TD Math & Python (les questions informatiques sont marquées d'un (♣))

On charge d'abord les modules dont on a besoin (en prenant les alias du memento)

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

Exercice A

1) Montrer que pour tout $n \in \mathbb{N}$, l'équation $e^{-x} \sum_{k=0}^{n} \frac{x^k}{k!} = \frac{1}{2}$ admet une unique solution $a_n \geq 0$.

2) Montrer que $(a_n)_{n\in\mathbb{N}}$ est croissante et tend vers $+\infty$.

3) (♣) Écrire en PYTHON une fonction calculant a_n. Représenter $(\frac{a_n}{n})_{10\leq n\leq 20}$.

Attention : On utilisera ici la fonction `resol.bisect(f,0,2*n)` qui procède par dichotomie, la fonction `resol.solve` qui utilise la méthode de Newton marche très mal avec la fonction considérée.

4) (*difficile*) Montrer mathématiquement que $a_n \sim n$ lorsque $n \to +\infty$.

Exercice B

Soit $M(t) = \begin{pmatrix} t & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$ où $t \in \mathbb{R}^+$.

1) Montrer que les valeurs propres de $M(t)$ sont réelles. On les note $\lambda_1(t) \leq \lambda_2(t) \leq \lambda_3(t)$.

2) (♣) On donne la fonction `sorted(L)` qui étant une liste (ou un tableau) L renvoie la liste triée.

Représenter sur un même graphe les fonctions $t \longmapsto \lambda_i(t)$, avec $i \in \{1, 2, 3\}$.

3) Le polynôme caractéristique χ_t de $M(t)$ vaut $\chi_t(x) = x^3 - 2x - t(x^2 - 1)$.

En déduire que $\lambda_1(t) < -1 < \lambda_2(t) < 1 < \lambda_3(t)$.

4) Déterminer les limites de λ_1, λ_2 et λ_3 en $+\infty$.

Indication : Utiliser $\lim_{t\to+\infty} \chi_t(M)$ pour justifier $\lim_{t\to+\infty} \lambda_3(t) = +\infty$.

Exercice C

Remarque :

- On déduit de `binomial` la loi de Bernoulli : `binomial(1,p)` est une variable de Bernoulli
- On peut définir une variable de Rademacher par : `2*randint(0,2)-1`

On considère une suite $(X_n)_{n\in\mathbb{N}}$ de variables de Bernoulli i.i.d. de paramètre $p \in]0, 1[$. On pose $q = 1 - p$.

On note Y la longueur de la première série de 0 consécutifs ou de 1 consécutifs.

On note Z la longueur de la deuxième série de 1 consécutifs ou de 0 consécutifs.

Par exemple, si $(X_n)_{n\in\mathbb{N}} = (1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, ...)$, on a $Y = 3$ et $Z = 5$.

Note Bene : Comme $0 < p < 1$, $P(Y = +\infty) = 0$, donc Z est bien définie, et Y et Z sont à valeurs dans $\mathbb{N}^*$.

1) (♣) Écrire un programme en PYTHON permettant d'évaluer le couple $(E(Y), E(Z))$.

2) Donner la loi de Y. Montrer que $E(Y) = \dfrac{q}{p} + \dfrac{p}{q}$, et justifier brièvement que $E(Y) \geq 2$.

3) Donner la loi conjointe de (Y, Z). En déduire que $E(Z) = 2$.

Exercice D

On considère sur $\mathbb{R}[X]$ le produit scalaire $\langle P, Q\rangle = \int_0^1 \dfrac{P(t)Q(t)}{\sqrt{1-t^2}}\, dt$.

1) (♣) Définir une fonction `produit(P,Q)` qui étant donnés deux polynômes, renvoie una valeur numérique approchée du produit scalaire $\langle P, Q\rangle$.

2) (♣) Écrire une fonction `proj(n)` qui étant donné un entier $n \in \mathbb{N}^*$ renvoie le projeté orthogonal P de X^n sur $\mathbb{R}_{n-1}[X]$ (ou plutôt le vecteurs des coefficients de ce polynôme P).

