\omega)$.

On suppose dans la suite de cette partie que l'on applique la stratégie ETC.

12. Dans cette question $n = 10$, $r = 3$, $m = 2$, on réalise les 10 actions en appliquant la stratégie ETC et on obtient le tableau suivant :

j	1	2	3	4	5	6	7	8	9	10
$I_j(\omega)$	1	1	2	2	3	3	...	...	...	...
$Y_j(\omega)$	0	1	1	1	0	0	1	0	1	1

Préciser alors les valeurs de $\hat{X}_{1,6}(\omega)$, $\hat{X}_{2,6}(\omega)$, $\hat{X}_{3,6}(\omega)$ et $Z_2(\omega)$. En déduire par quelle(s) valeur(s) il faut compléter la deuxième ligne du tableau.

13. Écrire une fonction Python `Z(m,r,Rec)` qui renvoie la valeur de Z_m obtenue lorsque la variable `Rec` est le vecteur des récompenses obtenues pour les rm premières actions suivant la stratégie ETC.

14. Soit $i \in [\![1, r]\!]$.

a) Que vaut $N_{i,mr}$? En déduire que, $\hat{X}_{i,mr} = \overline{X}_{i,m}$.

b) En déduire que pour tout $\varepsilon \geqslant 0$,

$$\mathbb{P}(\hat{X}_{i,mr} - p_i \geqslant \varepsilon) \leqslant \exp(-2m\varepsilon^2) \text{ et } \mathbb{P}(\hat{X}_{i,mr} - p_i \leqslant -\varepsilon) \leqslant \exp(-2m\varepsilon^2)$$

15. On note s un élément de $[\![1, r]\!]$ tel que $p_s = p^*$. Soit $i \in [\![1, r]\!]$.

a) Montrer que $\mathbb{E}(N_{i,n}) = m + (n - mr)\mathbb{P}(Z_m = i)$.
En déduire que : $\mathbb{E}(N_{i,n}) \leqslant m + (n - mr)\mathbb{P}\left(\hat{X}_{i,mr} \geqslant \hat{X}_{s,mr}\right)$

b) Montrer que $\mathbb{P}\left(\hat{X}_{i,mr} \geqslant \hat{X}_{s,mr}\right) \leqslant \mathbb{P}\left(\hat{X}_{i,mr} - p_i \geqslant \frac{\delta_i}{2}\right) + \mathbb{P}\left(p_s - \hat{X}_{s,mr} \geqslant \frac{\delta_i}{2}\right)$.

c) En conclure que :

$$\mathbb{E}(N_{i,n}) \leqslant m + 2(n - mr)\exp\left(-m\frac{\delta_i^2}{2}\right) \leqslant m + 2n\exp\left(-m\frac{\delta_i^2}{2}\right)$$

16. a) Établir que :

$$\mathbb{E}(\Delta_n) \leqslant m\sum_{i=1}^{r} \delta_i + 2n\sum_{i=1}^{r} \delta_i \exp\left(-m\frac{\delta_i^2}{2}\right)$$

▷ On pose $\alpha = \sum_{i=1}^{r} \delta_i$ et $\beta = \sum_{1 \leqslant i \leqslant r, \delta_i \neq 0} \frac{1}{\delta_i}$.

b) Montrer que pout tout $x > 0$, $\mathrm{e}^{-x} \leqslant \frac{1}{x\,\mathrm{e}}$. En déduire que $\mathrm{E}(\Delta_n) \leqslant m\alpha + 2n\frac{\beta}{m}$.

c) En choisissant $m = \left\lfloor \sqrt{\frac{2n\beta}{\alpha}} \right\rfloor + 1$, montrer que pour n assez grand, $m \geqslant 2$, $mr < n$ et que

$$\mathrm{E}(\Delta_n) \leqslant \sqrt{8\alpha\beta}\sqrt{n} + \alpha$$

La stratégie UCB [2] et une majoration de son regret

On conserve les notations de la partie précédente mais on va redéfinir les variables I_j.

17. Soit $\theta \in]0,1[$, $j \in [\![1,n]\!]$ et $i \in [\![1,r]\!]$, on pose $\gamma = \sqrt{\frac{-\ln(\theta)}{2j}}$.

Montrer que $\mathrm{P}(\overline{X}_{i,j} + \gamma \leqslant p_i) \leqslant \theta$.

