# Devoir de Mathématiques

Pour le 11 mars 2017

---

## ❖ Problème ❖

**1.** Pour tout entier $n \geqslant 1$, on pose

$$a_n = \frac{\sqrt{n}}{4^n}\binom{2n}{n}.$$

**1. a.** Calculer la limite de $a_n$ lorsque $n$ tend vers $+\infty$.
**1. b.** En déduire un équivalent de

$$A_N = \sum_{n=1}^{N} \frac{a_n^2}{n}$$

lorsque $N$ tend vers $+\infty$.

À l'instant $n = 0$, un kangourou (mathématique) se trouve en $(0, 0)$.

Si le kangourou est au point $(i, j) \in \mathbb{Z}^2$ à l'instant $n \in \mathbb{N}$, il saute pour se trouver vers l'un des quatre points

$$(i-1, j-1), \quad (i-1, j+1), \quad (i+1, j+1), \quad (i+1, j-1)$$

à l'instant $(n+1)$. Ces quatre destinations sont équiprobables et chaque saut est indépendant du précédent.

Les coordonnées du kangourou à l'instant $n \in \mathbb{N}$ sont notées

$$(X_n, Y_n) \in \mathbb{Z}^2.$$

On note $R_n$, le nombre de fois que le kangourou repasse par l'origine entre les instants 1 et $2n$.

**2.** Proposer une modélisation *mathématiquement correcte* des déplacements du kangourou.

**3.** En utilisant ce modèle probabiliste :

**3. a.** Analyser les événements $[X_k = 0]$ et $[Y_k = 0]$ et en déduire que

$$\forall\, i \in \mathbb{N}, \quad \mathbf{P}(X_{2i+1} = 0) = \mathbf{P}(Y_{2i+1} = 0) = 0$$

et que

$$\forall\, i \in \mathbb{N}^*, \quad \mathbf{P}(X_{2i} = 0) = \mathbf{P}(Y_{2i} = 0) = \frac{a_i}{\sqrt{i}}.$$

**3. b.** Exprimer $R_n$ comme une variable aléatoire, démontrer que cette variable aléatoire est d'espérance finie et que

$$\mathbf{E}(R_n) = \sum_{k=1}^{n} \frac{a_k^2}{k}.$$

**3. c.** En déduire un équivalent de $\mathbf{E}(R_n)$ lorsque $n$ tend vers $+\infty$.

**3. d.** Proposer une simulation informatique des déplacements du kangourou qui permette de vérifier l'équivalent trouvé.

## Solution ❀ Retours à l'origine

**1. a.** D'après la formule de Stirling,

$$n! \sim \sqrt{2\pi n}e^{-n}n^n.$$

Donc

$$a_n \xrightarrow[n\to+\infty]{} \frac{1}{\sqrt{\pi}}.$$

**1. b.** Par conséquent

$$\frac{a_n^2}{n} \sim \frac{1}{\pi n}$$

lorsque $n$ tend vers $+\infty$. Or la série

$$\sum {}^1\!/_{\pi\, n}$$

est une série divergente de terme général positif. Par conséquent, la série

$$\sum \frac{a_n^2}{n}$$

est divergente et ses sommes partielles $A_N$,qui tendent vers $+\infty$, sont équivalentes aux sommes partielles

$$B_N = \sum_{n=1}^{N} \frac{1}{\pi n} = \frac{1}{\pi}\sum_{n=1}^{N}\frac{1}{n}$$

et donc à ${}^1\!/_\pi \ln N$ lorsque $N$ tend vers $+\infty$.

**2.** On considère un espace probabilisé $(\Omega, \mathcal{A}, \mathbf{P})$ sur lequel est définie une suite $(\Delta_n)_{n\geqslant 1}$ de variables aléatoires indépendantes et de même loi que la variable $\Delta$, caractérisée par

$$\begin{aligned}\mathbf{P}\big(\Delta = (1,1)\big) &= \mathbf{P}\big(\Delta = (-1,1)\big)\\ &= \mathbf{P}\big(\Delta = (1,-1)\big)\\ &= \mathbf{P}\big(\Delta = (-1,-1)\big) = \frac{1}{4}.\end{aligned}$$

Ces variables traduisent les hypothèses d'indépendance et d'équiprobabilité des sauts.

