DS07

Problèmes stationnaires et algèbre linéaire
Sources :

Proposition de corrigé

Exercice 1 : Quelques exemples d'analyse

Importation des modules nécessaires :

```
import scipy.optimize as so
import scipy.integrate as si
from math import sqrt, sin, cos, atan
from numpy import array
import numpy as np
```

Q 1 : Donner une approximation de $\int_{\alpha}^{\alpha+1} \cos(\sqrt{t})\, dt$ avec la méthode des trapèzes et 1000 subdivisions.

```
def trapeze(f,a,b,n):
    h=(b-a)/n
    S=0.5*(f(a)+f(b))
    for k in range(1,n):
        S+=f(a+k*h)
    return S*h

print("Qu. 1 : ", trapeze(lambda x : cos(sqrt(x)),alpha,alpha+1,1000))
```

Q 2 : Donnez une approximation (à 10^-5 près) de l'unique réel positif solution de l'équation $x^2+\sqrt{x}-10=\alpha$ avec la méthode de votre choix.

```
def newton(f, fp, x0, epsilon):
    """Zéro de f par la méthode de Newton
       départ : x0, f' = fp, critère d'arrêt epsilon"""
    u = x0
    v = u - f(u)/fp(u)
    k=0
    while abs(v-u) > epsilon:
        u, v = v, v - f(v)/fp(v)
        k+=1
    return u,k

def f2(t):
```

```
    return t**2+t**0.5-10-alpha

def f2p(t):
    return 2*t+0.5*t**(-0.5)

print("Qu. 2 : méthode de Newton ", newton(f2,f2p,1,1e-5)[0])
```

Q 3 : Donnez le nombre d'itérations nécessaires pour obtenir ce résultats avec la méthode de Newton en prenant pour valeur initiale α.

```
print("Qu. 3 : méthode de Newton ", newton(f2,f2p,alpha,1e-5)[1])
```

Q 4 : Donnez le nombre d'itérations nécessaires pour obtenir ce résultats avec la méthode de Dichotomie sur l'intervalle $[0, 12+\alpha]$.

```
def dichotomie(f, a, b, epsilon):
    """Zéro de f sur [a,b] àepsilon près, par dichotomie
       Préconditions : f(a) * f(b) <= 0
                       f continue sur [a,b]
                       epsilon > 0"""
    c, d = a, b
    fc, fd = f(c), f(d)
    k=0
    while d - c > 2 * epsilon:
        m = (c + d) / 2.
        fm = f(m)
        if fc * fm <= 0:
            d, fd = m, fm
        else:
            c, fc = m, fm
        k+=1
    return (c + d) / 2.,k

print("Qu. 4 : méthode de dichotomie ", dichotomie(f2, 0, 12+alpha, 1e-5)[1])
```

Q 5 : Donnez à l'aide une approximation (à 10^-5 près) de l'unique réel positif t tel que

$$\int_{\alpha}^{\alpha+t} (2+\sqrt{x}+\cos x)\, dx = 10$$

```
print("Qu. 5 : ", dichotomie(lambda t : trapeze(lambda x : 2+sqrt(x)+cos(x),alpha,
    alpha+t,1000)-10,0,50,1e-5))
```

Q 6 : Donner une valeur approchée de $x(1)$ avec x l'unique fonction vérifiant $x(0) = \alpha$ et pour tout $t \in \mathbb{R}$, $x'(t) = 3\cos(x(t)) + t$.

```
def F (x,t) :
    return 3*cos(x) + t

les_t = [i/10000 for i in range(10001)]

print("Qu. 6 : ", si.odeint(F,alpha,les_t)[-1,0])
```

Q 7 : Donner une valeur approchée de $x(1+\frac{\alpha}{10})$ avec x l'unique fonction vérifiant $x(0) = 0$, $x'(0) = 0$ et pour tout $t \in \mathbb{R}$, $x''(t) = 1+\sin(t+x(t))$.

```
def G (X,t) :
    a,b = X[0],X[1]
    return array([b,1+sin(t+a)])

les_t = [i*(1+alpha/10)/10000 for i in range(10001)]

print("Qu. 7 : ", si.odeint(G,array([0,0]),les_t)[-1,0])
```

Q 8 : Donner une approximation de $\beta \in \mathbb{R}$, pour que l'unique solution de l'équation différentielle non linéaire $x''(t) = 1 + \arctan(t + x(t))$ avec les conditions initiales $x(0) = 0$ et $x'(0) = \beta$ vérifie

$$x(1+\frac{\alpha}{10}) = 1 + \frac{2}{3}\alpha$$

```
def H (X,t) :
    a,b = X[0],X[1]
    return array([b,1+atan(t+a)])

les_t = [i*(1+alpha/10)/10000 for i in range(10001)]

f = lambda beta : si.odeint(H,array([0,beta]),les_t)[-1,0]-1-(2/3)*alpha

print("Qu. 8 : ",so.brentq(f,-2,2))
```

Exercice 2 : Algèbre linéaire

Importation des modules nécessaires :

```
from numpy import array
import numpy as np
```

Q 9 : Résoudre

$$A\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 3 \\ \alpha \end{pmatrix}$$

Donner la valeur de x_1.

```
A=array([[0,1,32,243],[1,32,243,1024],[32,243,1024,3125],[243,1024,3125,7776]])
B=array([[1],[2],[3],[alpha]])

X=np.linalg.solve(A,B)

print("Qu. 9 : ",X[0][0])
```

Q 10 : Calculer $B = A^3$ et donner le reste du coefficient de B situé sur la première ligne et la première colonne de B (donc d'indices 0 et 0 en numpy) dans la division par $10\,000 + \alpha$.

```
  B=np.dot(A,np.dot(A,A))
print("Qu. 10 : ",B[0][0]%(10000+alpha))
```