Math-info. Algèbre. Corrigé

On charge d'abord les modules dont on a besoin (en prenant les mêmes alias que dans les documents).

```
from math import *
from scipy import *
import numpy as np
import matplotlib.pyplot as plt
import scipy.integrate as integr
import scipy.optimize as resol
import numpy.random as rd
from numpy.polynomial import *
```

Exercice A

1)
```
def tournoi(n) :
    A = np.zeros((n,n))
    for i in range(n) :
        for j in range(i+1,n) :
            A[i,j] = rd.randint(0,2) ; A[j,i] = - A[i,j]
    return A
```

Exemple : `print(tournoi(2))` affiche $\begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$ ou $\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$

2)
```
for n in range(5,9) :
    print( [ alg.det(tournoi(n)) for i in range(10)] )
```

Remarque : Pour afficher des entiers (et non des flottants), utiliser : `int(round(alg.det(tournoi(n))))`.

En effet, `round(x)=round(x,0)` renvoie l'entier le plus proche (écrit en flottant, par exemple `2.0`).

3) a) $A^T = -A$, donc $\det A = (-1)^n \det A$, donc $\det A = 0$ si n impair.

4) a) On a $\det(J_n - I_n) = (n-1)(-1)^{n-1}$ car J_n semblable à $\mathrm{Diag}(n, 0, 0, ..., 0)$.

b) Dans $\mathbb{Z}$, on note $x \equiv y$ [2] ssi $x - y$ est pair. Si $x \equiv y$ [2] et $x' \equiv y'$ [2], alors $x + x' \equiv y + y'$ [2] et $xx' \equiv yy'$ [2].

Par récurrence en développant le déterminant selon la première colonne et en raisonnant modulo 2 :

On a $\det(A) = \sum_{i=1}^n a_{ij} C_{ij} = \sum_{i=1}^n b_{ij} C'_{ij}$ [2], car $a_{ij} \equiv b_{ij}$ [2] et par récurrence $C_{ij} \equiv C'_{ij}$ [2].

Remarque : Noter au passage que le déterminant d'une matrice à coefficients entiers est un entier.

c) Résulte de a) et b) : Par b), $\det A$ a même parité que $\det(J_n - I_n)$, donc par a), $\det A$ a même parité que $(n-1)$.

Exercice B

1) $M(t)$ est symétrique réelle, donc diagonalisable.

2) `L = sorted(L)`

3)
```
def L(t) :
    return sorted(alg.eigvals(array([[t,0,1],[0,0,1],[1,1,0]])))
T = np.arange(-20,20,0.1)
```

```
for i in range(3) :
    plt.plot(T, [ M(t)[i] for t in T] )
```
variante sans boucle : `plt.plot(T, [ M(t) for t in T] )`
```
plt.show()
```

On conjecture $\lim_{t\to+\infty} \lambda_1(t) = -1$, $\lim_{t\to+\infty} \lambda_2(t) = -1$ et $\lim_{t\to+\infty} \lambda_3(t) = +\infty$.

4) Les signes χ_t en $-\infty$, -1, 1 et $+\infty$ sont resepctivement -1, $+1$, -1 et $+1$.

Conclure avec le TVI et le fait que χ_t admet au plus trois racines.

Donc ce sont les seules racines de χ_t et elles sont simples.

5) Soit $\varepsilon > 0$. On a $\chi_t(-1) = +1$ et $\lim_{t\to+\infty} \chi_t(-1-\varepsilon) = +\infty$.

Donc $\chi_t(-1-\varepsilon)\chi_t(-1) < 0$ pour t assez grand. Donc $\lambda_1(t) \in]-1-\varepsilon, -1[$ pour t assez grand.

Comme ε est choisi arbitrairement, on a $\lim_{t\to+\infty} \lambda_1(t) = -1$.

On montre de même que $\chi_t(1-\varepsilon)\chi_t(1) < 0$ pour t assez grand. Donc $\lim_{t\to+\infty} \lambda_2(t) = -1$.

Par les relations entre coefficients et racines d'un polynôme scindé, on a : $\lambda_1(t) + \lambda_2(t) + \lambda_3(t) = t$.

Donc $\lambda_3(t) = t + \mathfrak{o}_{+\infty}(1)$ et a fortiori, on a $\lim_{t\to+\infty} \lambda_3(t) = +\infty$.

Exercice C

1)

```
def A(n,a) :
    M = np.zeros((n,n)) ;  b = 1/a
    for i in range(n-1) :  M[i+1,i] = a ; M[i,i+1] = b
    return M
```

2)

```
for n in range(3,9) :
    for a in [-2,-1,1,2,3] :
        print( (a,n) , ":" ,alg.eigvals(A(n,a)) )
```

Remarque : Il vaut mieux afficher les différentes valeurs de a consécutives, car on observe que les valeurs propres ne dépendent pas de la valeur de a (car les matrices correspondantes sont semblables).

3) a)

`P = [0]*9 #` on commence par déclarer une liste de longueur 9, qui sert ensuite à calculer $[P_0, ..., P_8]$

```
P[0] = Polynomial([1])
P[1] = Polynomial([0,1])
P[2] = Polynomial([-1,0,1])
X = Polynomial([0,1])
for i in range(3,9) :
    P[i] = X*P[i-1] - P[i-2]
    print( "Coefficients de P", i, ":" , P[i])
```

b)

```
for i in range(3,9) :
    print( "Racines de P", i, ":" , P[i].roots)
```

c) Le polynôme P_n est le polynôme caractéristique de A_n.

On le prouve en développant le déterminant selon la première ligne (et en utilisant $ab = 1$).

Remarque : En fait, les A_n sont semblables les unes aux autres (avec le changement de base $(e_1, be_2, ..., b^{n-1}e_n)$.

4) Avec la récurrence définissant P_n, on montre par récurrence d'ordre 2 que $P_n(0) = 0$ si n impair et $(-1)^{n/2}$ sinon.