Indication : Avec $P = \sum_{j=0}^{n-1} a_i X^i$, on a $\forall j \in [\![0, n-1]\!]$, $\left\langle X^n - P, X^i\right\rangle = 0$.
Il s'agit donc de résoudre le système linéaire $\forall j \in [\![0, n-1]\!]$, $\sum_{j=0}^{n-1} a_j \left\langle X^i, X^j\right\rangle = \left\langle X^n, X^i\right\rangle$.
Construire la matrice $A = (\left\langle X^i, X^j\right\rangle)_{0\leq i<n, 0\leq j<n}$ et le vecteur $Y = (\left\langle X^n, X^i\right\rangle)_{0\leq i<n}$.
Utiliser la fonction `alg.solve` (du calcul matriciel) pour résoudre le système et obtenir $(a_j)_{0\leq j\leq n}$.

3) (♣) En déduire une fonction `distance` qui étant donné un entier $n \in \mathbb{N}^*$, renvoie la valeur de

$$d_n = \inf_{P\in\mathbb{R}_{n-1}[X]} \int_0^1 \frac{(t^n - P(t))^2}{\sqrt{1-t^2}}\, dt$$

4) (♣) Représenter la suite $(d_n)_{1\leq n\leq 10}$.

Corrigé

A.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 $[0, 1[$.

A.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.

A.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
        ### variante avec : s = s + x**k/factorial(k)
        return s*exp(-x)-1/2
    return resol.bisect(f,0,2*n)
L = [k for k in range(10,21)] ; B = [a(n) for n in L]
plt.plot(L,B)
```

A.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.

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

B.2)
```
def L(t) :
    L = alg.eigvals(array([[t,0,1],[0,0,1],[1,1,0]]))
    return sorted(L)

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 $\lim_{t\to+\infty} \lambda_3(t) = +\infty$.

B.3) Les signes χ_t en $-\infty$, -1, 1 et $+\infty$ sont respectivement -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.

B.4) 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 $\lim_{t\to+\infty} \lambda_3(t) = +\infty$.

C.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
```

C.2) On note x la valeur de X_1.

On a $\forall n \in \mathbb{N}^*$, $P(Y \geq n) = P(Y \geq n \mid x = 1)P(x = 1) + P(Y \geq n \mid x = 0)P(x = 0)$.

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

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

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

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

C.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 $P(Z=m) = \sum_{n=1}^{+\infty}(Y=n, Z=m) = \dfrac{p^2}{q} \times q^m + \dfrac{q^2}{p} \times p^m = p^2 q^{m-1} + q^2 p^{m-1}$.

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

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

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

D.1)
```
def produit(P,Q) :
    def f(t) :
        return P(t)*Q(t)/(1-t**2)**0.5
    return integr.quad(f,0.01,0.99)[0]
      # attention à ne pas oublier la sélection [0] dans quad
```

On teste sur un exemple :

```
P = Polynomial([1,1]) ; Q = Polynomial([1,2]) ; print(produit(P,Q))
```

D.2)
```
def proj(n) :
    A = np.zeros((n,n)) ; Y = np.zeros(n)
    X = Polynomial([0,1])    # définit le polynôme X
    for i in range(n) :
        for j in range(n) :
            A[i,j] = produit(X**i,X**j)
        Y[i] = produit(X**i,X**n)
    return alg.solve(A,Y)
```

Remarque : Pour définir des matrices $n \times n$, commencer par définir la matrice nulle, puis modifier les coefficients un par un à l'aide de deux boucles `for` imbriquées.

On teste sur un exemple :

```
print(proj(3))    #    renvoie   [ 0.06399327 -0.68292373 1.58539687]
```

D.3) On a $d_n = d(X^n, \mathbb{R}_{n-1}[X])^2 = \|X^n - P\|^2$, où P est le projeté de X^n sur $\mathbb{R}_{n-1}[X]$. D'où :

```
def d(n) :
    P = Polynomial(proj(n)) ; X = Polynomial([0,1])
    Q =X**n-P
    return produit(Q,Q)
```

On teste sur un exemple :

```
print(proj(3))    #    renvoie    2.889228028874979e-05
```

D.4)
```
N = [n for n in range(1,11)] ; D = [d(n) for n in N]
plt.plot(N,D); plt.show()
```

On obtient une suite qui décroît très rapidement vers 0.