Définissons deux fonctions $H_n$ et $V_n$ de $\Omega$ dans $\{\pm 1\}$ en posant

$$\forall\, n \in \mathbb{N}^*,\ \forall\, \omega \in \Omega, \quad \Delta_n(\omega) = \big(H_n(\omega), V_n(\omega)\big).$$

On a alors

$$\begin{array}{rcll}[H_n = 1] &=& [\Delta_n = (1,1)] \sqcup [\Delta_n = (1,-1)] & \in \mathcal{A}\\ [H_n = -1] &=& [\Delta_n = (-1,1)] \sqcup [\Delta_n = (-1,-1)] & \in \mathcal{A}\\ [V_n = 1] &=& [\Delta_n = (1,1)] \sqcup [\Delta_n = (-1,1)] & \in \mathcal{A}\\ [V_n = -1] &=& [\Delta_n = (1,-1)] \sqcup [\Delta_n = (-1,-1)] & \in \mathcal{A}\end{array}$$

donc $H_n$ et $V_n$ sont des variables aléatoires discrètes, de même loi :

$$\mathbf{P}(H_n = 1) = \mathbf{P}(V_n = 1) = {}^1\!/_2,$$
$$\mathbf{P}(H_n = -1) = \mathbf{P}(V_n = -1) = {}^1\!/_2$$

et indépendantes : quels que soient $h$ et $v$ dans $\{\pm 1\}$,

$$[H_n = h] \cap [V_n = v] = [\Delta_n = (h,v)]$$

donc

$$\mathbf{P}(H_n = h, V_n = v) = \frac{1}{4} = \mathbf{P}(H_n = h)\,\mathbf{P}(V_n = v).$$

On en déduit que

$$(H_1, V_1, H_2, V_2, \dots, H_n, V_n, \dots)$$

est une famille de variables aléatoires indépendantes et de même loi : en effet, quel que soit $n \geqslant 1$, quels que soient les couples $(h_k, v_k)$ dans $\{\pm 1\}^2$,

$$\bigcap_{k=1}^{n}[H_k = h_k, V_k = v_k] = \bigcap_{k=1}^{n}[\Delta_k = (h_k, v_k)]$$

donc

$$\begin{aligned}\mathbf{P}\Big(\bigcap_{k=1}^{n}[H_k = h_k, V_k = v_k]\Big)\\ &= \prod_{k=1}^{n}\mathbf{P}\big(\Delta_k = (h_k, v_k)\big)\\ &= \prod_{k=1}^{n}\mathbf{P}(H_k = h_k)\,\mathbf{P}(V_k = v_k).\end{aligned}$$

On pose alors $(X_0, Y_0) = (0,0)$ et

$$\forall\, n \in \mathbb{N}, \quad (X_{n+1}, Y_{n+1}) = (X_n, Y_n) + \Delta_{n+1} = \sum_{k=1}^{n+1}\Delta_k$$

c'est-à-dire

$$X_n = \sum_{k=1}^{n} H_k \quad \text{et} \quad Y_n = \sum_{k=1}^{n} V_k.$$

En tant que sommes (finies) de variables aléatoires discrètes, $X_n$ et $Y_n$ sont bien des variables aléatoires discrètes et d'après le lemme des coalitions, $X_n$ (fonction déterministe des variables $H_1, \dots, H_n$) et $Y_n$ (fonction déterministe des variables $V_1, \dots, V_n$) sont indépendantes.

**3. a.** Comme $X_k(\omega)$ est une somme d'entiers qui sont égaux à 1 ou à $-1$, l'entier $X_k(\omega)$ est nul si, et seulement si, il y a autant de termes égaux à 1 que de termes égaux à $-1$ dans cette somme.

Autrement dit, notre kangourou est revenu à son point de départ si, et seulement si, il a effectué autant de sauts vers la gauche que de sauts vers la droite.

❧ En particulier, pour tout entier *impair* $k$,

$$[X_k = 0] = \varnothing$$

donc

$$\forall\, i \in \mathbb{N}, \quad \mathbf{P}(X_{2i+1} = 0) = 0.$$

❧ Pour tout entier *pair* $k = 2i$, la somme $X_{2i}(\omega)$ est nulle si, et seulement si, il existe une partie $A \subset \{1, \dots, 2i\}$ constituée de $i$ indices exactement telle que

$$\forall\, k \in A, \quad X_k(\omega) = 1 \quad \text{et} \quad \forall\, k \notin A, \quad X_k(\omega) = -1$$

c'est-à-dire

$$[X_{2i} = 0] = \bigsqcup_{A \in M_i}\Big[\bigcap_{k\in A}[H_k = 1]\Big] \cap \Big[\bigcap_{k\in A^c}[H_k = -1]\Big]$$

où $M_i$ est l'ensemble des parties à $i$ éléments de l'ensemble $\{1, \dots, 2i\}$.