Donc la matrice A_n est inversible ssi n est pair.

Lorsque $a = 1$, la matrice A_n est symétrique donc diagonalisable

Comme les A_n sont semblables les unes aux autres, il en est de même dans le cas général.

5) Si $|\lambda| > |a| + |b|$, la matrice A_n est à diagonale dominante, donc inversible (on prouve $AX = 0 \Rightarrow X = 0$).

Exercice D

Pour P et $Q \in E = \mathbb{R}_4[X]$, on pose $\phi(P, Q) = \int_{-2}^{+2} P(t)Q(t)\ dt$.

1)

```
from math import *
from scipy.integrate import *
from numpy.polynomial import *
def produit(P,Q) :
    v,e = integr.quad(P*Q,-2,2) ; return v
```

2) On applique le procédé de Gram-Schmidt d'une part à $(1, X^2, X^4)$ et d'autre part à (X, X^3).

4) A l'aide d'IPP, on vérifie que $\phi(f(P), Q) = \phi(P, f(Q))$.

```
from numpy import *
M = np.zeros((5,5)) ; x = Polynomial([0,1])
for i in range(5) :
    for j in range(5) :  M[i,j] = produit(x**i,y**j)
print(alg.eig(M))
```

Exercice E

1)
```
from numpy.random import *
def va(n) :
    s = 0 ;
    X = array( [ randint(-1000,1001) for i in range(n) ] )
    for k in range(n) :  s = s + X[k]**2
    norme = s**0.5
    if norme == 0 :  return va(n)
    return X/norme
```

Remarque : La probabilité de tomber sur le vecteur nul est proche de 0. Dans ce cas, on relance la fonction.

2)

```
A = zeros((4,4))
for i in range(4) :
     for j in range(4) :  A[i,j] = i+j-1
def hansdorff(A,X) :
     return dot(transpose(X),dot(A,X))
W = [] ; N =100
for n in range(N) : W.append(hansdorff(A,va(4)))
```

Une matrice carrée d'ordre 1 d'identifie à un réel : autrement dit, `dot` définit aussi le produit scalaire

```
plot(W,[0]*N,"o")    # On représente les points (α,0), où α = X^T AX et X vecteur aléatoire
print(eigvals(A))
```

3)

```
B = zeros((4,4))
for i in range(4) :
     for j in range(4) :
         if i = j+1 :  B[i,j] = 1
         else :  B[i,j] = 0
def hansdorff(B,X) :
     return dot(transpose(X),dot(B,X))
W = [] ; N =100
for n in range(N) : W.append(hansdorff(B,va(4)))
plot(W,[0]*N,"o")
print(eigvals(B))
```

Exercice F

1) `print((5+1j)**4*(239-1j))` renvoie `(114244+114244j)`

En sachant que pour tout $x > 0$, $\arg(x + iy) = \arctan(y/x)$, alors on obtient bien :

$\theta = 4\arctan(\frac{1}{5}) - \arctan(\frac{1}{239}) = \frac{\pi}{4}\ [2\pi]$. Comme $\arctan(\frac{1}{5}) \in [0, \frac{\pi}{4}]$, alors $\theta \in [-\pi, \pi]$. Donc $\theta = \frac{\pi}{4}$.

2) Si on sait que $\arctan(x) - \sum_{k=0}^{+\infty} \frac{(-1)^k}{2k+1} x^{2k+1}$, la propriété résulte de la majoration du reste d'une série alternée.

Sinon, on peut utiliser : $\arctan'(x) = \frac{1}{1+x^2} = \frac{1}{2i}\left(\frac{1}{x-i} - \frac{1}{x+i}\right)$.

En effet, $\arctan^{(2n+3)}(x) = \left(\frac{1}{1+x^2}\right)^{(2n+2)} = \frac{1}{2i}\left(\frac{1}{x-i} - \frac{1}{x+i}\right)^{(2n+2)} = \frac{(2n+2)!}{2i}\left(\frac{1}{(x-i)^{2n+3}} - \frac{1}{(x+i)^{2n+3}}\right)$.

Donc $\forall x \in [0,1]$, $\left|\arctan^{(2n+3)}(x)\right| \leq \frac{(2n+2)!}{2}(1+1) = (2n+2)!$

Par l'inégalité de Taylor-Lagrange, on a donc $\forall x \in [0,1]$, $\left|\arctan(x) - \sum_{k=0}^{+\infty} \frac{(-1)^k}{2k+1} x^{2k+1}\right| \leq \frac{(2n+2)!}{(2n+3)!}\,|x|^{2n+3}$.

On prend par exemple n tel que $\left(\frac{1}{5}\right)^n < 10^{-15}$ c'est-à-dire $n > 15\frac{\log 10}{\log 5}$, c'est-à-dire $n \geq 22$.

On a alors : $16\frac{1}{2n+1}\left(\frac{1}{5}\right)^n + 4\frac{1}{2n+1}\left(\frac{1}{239}\right)^n < 10^{-15}$.

D'où :

```
def somme(x)
     s = 0
     for k in range(23) :  s += (-1)**k/(2*k+1)*x**(2*k+1)
     return s
print( 16 * somme(1/5) - 4 * somme(1/239)) ;
```

On obtient bien 3.141592653589794

Math-info. Analyse. Corrigé

```
from math import *
from scipy import *
import numpy as np
import matplotlib.pyplot as plt
import scipy.integrate as integr
import scipy.optimize as resol
import numpy.random as rd
from numpy.polynomial import *
```

Exercice A

1) f_n' est strictement positive et f_n est une bijection strictement croissante de $\mathbb{R}$ sur $\mathbb{R}$. Donc f_n admet un unique zéro α_n et f_n est strictement positive sur l'intervalle $]\alpha_n, +\infty[$.

2)

```
def x(n) :
    def f(x) :  n*(x + x**3)
    return resol.fsolve(f,1)
T = [n for n in range(1,30)] ; X = [x(n) for n in range(1,30)] ; plot(T,X)
```

On a $\lim_{n\to+\infty} x_n = 0$.