▶ La stratégie UCB consiste à définir les I_j comme suit pour tout $j \in [\![1,n]\!]$ et pour tout $\omega \in \Omega$:

- si $j \in [\![1,r]\!]$ alors $I_j(\omega) = j$;
- si $j \in [\![r,n-1]\!]$, posons pour tout $i \in [\![1,r]\!]$, $U_{i,j}(\omega) = \hat{X}_{i,j}(\omega) + \sqrt{\frac{\ln(n)}{N_{i,j}(\omega)}}$.

  $I_{j+1}(\omega)$ est égal au plus petit indice $i \in [\![1,r]\!]$ pour lequel on a $U_{i,j}(\omega) = \max_{1 \leqslant k \leqslant r} U_{k,j}(\omega)$.
- On note aussi pour $i \in [\![1,r]\!]$ et $j \in [\![1,n-1]\!]$, $V_{i,j}(\omega) = \overline{X}_{i,j}(\omega) + \sqrt{\frac{\ln(n)}{j}}$.

On suppose dans la suite de cette partie que l'on applique la stratégie UCB.

18. a) Justifier que pour $i \in [\![1,r]\!]$ on a $N_{i,r} = 1$ et $\hat{X}_{i,r} = X_{i,1}$.

b) Ecrire une fonction Python `Bernoulli(p)` qui renvoie la valeur d'une simulation d'une variable aléatoire suivant la loi de Bernoulli de paramètre p.

c) En supposant que `hatX` contient $\hat{X}_{i,j}$, `N` contient $N_{i,j}$ et le vecteur `P` contient les valeurs de $p_1, \ldots, p_r$, écrire une fonction Python `maj(hatX,N,i,P)` qui simule la récompense de l'action i, celle-ci étant $(j+1)$-ème action réalisée et renvoie la valeur de $\hat{X}_{i,j+1}$ dans ces conditions.

19. Compléter l'écriture de la fonction Python `Actions(P,n)` ci-dessous pour qu'elle renvoie la simulation d'un vecteur des n actions réalisées par l'algorithme UCB si le vecteur `P` contient les valeurs de $p_1, \ldots, p_r$.

```
def Actions(P,n):
    r=np.shape(P)[0]; I=np.zeros(n)
    I[0:r]=[k+1 for k in range(r)];N=np.ones(r)
    hatX=np.array([Bernoulli(P[i]) for i in range(r)])
    for j in range(.....):
        Max=hatX[0]+np.sqrt(np.log(n)/N[0])
        I[j]=1
```

2. Upper confidence bound

```
    for k in range(1,r):
      val=hatX[k]+np.sqrt(np.log(n)/N[k])
      if val>Max:
          Max=...
          I[j]=...
    hatX[I[j]-1]=maj(hatX[I[j]-1],N[I[j]-1],I[j],P)
    N[I[j]-1]=...
  return I
```

▶ Dans la suite de l'énoncé, on considère $s \in [\![1,r]\!]$ tel que $p^* = p_s$, $i \in [\![1,r]\!]$ tel que $p_i < p^*$ et $u \in [\![1,n-1]\!]$. On note $A_{i,u}$ l'événement $\left[\min_{j\in[\![r,n-1]\!]} U_{s,j} \leqslant p_s\right] \cup [V_{i,u} \geqslant p_s]$.

20. *Majoration de* $\mathbb{E}(N_{i,n})$

a) Montrer que si $[N_{i,n} > u]$ est réalisé alors il existe $k \in [\![r, n-1]\!]$ tel que $[N_{i,k} = u] \cap [U_{i,k} \geqslant U_{s,k}]$ est réalisé, puis que $\left[V_{i,u} \geqslant \min_{j\in[\![r,n-1]\!]} U_{s,j}\right]$ l'est.

b) Montrer que si $[N_{i,n} > u]$ est réalisé alors $A_{i,u}$ l'est. En déduire que

$$\mathbb{E}(N_{i,n}) \leqslant u + \mathbb{P}(A_{i,u})n$$

21. *Majoration de* $\mathbb{P}(A_{i,u})$

a) Montrer en utilisant la question 17 que :

$$\mathbb{P}\left(\min_{j\in[\![r,n-1]\!]} U_{s,j} \leqslant p_s\right) \leqslant \mathbb{P}\left(\min_{k\in[\![1,n-1]\!]} V_{s,k} \leqslant p_s\right) \leqslant \frac{1}{n}$$

b) Etablir que si $\delta_i - \sqrt{\dfrac{\ln(n)}{u}} \geqslant 0$, $\mathbb{P}(V_{i,u} \geqslant p_s) \leqslant \exp\left(-2u\left(\delta_i - \sqrt{\dfrac{\ln(n)}{u}}\right)^2\right)$.