Comme les variables aléatoires $H_i$ sont indépendantes, on en déduit que

$$\begin{aligned}\mathbf{P}(X_{2i} = 0) &= \sum_{A \in M_i} \prod_{k \in A} \mathbf{P}(H_k = 1) \prod_{k \in A^c} \mathbf{P}(H_k = -1) \\ &= \left(\frac{1}{2}\right)^{2i} \#(M_i) \\ &= \binom{2i}{i} \frac{1}{4^i} = \frac{a_i}{\sqrt{i}}.\end{aligned}$$

❧ Les vecteurs aléatoires

$$(H_1, \dots, H_k) \quad \text{et} \quad (V_1, \dots, V_k)$$

ont mêmes lois conjointes (ce sont des familles de variables aléatoires indépendantes qui suivent toutes la même loi). Comme $X_k$ et $Y_k$ se déduisent de ces deux vecteurs aléatoires par la même fonction déterministe :

$$[(x_1, \dots, x_k) \mapsto x_1 + \dots + x_k]$$

les variables $X_k$ et $Y_k$ ont même loi. En particulier,

$$\forall\, k \geqslant 1, \quad \mathbf{P}(X_k = 0) = \mathbf{P}(Y_k = 0).$$

**3. b.** Le kangourou passe par l'origine à l'instant $k$ si, et seulement si, $X_k = Y_k = 0$, donc le nombre de passages par l'origine entre l'instant 1 et l'instant $2n$ est défini par

$$R_n = \sum_{k=1}^{2n} \mathbb{1}_{[X_k=0, Y_k=0]} = \sum_{i=1}^{n} \mathbb{1}_{[X_{2i}=0, Y_{2i}=0]}$$

puisque $[X_k = 0, Y_k = 0] = \varnothing$ pour tout $k$ *impair*.

En tant que somme finie de variables aléatoires de Bernoulli, $R_n$ est une variable aléatoire discrète bornée (elle ne prend qu'un nombre fini de valeurs), donc d'espérance finie.

❧ Par linéarité de l'espérance,

$$\begin{aligned}\mathbf{E}(R_n) &= \sum_{i=1}^{n} \mathbf{P}(X_{2i} = 0, Y_{2i} = 0) \\ &= \sum_{i=1}^{n} \mathbf{P}(X_{2i} = 0)\, \mathbf{P}(Y_{2i} = 0) = \sum_{i=1}^{n} \frac{a_i^2}{i}\end{aligned}$$

puisque $X_{2i}$ et $Y_{2i}$ sont indépendantes.

**3. c.** On déduit de **1.b.** que

$$\mathbf{E}(R_n) \sim \frac{\ell n\, n}{\pi}$$

lorsque $n$ tend vers $+\infty$.

**3. d.** Une manière élémentaire consiste à simuler successivement les vecteurs aléatoires $\Delta_k$ à partir d'un échantillon de deux variables aléatoires de Bernoulli indépendantes : si $H_k = 2B_k - 1$, alors

$$H_k = 1 \iff B_k = 1 \quad \text{et} \quad H_k = -1 \iff B_k = 0.$$

On construit ensuite la trajectoire par récurrence.

```
import numpy as np
from numpy.random import binomial as Bernoulli

def Delta():
  # Δ_{k+1} = (H_{k+1}, V_{k+1})
  return 2*Bernoulli(1, 0.5, size=2)-1

def Trajectoire(n):
  T = [np.zeros(2)]     # (X_0, Y_0) = (0, 0)
  for i in range(n):
    # (X_{k+1}, Y_{k+1}) = (X_k, Y_k) + Δ_{k+1}
    T.append(T[-1]+Delta())
  return np.array(T)
```