En effet, pour tout $\varepsilon > 0$, on a $f_n(0) = 0$ et $\lim_{n\to+\infty} f_n(\varepsilon) = +\infty$, donc $0 \leq x_n \leq \varepsilon$ pour n assez grand.

Remarque : On pourrait aussi noter que $0 \leq x_n \leq \frac{1}{n}$, et conclure par pincement.

3) On a $\lim_{n\to+\infty} nx_n = 1$, c'est-à-dire $x_n \sim \frac{1}{n}$. En effet, vu que $\lim_{n\to+\infty} x_n = 0$, $x_n^3 = \mathfrak{o}\left(nx_n\right)$, donc $nx_n \sim 1$.

4) Le principe consiste évaluer $a = \lim_{n\to+\infty} n^2(x_n - \frac{1}{n})$ puis $b = \lim_{n\to+\infty} n^3(x_n - \frac{1}{n} - \frac{a}{n^2})$.

5) On procède par approximations successives : On pose $x_n = \frac{1}{n} + \varepsilon_n$, avec $\varepsilon_n = \mathfrak{o}\left(\frac{1}{n}\right)$.

On a $n\varepsilon_n + n(\frac{1}{n} + \varepsilon_n)^3 = 0$, donc $n\varepsilon_n \sim -n(\frac{1}{n})^3$, c'est-à-dire $\varepsilon_n \sim -\frac{1}{n^3}$.

Donc $x_n = \frac{1}{n} - \frac{1}{n^3} + \mathfrak{o}\left(\frac{1}{n^3}\right)$.

On a $(x_n)^{nx_n} = \frac{1}{n}\left(1 + O(\frac{1}{n^2}\right) e^{(\ln n)O(1/n^2))} \sim \frac{1}{n}$, donc la série $\sum (x_n)^{nx_n}$ diverge.

Exercice B

1) On pose $\forall x \geq 0$, $f_n(x) = e^{-x}\sum_{k=0}^{n} \dfrac{x^k}{k!}$. On a $f_n'(x) = -e^{-x}\dfrac{x^n}{n!}$.

La fonction f_n est une bijection décroissante de $[0, +\infty[$ sur $[0, 1[$.

2) $0 = f_n(a_n) \leq f_{n+1}(a_n)$, donc $f_{n+1}(a_{n+1}) \leq f_{n+1}(a_n)$, donc $a_{n+1} \leq a_n$ car f_{n+1} est strictement décroissante.

Soit $M \in \mathbb{R}$. On a $\lim_{n\to+\infty} f_n(M) = 1$, donc $f_n(M) > \frac{1}{2}$ pour n assez grand.

Comme $\lim_{+\infty} f_n = 0$, donc $x_n \geq M$ pour n assez grand.

3)
```
def a(n) :
    def f(x) :
        s = 1 ; y = 1
```

```
        for k in range(1,n+1) :  y = y*x/k ; s = s + y
        return s*exp(-x)-1/2
    x = 0 ; y = 2*n
    while abs(f(x))>0.001 :
    z = (x+y)/2
    if f(x)*f(z)<0 :  y = z
    else :  x = z
    return x
L = [k for k in range(1,21)] ; B = [a(n) for n in L]
plt.plot(L,B)
```

4) On prouve le résultat en utilisant une v.a. de loi de Poisson de paramètre $\mathcal{P}(x)$.

On a $f_n(x) = P(X \leq n) = \frac{1}{2}$.

Par BT, on a $P(|X - x| \geq \varepsilon) \leq \frac{x}{\varepsilon^2}$. Donc pour $\alpha < \frac{1}{2} < \beta$, $P(X \leq \alpha n) \to 0$ et $P(X \geq \beta n) \to 1$.

Donc $\alpha n < x < \beta n$ pour n assez grand.

Exercice C

1) La fonction intégrée est en $O_{+\infty}(e^{-t})$.

2)

```
def I(n) :
    v = integr.quad(lambda t :  exp(-t)*(sin(t))**(2*n),0,50)
    return v[0]
```

3) $0 \leq I_n \leq \sum_{k=0}^{+\infty} e^{-\pi k} J_n = \dfrac{1}{1 - e^{-\pi}} W_n$, où $W_n = \int_0^{\pi} (\sin t)^{2n}\ dt \to 0$ (intégrale de Wallis)

Donc $\lim_{n \to +\infty} I_n = 0$. Comme $\forall t \in [0, +\infty[,\ e^{-t}(\sin t)^{2n+2} \leq e^{-t}(\sin t)^{2n}$, la suite $(I_n)_{n \in \mathbb{N}}$ décroît.

Donc $\sum (-1)^n I_n$ converge car elle vérifie le critère spécial des séries alternées.

4) $\{n \in \mathbb{N} \mid I_n \leq \varepsilon\}$ est non vide (et même contient tous les entiers à partir d'un certain rang), donc $N(\varepsilon)$ existe.

```
def N(e) :
    n = 0
    while I(n)>e :  n += 1
    return n
```

5) Par la propriété sur les sommes des séries alternées, on a $R_n = \left|\sum_{k=n+1}^{+\infty} (-1)^k I_k\right| \leq I_{n+1}$. D'où :

```
n = N(10**(-3)) ; s = 0
for k in range(n) :  s += (-1)**k*I(k)
print(s)  # renvoie 0.7676
```

Exercice D

1)
```
def f(x) :
    def F(t) :  return sin(t)**3/t**2
    return integr.quad(F,0,x)[0]
```

```
X = arange(0,5*pi,0.1) ; Y = [ f(x) for x in X ] ; plt.plot(X,Y)
```

2) Par produit de Cauchy, $(\sin t)^3$ admet une DSE sur $\mathbb{R}$, et on a $(\sin t)^3 = \sum_{n=3}^{+\infty} a_n t^n$.