▶ On suppose dans la suite que n est assez grand pour que $\dfrac{4\ln(n)}{\delta_i^2} \in [\![1, n-2]\!]$. On choisit $\left\lfloor \dfrac{4\ln(n)}{\delta_i^2} \right\rfloor + 1$ comme valeur de u pour la suite de cette question.

c) Montrer que la fonction $\varphi : t \mapsto \exp\left(-2(\delta_i\sqrt{t} - \sqrt{\ln(n)})^2\right)$ est décroissante sur $\left[\dfrac{\ln(n)}{\delta_i^2}, +\infty\right[$.

d) En déduire que $\mathbb{P}(V_{i,u} \geqslant p_s) \leqslant \dfrac{1}{n^2}$ puis que $\mathbb{P}(A_{i,u}) \leqslant \dfrac{2}{n}$.

22. Etablir que $\mathbb{E}(N_{i,n}) \leqslant \dfrac{4\ln(n)}{\delta_i^2} + 3$ puis que $\mathbb{E}(\Delta_n) \leqslant 4\beta\ln(n) + 3\alpha$.

Tourner la page SVP

Aide-mémoire

Toutes les fonctions et instructions présentées ne sont pas utiles et il est possible d'utiliser d'autres fonctions ou instructions absentes de cet aide-mémoire.

SQL

`SELECT col1, col2, ... FROM table ...`
Renvoie un tableau formé des colonnes nommées `col1, col2,...` de la table nommée `table`. On peut utiliser * pour sélectionner toutes les colonnes et aussi la clause `WHERE`, en fin de requête, pour ne sélectionner que certains enregistrements.

Si `col` est une colonne de la table `table` :

`SELECT MAX(col) FROM table ...`
Retourne la valeur maximale du champ `col` pour les enregistrements sélectionnés dans `table`. La fonction peut s'appliquer à des données numériques ou alphanumériques.

`SELECT MIN(col) FROM table ...`
Retourne la valeur minimale du champ `col` pour les enregistrements sélectionnés dans `table`. La fonction peut s'appliquer à des données numériques ou alphanumériques.

`SELECT SUM(col) FROM table ...`
Retourne la somme des valeurs du champ `col` pour les enregistrements sélectionnés dans `table`.

`SELECT AVG(col) FROM table ...`
Retourne la moyenne des valeurs du champ `col` pour les enregistrements sélectionnés dans `table`.

`SELECT COUNT(col) FROM table ...`
Retourne le nombre d'éléments du champ `col` pour les enregistrements sélectionnés dans `table`. La fonction peut s'appliquer à des données numériques ou alphanumériques.

`WHERE` - Commande qui dans une requête permet de sélectionner les enregistrements d'une table qui respectent une condition.

`AND, OR, NOT` - Opérateurs logiques pour la clause `WHERE`.

Module mathématique numpy de Python

```
import numpy as np
```

`np.array(L)`	Transforme la liste `L` en vecteur ou matrice numpy
`np.zeros([n,m])`	Crée la matrice nulle de taille $n \times m$
`np.zeros(n)`	Crée le vecteur nul de taille n
`np.ones([n,m])`	Crée la matrice de taille $n \times m$ dont tous les coefficients valent 1
`np.ones(n)`	Crée le vecteur de taille n dont tous les coefficients valent 1
`np.max(M)`	Renvoie le plus grand élément de M, matrice ou vecteur
`np.min(M)`	Renvoie le plus petit élément de M, matrice ou vecteur
`np.shape(M)`	Renvoie dans un couple le format de la matrice M
`np.mean(M)`	Renvoie la moyenne des éléments de M
`np.sqrt(x)`	Renvoie $\sqrt{x}$ si $x \geq 0$
`np.log(x)`	Renvoie $\ln(x)$ si $x > 0$

Sous module random de numpy pour la simulation probabiliste

```
import numpy.random as rd
```

`rd.random([r,s])`	Simule une réalisation d'une matrice (r, s) dont les coefficients sont des variables aléatoires indépendantes qui suivent la loi uniforme $\mathcal{U}([0,1])$

Si le paramètre `[r,s]` est remplacé par `r`, cette fonction renvoie une réalisation d'un vecteur de longueur r correspondant à la loi en question, et si ce paramètre est omis, elles renvoient un seul coefficient suivant les mêmes contraintes.

Fin