❧ Il est plus efficace de simuler en une seule fois toutes les variables

$$\big((H_k, V_k)\big)_{1 \leqslant k \leqslant n} \in \mathfrak{M}_{n,2}(\mathbb{Z})$$

et d'appliquer la fonction `cumsum` qui calcule la famille des sommes partielles sur chacune des deux colonnes (avec `axis=0`, on somme en faisant varier l'indice de rang 0, c'est-à-dire l'indice des lignes).

```
def Trajectoire(n):
  Deltas = 2*Bernoulli(1, 0.5, size=(n,2))-1
  Deltas[0] = [0,0]
  T = np.cumsum(Deltas, axis=0)
  return T
```

Avant de calculer la liste des positions successives, il faut annuler le premier déplacement pour que la trajectoire commence bien en $(X_0, Y_0) = (0, 0)$.

❧ Pour chaque trajectoire `T`, on compte le nombre de retours à l'origine en comparant les positions de rang pair à $(0, 0)$ à partir du rang 2 (les positions de rang impair sont nécessairement différentes de $(0, 0)$ et le point de départ ne compte pas comme un retour à l'origine).

```
def NbRetours(T):
  Retour = [(t[0]==0 and t[1]==0)
            for t in T[2::2]]
  return sum(Retour)
```

Par convention, les booléens `True` et `False` sont respectivement assimilés aux entiers 1 et 0 : compter le nombre de `True` revient donc à calculer la somme de tous les booléens.

❧ On peut alors calculer des valeurs approchées de $\mathbf{E}(R_n)$ de trois manières différentes.

On peut estimer $\mathbf{E}(R_n)$ en simulant un assez grand nombre $N$ de fois (par exemple $N = 10^4$, pour que la Loi des grands nombres ait une chance de s'appliquer) les trajectoires

$$\big((X_k, Y_k)\big)_{0 \leqslant k \leqslant 2n}.$$

On compte le nombre $R_n$ de retours à l'origine pour chaque trajectoire et on calcule la moyenne (empirique) de ce nombre.

```
def simEsperance(n, N=10**4):
  s = 0
  for i in range(N):
    T = Trajectoire(2*n)
    s += NbRetours(T)
  return s/N
```

On a calculé un équivalent simple de $\mathbf{E}(R_n)$.

```
def equivEsperance(n):
  return np.log(n)/np.pi
```

Enfin, on connaît une expression exacte de $\mathbf{E}(R_n)$ sous forme d'une somme.

```
from scipy.special import binom as c_bin

def esperance(n):
  tg = [(0.25**k*c_bin(2*k,k))**2
        for k in range(1,n+1)]
  return np.sum(tg)
```

Cependant, la fonction `scipy.special.binom` donne un résultat infini ou indéterminé (nan) à partir de $k = 520$ ! Il vaut donc mieux calculer les coefficients binomiaux à la main.

```
def esperance(n):
  return np.sum([(np.prod([0.25*(k+i)/i
                  for i in range(1,k+1)]))**2
         for k in range(1, n+1)])
```

En pratique, pour $n = 100$, l'espérance $\mathbf{E}(R_n)$ est à peu près égale à $1,534\,527$ ; l'équivalent est à peu près égal à $1,69$ et une dizaine de simulations réalisées avec $N = 10^4$ donnent des moyennes empiriques comprises entre $1,51$ et $1,57$.

Ces simulations sont en accord avec la valeur exacte de l'espérance, mais pas avec l'équivalent trouvé au **1.b.** car $n = 100$ est trop petit pour être considéré comme tendant vers l'infini.

REMARQUE.— Si on s'intéresse seulement à $\mathbf{E}(R_n)$ (et pas aux trajectoires du kangourou), le code suivant fournit une estimation de l'espérance avec une complexité spatiale moindre (on ne mémorise pas les trajectoires).

```
dZ = [-1, 1]  # les deux déplacements possibles

def saut():
  # On choisit un déplacement au hasard
  i = Bernoulli(1, 0.5)
  return dZ[i]

def R(n):
  X, Y, cptr = 0, 0, 0
  for k in range(2*n):
    X += saut()
    Y += saut()
    # Après chaque saut, on vérifie si Skippy
    # est de retour à l'origine.
    if (X==0) and (Y==0):
      cptr += 1
  return cptr

def simul(n, N=1000):
  # Nombre moyen de retours à l'origine
  # sur N simulations.
  s = 0.
  for k in range(N):
    s += R(n)
  return s/N
```