Donc $f(x) = \int_0^x \sum_{n=3}^{+\infty} a_n t^{n-2}\ dt = \frac{1}{n-1} \sum_{n=3}^{+\infty} a_n x^{n-1}$.

3) f' croît sur $[0, \frac{\pi}{2}]$ puis décroît sur $[\frac{\pi}{2}, \pi]$.

On a $J = \int_0^{\pi} \frac{(\sin t)^3}{t^2}\ dt = \int_0^{\pi/2} \frac{(\sin t)^3}{t^2} + \int_0^{\pi/2} \frac{(\sin(\pi - t))^3}{(\pi - t)^2}\ dt = \int_0^{\pi/2} \left(\frac{1}{t^2} - \frac{1}{(\pi - t)^2} \right)\ (\sin t)^3\ dt > 0$.

4) Comme $\forall x \in]0, \frac{\pi}{2}[$, $f'(x) > 0$ et f continue, alors f est strictement croissante sur $[0, \frac{\pi}{2}]$.

Donc f induit une bijection de $[0, \frac{\pi}{2}]$ sur $[0, M]$, qu'on note aussi f. D'où l'existence et l'unicité de $x_{k,n}$.

5) Par les sommes de Riemann, $\lim_{n \to +\infty} u_n = \int_0^1 \left(f^{-1}(\theta M) \right)^2\ d\theta$.

Or, $\int_0^1 \left(f^{-1}(\theta M) \right)^2\ d\theta = \int_0^{\pi/2} u^2 f'(u)\ du$ avec $f(u) = \theta M$ donc $M\ d\theta = f'(u)\ du$.

On en conclut que $\lim_{n \to +\infty} u_n = \int_0^{\pi/2} (\sin u)^3\ du = \int_0^{\pi/2} (1 - \cos^2 u)(\sin u)\ du = \int_0^1 (1 - x^2)\ dx = \frac{2}{3}$.

Exercice E

2) $a_0 = 1$ et $b_n = 0$; $b_{n+1} = \frac{1}{2} b_n + 1$ et $a_{n+1} = 2\sqrt{a_n}/b_{n+1}$.

3) On a $(b_{n+1} - 2) = \frac{1}{2}(b_n - 2)$, donc $b_n = 2 - \frac{1}{2^{n-1}}$, et en particulier $\lim_{n \to +\infty} b_n = 2$.

4)
```
def a(n) :
    b = 0 ; x = 1
    for k in range(n) :
        b = 1 + b/2 ; x = 2/b*x**(0.5)
    return x
T = liste(range(2,21)) ; A = [ a(n) for n in T ] ; plot(T,X) ; show()
```

Remarque : On peut montrer que la suite $(a_n)_{n \in \mathbb{N}}$ décroît et converge vers $L = 1$, car L vérifie $L = 2\sqrt{L}/2$.

5)
```
T = range(2,21,2) ;  X = np.arange(0,1.01,0.01)
clf()
for n in T :
    b = 2-2**(1-n)
    def f(x) :  return a(n)*x**b
    Y = [f(x) for x in X ] ; plt.plot(X,Y)
plt.show()
```

6) On a $\forall n \in \mathbb{N}^*$, $b_n \in [1, 2]$.

Donc $\forall x \in [0, 1]$, $f_n(x) - x^2 = a_n x^{b_n} - x^{b_n} + x^{b_n} - x^2$.

Par l'inégalité des accroissements finis, $\forall \beta \in [1, 2]$, $\left| x^{\beta} - x^2 \right| \leq (2 - \beta) \left| \ln x \right| x^{\theta} \leq (2 - \beta) \left| \ln x \right| x$, car $\theta \in [1, 2]$.

Il existe $M = \sup_{x \in]0,1]} (x \left| \ln x \right|) = e^{-1}$.

Donc $\forall n \in \mathbb{N}^*$, $\forall x \in [0, 1]$, $\left| f_n(x) - x^2 \right| \leq |a_n - 1| + e^{-1} \left| b_n - 2 \right|$.

Donc $(f_n)_{n \in \mathbb{N}}$ converge uniformément vers $f : x \longmapsto x^2$ sur $[0, 1]$.

Exercice F

1) On considère le système d'ordre 1 associé : $\begin{cases} y_0' = y_1 \\ y_1' = -y_1/x - y_0 \end{cases}$, et on représente les (x, y_0)

```
def f(Y,x) :  return np.array([Y[1],-Y[0]-Y[1]/x])
X = np.arange(0.001,20,0.01) ; X0 = np.array([1,0]) # prendre 0.001 car sinon problème en 0
V = np.array(odeint(f, X0 , T))
Y = V[:,0]
plt.plot(T,X) ; plt.show()
```

2) On approche $B(x)$ par $\sum_{n=0}^{N}(-1)^n \frac{1}{4^n(n!)^2}x^{2n}$, avec une erreur $\leq \frac{x^{2n}}{4^n(n!)^2}$.

On prend donc n tel que $20^{2n} \leq 4^n(n!)^2$, c'est-à-dire $10^n \leq n!$

```
n = 1 ; x = 10
while x > 1 :  n = n+1 ; x = x*10/n
print(n) # renvoie 25
```

```
def B(x) :
    s = 1 ; a = 1 ; m = 1
    for n in range(1,26) :
    a = (- a) * x**2 / 4 / n **2 ; s = s + a
            # on calcule les coefficients a_n en utilisant la récurrence a_0 = 1 et a_n = -a_{n-1} x^2/(4n^2)
    return(s)
print(B(2))
```

```
X = [ 0.1*k for k in range(201) ]
B = np.array([B(x) for x in X])
plt.plot(X,B) ; plt.show()
```

Math-info. Probabilités. Corrigé

```
from math import *
from scipy import *
import numpy as np
import matplotlib.pyplot as plt
import scipy.integrate as integr
import scipy.optimize as resol
import numpy.random as rd
from numpy.polynomial import *
```

Exercice A

Remarque : Pour simuler $S_{r,n}$ (mais inutile ici) :

```
def uniforme(r) :  return rd.randint(r+1)
def S(r,n) :
    s = 0
    for i in range(n) :  s = s + uniforme(r)
    return(s)
```

1) 2) $E(X) = \dfrac{1}{r+1}\sum_{k=0}^{r} k = \frac{1}{2}r$ et $G_X(z) = \dfrac{1}{r+1}\sum_{k=0}^{r} z^k$ et $G_{S_{r,n}}(z) = (G_X(z))^n$.

3)

```
def S(r,n) :
    L = [1 for i in range(r+1)]
    return (Polynomial(L)/(r+1))**n
def loi(r,n) :
    G = S(r,n)
    T = [k for k in range(n*r+1)] ; plt.pot(T,G.coeff) ; plt.show()
```

4) a)

```
def somme(k,G) :
    L = G.coef ; s = 0
    for i in range(min(k,len(L))) : s += L[i]
    return s
```

b)

```
def u(lam,r,n) :      # attention : lambda est un mot clé du langage Python
    G = S(r,n)
    return somme(int(lam*r*n),G)
for lam in [1/4,1/2,3/4] ] :
    print( [ u(l,3,2*k) for k in range(1,6) ] )
```

c)

On a par symétrie de la loi : $u_{p,n} = \frac{1}{2} + O\left(\frac{1}{\sqrt{p}}\right)$ si $\lambda = \frac{1}{2}$ (le terme correctif provient du terme central).

Montrons à l'aide de Bienaymé-Tchebychev que $\lim_{n\to+\infty} u_{p,n} = 0$ si $\lambda < \frac{1}{2}$ et $\lim_{p\to+\infty} u_{p,n} = 1$ si $\lambda > \frac{1}{2}$

On a $E(S_p) = \frac{1}{2}pn$. On a $V(S_p) = n\sigma^2$, où σ^2 ne dépend pas de n. Et $P(\left|S_p - \frac{1}{2}rn\right| \geq \gamma) \leq \frac{n\sigma^2}{\gamma^2}$.

On prend $\gamma = \varepsilon n$, avec $\varepsilon > 0$. Donc $\lim_{p\to+\infty} P((\frac{1}{2} - \varepsilon)n \leq S_{p,n} \leq (\frac{1}{2} + \varepsilon)n) = 1$, d'où on peut conclure.

5)

a)

```
def w(f,r,n) :
    G = S(r,n) ; L = G.coef ; s = 0
    for k in range(len(L)) : s = s + f(k/n)*L[k]
    return s
```

b)

```
def f(a) :
    def g(x) :  return x**a
    return g
print( [ w(f(a),3,10) for a in [1/2,1,2,3] ] )
```

c) En fait $\left(\frac{1}{n}S_{r,n}\right)_{n\in\mathbb{N}^*}$ converge en probabilités vers la v.a. constante de valeur $\frac{1}{2}r$, donc $\lim_{n\to+\infty} u_{r,n} = 0$ ou 1 selon que $\lambda < \frac{1}{2}$ et $\lambda > \frac{1}{2}$. Lorsque $\lambda = \frac{1}{2}$, on a par symétrie de la loi $\lim_{n\to+\infty} u_{r,n} = \frac{1}{2}$.

Exercice B

1)
```
def bernoulli(p) :  return rd.binomial(1,p)
def calculYZ(p) :
    x = bernoulli(p) ; y = 1
    while bernoulli(p) == x :  y += 1
    z = 1
    while bernoulli(p) == 1-x :  z += 1
    return array([y,z])
      # on renvoie un vecteur pour pouvoir calculer simplement sa moyenne
def esperance(V,N,p) :
    for k in range(N) : V = V + calculYZ(p)
    return V/N
```

2) On note x la valeur de X_1.

On a $\forall n \geq 1$, $P(Y \geq n) = E(Y \geq n \mid x = 1)P(x = 1) + E(Y \geq n \mid x = 0)P(x = 0)$.

Or, $E(Y \geq n \mid x = 1) = P(X_2 = X_3 = ... = X_n = 1) = p^{n-1}$ et $E(Y \geq n \mid x = 0) = q^{n-1}$

Donc $\forall n \geq 1$, $P(Y \geq n) = p^n \times p + q^{n-1} \times q = p^n + q^n$.

Donc $E(Y) = \sum_{n=0}^{+\infty} P(Y > n) = \sum_{n=1}^{+\infty} P(Y \geq n) = \sum_{n=1}^{+\infty} p^n + \sum_{n=1}^{+\infty} q^n = \frac{p}{q} + \frac{q}{p}$.

On a $\forall t > 0$, $t + \frac{1}{t} \geq 2$ (minimum atteint en $t = 1$). Donc $\frac{p}{q} + \frac{q}{p} \geq 2$.

3) On note x la valeur de X_1.

On a $(Y = n, Z = m) = p^n \times q^m \times p + q^n \times p^m \times q$: on considère la partition définie par $(x = 0)$ et $(x = 1)$.

On en déduit que $P(Z = m) = \sum_{n=1}^{+\infty}(Y = n, Z = m) = \frac{p^2}{q} \times q^m + \frac{q^2}{p} \times p^m = p^2q^{m-1} + q^2p^{m-1}$.

Donc $P(Z > m) = \sum_{k=m+1}^{+\infty} P(Z = k) = pq^m + qp^m$, et $E(Z) = \sum_{m=0}^{+\infty} P(Z > m) = \frac{p}{p} + \frac{q}{q} = 2$.

Interprétation : La formule se déduit de la notion d'espérance conditionnelle.

Les espérances conditionnelles de Y et Z sachant la valeur de x correspondent à l'espérance d'une loi géométrique.

On a $E(Y) = E(Y \mid x = 1)P(x = 1) + E(Y \mid x = 0)P(x = 0) = \frac{1}{1-p} \times p + \frac{1}{1-q} \times q = \frac{p}{q} + \frac{q}{p}$.

Et de même $E(Z) = E(Z \mid x = 1)P(x = 1) + E(Z \mid x = 0)P(x = 0) = \frac{1}{1-q} \times p + \frac{1}{1-p} \times q = 1 + 1 = 2$.

Exercice C

1)
```
def vaY() :
    y = 0
    N = rd.poisson(1/10)
    for k in range(N) : y = y + rd.binomial(50,1/50)
    return y
def momentY() :
    e = 0 ; v = 0
    for k in range(1000) :
        y = vaY() ; e = e + y ; v = v + y**2
    return e/N , (v/N-(e/N)**2)**(0.5)
```

2) a) Par la formule de Wald, on a $G_Y(z) = G_N(G_X(z))$ et $E(Y) = G'_Y(1) = G'_N(1)G'_X(1) = E(N)E(X)$.

b) On a $E(Y) = E(N)E(X) = \lambda np$ et $V(Y) = G''_Y(1) + G'_Y(1) - G'_Y(1)^2$.

Or, $G''_Y(z) = G''_N(G_X(z))G'_X(z)^2 + G'_N(G_X(z))G''_X(z)$, donc $G''_Y(1) = G''_N(1)G'_X(1)^2 + G'_N(1)G''_X(1)$.

On a $G_N(z) = e^{\lambda(z-1)}$ et $G_X(z) = (q + pz)^n$, donc $G''_N(1) = \lambda^2$ et $G''_X(1) = p^2n(n-1)$.

On en conclut $V(Y) = \lambda^2p^2n^2 + \lambda p^2n(n-1) + \lambda np - (\lambda pn)^2 = \lambda np + \lambda p^2n(n-1)$.

Exercice D

1) a) Posons $f_n(x) = e^{-x}\sum_{k=0}^{n} \frac{x^k}{k!}$. On a $f_n(0) = 1$, $\lim_{x\to+\infty} f_n(x) = 0$ et $f'_n(x) = -e^{-x}\frac{x^n}{n!} < 0$.

b) On a $f_{n+1}(x_{n+1}) = 0 = f_n(x_n) < f_{n+1}(x_n)$, et comme f_{n+1} décroît strictement, $x_{n+1} > x_n$.

2)
```
from math import * ; from scipy.optimize import * ; from scipy import *
def somme(n,x) :
    s = 1 ; y = 1
    for k in range(1,n+1) :  y = y*x/k ; s = s+y
def g(n) :
    f = lambda x :  exp(-x)*somme(n,x)
    return fsolve(f,1)
```

```
X = list(range(1,21))
Y = [ g(n) for n in X ]
plot(X,Y)
```

3) a) Considérons Z une variable aléatoire de loi de Poisson $\mathcal{P}(an)$. On a $e^{-an}\sum_{k=0}^{n}\frac{(an)^k}{k!} = P(Z \leq n)$.

On a $E(Z) = V(Z) = an$. Par Bienaymé-Tchebychev, on a : $P(|Z - an| \leq \varepsilon) \leq \frac{na}{\varepsilon^2}$.

Pour $a > 1$, $P(Z \leq n) \leq P(|Z - an| \geq n(1-a))$, donc $P(Z \leq n) \leq \frac{a}{n(1-a)^2} \to 0$ lorsque $n \to +\infty$.

Pour $a > 1$, $P(Z > n) \leq P(|Z - an| \geq n(a-1))$, donc $P(Z > n) \leq \frac{a}{n(1-a)^2} \to 0$ lorsque $n \to +\infty$.

b) Pour $a < 1 < b$, on a donc $\lim_{n\to+\infty} f_n(na) = 1$ et $\lim_{n\to+\infty} f_n(nb) = 0$.

Donc pour n assez grand, $f_n(na) < \lambda < f_n(nb)$, d'où $na \leq x_n \leq nb$ pour n assez grand.

On en déduit que $\frac{x_n}{n}$ peut être rendu arbitrairement proche de 1 pour n assez grand, c'est-à-dire $\frac{x_n}{n} \to 1$.

Exercice E

1)
```
def e(a,b) :
    M = array([3*a-2*b,-6*a+6*b+3],[a-b,-2*a+3*b+1])
    V,P = eig(M)
    return abs(V[1]-V[0])
```

2) a)
```
def hasard(p)
    N = 500 ; c = 0
    for k in range(N) :
        a = geometric(p) ; b = geometric(p)
        if ecart(a,b) > 1/10 :  c = c + 1
    return c
```

b)
```
P = [k/100 for k in range(1,100)] ; M = [ hasard(p)/500 for p in P]
plot(P,M)
F = [ (2-2*p+p**2)/(3-p) /500 for p in P] ; plot(P,F)
show()
```

c) $N(a,b)$ est diagonalisable ssi elle admet deux valeurs propres distinctes, c'est-à-dire $e(a,b) > 0$.

3) a) En résolvant $MX = (a+1)X$, on obtient le vecteur propre $(3,1)$.
On cherche ensuite $P = \begin{pmatrix} 2 & \alpha \\ 3 & \beta \end{pmatrix}$ inversible telle que $MP = PN$.

b) Justifier que $P(a+1=b) = \sum_{k=1}^{+\infty} P(a+1=k, b=k) = \sum_{k=1}^{+\infty} p^2 q^{2k-1} = \frac{p^2 q}{(1-q^2)} = \frac{pq}{(1+q)}$.

Donc $P(a+1 \neq b) = 1 - \frac{pq}{1+q} = \frac{2-2p+p^2}{2-p}$.

Remarque : Ici, on a $\{\lambda, \mu\} = \{a+1, b\}$, donc $e(a,b)$ est un entier naturel.

Autrement dit, $e(a,b) \geq \frac{1}{2}$ ssi λ et μ sont distincts. Donc la probabilité que $e(a,b) \geq \frac{1}{2}$ vaut $P(a+1 \neq b)$.

On prend $\frac{1}{2}$ et non pas 1 compte tenu des erreurs d'évaluations numériques (de même si on testait $\lambda = \mu$).

Exercice F

Remarque : S'il y avait remise, il s'agirait de la loi de premier succès de paramètre $\frac{1}{n}$.

1) *Remarque* : Il y a de nombreuses façons de coder le processus (avec tableau de marquage, liste triée, etc).

On considère ici une liste L contenant les n entiers de 1 à n.

Par des échanges successifs, on place à la k-ième place l'élément choisi à la k-ième étape.

Autrement dit, à la k-ième étape, on choisit aléatoirement un élément parmi les $n-k$ derniers éléments de la liste.

On arrête le processus dès l'obtention de l'élément n.

```
def vaX(n) :
    L = [i for i in tange(1,n+1)]
    for k in range(n) :
        j = randint(k,n)
        if L[j] == n :  return (k+1)
        L[j],L[k] = L[k],L[j]
def esperanceX(n) :
    e = 0 ; N = 1000
    for i in range(N) : e = e + vaX(n)
    return e/N
```

2) On a $\forall k \in \{0,1,...,n\}$, $P(X>k) = \frac{\binom{n-1}{k}}{\binom{n}{k}} = \frac{n-k}{n} = 1 - \frac{k}{n}$.

Remarque : En utilisant la formule de l'intersection, on peut retrouver ce résultat par le produit télescopique :

$P(X>k) = \frac{n-1}{n}\frac{n-2}{n-1}...\frac{n-k}{n-k+1} = \frac{n-k}{n}$.

On en déduit $E(X) = \sum_{k=0}^{n} P(X>k) = \sum_{k=0}^{n} \frac{n-k}{n} = \sum_{j=0}^{n} \frac{j}{n} = \frac{n+1}{2}$ (valeur intuitivement évidente).

Et $E(X^2) = \sum_{k=0}^{n}((k+1)^2 - k^2) = \sum_{k=0}^{n}(2k+1)P(X>k)$.

On en déduit après des calculs sans intérêt : $V(X) = E(X^2) - E(X)^2 = \frac{n^2-1}{12}$.

Exercice G

1)
```
def test(i) :
    L = [randint(3)]
    for k in range(2,31) :
        x = randint(3)
        if not(x in L) : L.append(x)
        if len(L) == i :  return k
```

2)
```
def repartition(i,n) :
    Y = zeros(31)
    for j in range(100) :
        k = test(i) ; Y[k] += 1
    return(Y)
```

Par exemple, `repartition(2,100)` renvoie $[0,0,58,31,7,4,0,0,0,...,0]$.

Et `repartition(3,100)` renvoie $[0,0,0,12,23,25,14,9,2,5,3,2,2,1,0,1,0,0,0,1,0,...,0]$.

3) $Y_3 - Y_2$ suit la loi géométrique (à valeurs dans $\mathbb{N}^*$) associée à la loi de Bernoulli $\mathcal{B}(\frac{1}{3})$.

On a notamment $P(Y_3 = n + k, Y_2 = n) = \frac{1}{3}\left(\frac{2}{3}\right)^{k-1}$ et $E(Y_3 - Y_2) = 3$.

Math-info. Exos supplémentaires. Corrigé

Exercice A

On considère $\forall x > 0$, $f(x) = \int_x^{+\infty} \frac{\sin t}{t^2}\, dt$ et $g(x) = \int_x^{+\infty} \frac{\cos t}{t^3}\, dt$.

1) f et g sont en $O_{+\infty}(t^{-2})$, donc intégrables. Par IPP (sur $[x, A]$ et $A \to +\infty$), on a $f(x) = \frac{\cos x}{x^2} - 2g(x)$.

2)

```
def g(x) :
     def gg(t) :  return cos(t)/t**3
     v = integr.quad(gg,x,100)
     return v[0]
X = np.arange(0.2,20,0.1) ; Y = [ x**2*g(x) for x in X] ; plt.plot(X,Y) ; plt.show()
```

3)

```
def f(x) :
     v = integr.quad(lambda t :  sin(t)/t**2,x,100)
     return v[0]
def x(n) :
     y = resol.fsolve(f,n*pi,(n+1)*pi)
     return y[0]
N = range(1,105,5) ; X = [x(n) for n in N] ; clf() ; plot(N,X) ; show()
```

On conjecture $\lim_{n\to+\infty} 2x_n - n\pi = \pi$.

Par 1), on a $f(x_n) = 0$, donc $\cos x_n = 2x_n^2 g(x_n)$. Comme $\lim_{n\to+\infty} x_n = +\infty$, on a $\lim_{n\to+\infty} x_n^2 g(x_n) = 0$.

Donc $\lim_{n\to+\infty} \cos x_n = 0$. Comme $x_n - n\pi \in]0, \pi[$, alors $x_n - n\pi = \arccos(\cos x_n)$.

Donc $x_n - n\pi \to \arccos(\cos 0) = \frac{\pi}{2}$.

4) On a $f(n\pi) = \int_{n\pi}^{+\infty} \frac{\sin(t)}{t^2}\, dt$.

On découpe l'intégrale selon la subdivision (infinie) des segments $[k\pi, (k+1)\pi]$.

Donc $f(n\pi) = \sum_{k=n}^{+\infty} \int_{k\pi}^{(k+1)\pi} \frac{\sin(t)}{t^2}\, dt$. Or, $\int_{k\pi}^{(k+1)\pi} \frac{\sin(t)}{t^2}\, dt = \int_0^{\pi} \frac{\sin(k\pi+\theta)}{(k\pi+\theta)^2}\, d\theta = (-1)^k \int_0^{\pi} \frac{\sin\theta}{(\theta+k\pi)^2}\, d\theta$.

On vérifie aisément que la suite $k \longmapsto \int_0^{\pi} \frac{\sin\theta}{(\theta+k\pi)^2}\, d\theta$ est strictement décroissante et positive.

Par la propriété des sommes des séries alternées, $f(n\pi) > 0$ si n pair, et < 0 si n impair.

D'où l'existence de x_n.

D'autre part, on a $f'(x) = -\frac{\sin(x)}{x^2}$, donc f est strictement monotone sur chaque intervalle $[n\mu, (n+1)\pi]$.

Exercice B

1) On a $|X_n| \leq \sum_{k=1}^{+\infty} 2^{-k} = 1$.

Remarque utile pour la suite : La définition de X_n est liée à l'écriture en base 2 des nombres réels. Si les ε_k valaient 0 ou 1, la loi de X_n serait la loi uniforme sur l'ensemble des nombres réels de $[0, 1[$ s'écrivant en base 2 avec n bits (ε_k représentant alors le k-ième chiffre du développement en base 2).

Ici, par un simple changement affine $[0,1] \to [-1,1]$, on en déduit que X_n suit une loi uniforme sur un ensemble équiréparti de $[-1,1]$ dont la limite (la réunion) quand $n \to +\infty$ est dense dans $[-1,1]$.

D'où, pour toute fonction $f \in C^0([-1,1])$, $\lim_{n\to+\infty} E(f(X_n)) = \frac{1}{2}\int_{-1}^{+1} f(x)\,dx$ valeur moyenne de f sur $[-1,1]$.

2) Posons $S_{n,N} = \dfrac{1}{N}\sum_{k=1}^{N} \cos(tX_{n,k})$. On a $E(S_{n,N}) = E(X_n)$ et $V(S_N) = \dfrac{N\ V(X_n)}{N^2} = \dfrac{V(X_n)}{N}$.

Par Bienaymé-Tchebychev, $P\left(|S_{n,N} - E(S_{n,N})| \geq \varepsilon\right) \leq \dfrac{V(S_N)}{\varepsilon^2} \to 0$ lorsque $N \to +\infty$.

3)
```
def Rademacher() :  return 2*binomial(1,1/2)-1
def X(n) :
     s = 0
     for k in range(1,n):  s += Rademacher() / 2**k
     return s

def moyenne(n,t) :
     A = array([ cos(t*X(n)) for k in range(1000) ])
     return mean(A)

def graphe(n) :
     X = arange(-10,10.1,0.1) ; Y = [ moyenne(n,t) for t in X ]
     plot(X,Y)

graphe(10) ; graphe(20)  # trop de graphes superposés → problèmes possibles ....
show()
```

On conjecture que la suite des fonctions $t \longmapsto E(\cos(tX_n)$ converge.

Compte tenu de la remarque du 1), la limite est $L(t) = \frac{1}{2}\int_{-1}^{+1} \cos(tx)\,dx = \frac{\sin t}{t}$ fonction sinus cardinal.

4) Par indépendance des $e^{it\varepsilon_k}$, On a $\phi_{X_n(t)} = \prod_{k=1}^{n} E(e^{it\varepsilon_k}) = \prod_{k=1}^{n} \cos\left(\dfrac{t}{2^k}\right)$.

En utilisant $\cos\theta = \dfrac{1}{2}\dfrac{\sin(2\theta)}{\sin\theta}$, on a $\phi_{X_n(t)} = \prod_{k=1}^{n} \cos\left(\dfrac{t}{2^k}\right) = \dfrac{1}{2^n}\dfrac{\sin(t)}{\sin(t/2^n)}$.

Donc $\lim_{n\to+\infty} \phi_{X_n(t)} = \dfrac{\sin t}{t}$, car $\sin\left(\dfrac{t}{2^n}\right) \sim \dfrac{t}{2^n}$ lorsque $n \to +\infty$.

Exercice C

1) Pour construire dans le cas de matrices de d'ordre 2 :

```
def tensoriel(A,B) :
     M0 = np.concatene((A[0,0]*B,A[0,1]*B),axis = 1)
     M1 = np.concatene((A[1,0]*B,A[1,1]*B),axis = 1)
     return np.concatene(M0,M1,axis = 0)
```

Complément culturel hors-oral : Pour construire dans le cas général :

```
def tensoriel(A,B) :
     n = len(A) ; p = len(B)
     for i in range(p) :
```

```
        M[i] = A[i,0]*B
        for j in range(1,n) :  M[i] = concatene((M[i],A[i,j]*B),axis = 1)
    M = M[0]
    for i in range(p) :  M = concatene((M,M[i]),axis = 0)
    return M
```

Autre méthode : `def tensoriel(A,B) :`

```
    n = len(A) ; p = len(B) ; m = n*p ; v = zeros((m,m))
    for i in range(m) :
        for j in range(m) :  M[i,j] = A[i%n,j%n]*B[i//n,j//n]
    return(M)
```

2) `A = np.array( [[1,4],[2,1]]) ; B = array([[1,2],[2,1]])`

`M = tensoriel(A,B) ; print(M)`

`alg.eigvals(M)` renvoie `[ 11.48528137,1.82842712,-3.82842712,-5.48528137]`

Il y a donc quatre valeurs propres distinctes, donc $A \otimes B$ est diagonalisable.

`C = np.array([[1,1],[0,1]]) ; N = tensoriel(A,C)`

`V,P = eig(N)` renvoie $V = [3.82842712, -1.82842712, 3.82842712, -1.82842712]$

`det(P)` vaut 0 (en fait 10^{-32}) : La matrice P donnant des vecteurs générateurs des sev propres n'est pas inversible.

Donc N n'est pas diagonalisable.

4) 5) 6) 7) En considérant le produit par blocs, on vérifie que $(A \otimes B)(A' \otimes B') = (AA') \otimes (BB')$.

On en déduit que si P et Q sont inversibles, alors $(P \otimes Q)$ est inversible et $(P \otimes Q)^{-1} = (P^{-1}) \otimes (Q^{-1})$.

Donc si $A' = P^{-1}AP$ et $B' = Q^{-1}BQ$, alors on a $(A \otimes B) = (P \otimes Q)^{-1}(A' \otimes B')(P \otimes Q)$.

En particulier, si A et B sont diagonalisable, alors $A \otimes B$ l'est aussi.