Devoir Surveillé 3 - CORRECTION

Durée : 2 heures
Calculatrice interdite

Exercice 0.1. *Nous connaissons l'accélération instantanée d'un mobile en fonction du temps t, notée a telle que $a(t) = \frac{t}{2}$ et nous savons que la vitesse initiale de ce dernier est nulle.*

1. *Exprimer la vitesse $v(T)$ de ce mobile après T secondes grâce à une intégrale.*
   $v(T) = \int_0^T a(t)\,\mathrm{d}t = \int_0^T \frac{t}{2}\,\mathrm{d}x$
2. *Approximer numériquement cette vitesse grâce à la méthode de votre choix. Pour cela, on définira une fonction* ***vitesse(a,n,T)*** *recevant comme paramètres :*
   - *l'accélération*
   - *le nombre n de subdivisions*
   - *l'instant T à laquelle la vitesse doit être estimée.*

```
def a(t):
    return (t/2)

def trapezes(a, n, T):
    s=0
    x=0
    h=T/n
    for i in range(n):
        s=s+a(x)+a(x+h)
        x=x+h
    return(s*h/2)

print(trapezes(a,1000,1))
```

Exercice 0.2. *On dispose en Python de la fonction* ***randint*** *dans la bibliothèque* ***random****. Elle prend deux arguments entiers a et b, et renvoie un entier aléatoire entre a et b... inclus.*
On considère ici à des lancers successifs d'une pièce « non biaisée » : elle retombe sur PILE avec probabilité 1/2, et sur FACE avec la même probabilité .

1. *Écrire une fonction* ***lancer()*** *ne prenant pas d'argument, mais renvoyant la chaîne de caractères ou bien 'Pile' ou bien 'FACE', chacun avec probabilité $\frac{1}{2}$.*

```
from random import randint

def lancer():
    l=randint(0,1)
    if l==0:
        return('Pile')
    else:return('Face')
```

2. *Écrire une fonction* ***nbre_face(N)*** *prenant en entrée un entier N, réalisant N tirages, et renvoyant le nombre de tirages égaux à FACE.*

```
def nbre_face(N):
    s=0
    for i in range(N):
        l=randint(0,1)
        if l==1:
            s=s+1
    return(s)
```

3. *Écrire une fonction* ***stat(N)*** *prenant en entrée un entier $N > 0$, réalisant N tirages aléatoires $x_0, ..., x_{N-1}$ d'entiers égaux à 0 ou 1, et retournant deux paramètres :*
   - *une liste contenant les valeurs $x_0, ..., x_{N-1}$*
   - *la moyenne de toutes ces valeurs.*

```
def stat(N):
    s=0
    Liste=[]
    for i in range(N):
        l=randint(0,1)
        Liste.append(l)
        if l==1:
            s=s+1
    return(Liste,s/N)
```

Exercice 0.3. *Un filtre passe-bas d'ordre 2 dit de Butterworth est un filtre dont la fonction de transfert canonique s'écrit*

$$\underline{H}(jx) = \frac{1}{1+j\sqrt{2}x-x^2}$$

où $x = \frac{\omega}{\omega_0}$ *est la pulsation réduite.*
On montre que l'expression de son gain linéaire se met sous la forme

$$|\underline{H}| = \frac{1}{\sqrt{1+x^4}}$$

et celle de son gain en dB sous la forme $G_{dB} = -10log(1+x^4)$.
On cherche à déterminer la pulsation réduite de coupure de ce filtre, c'est-à-dire la pulsation pour laquelle $G_{x_C} = -3dB$.
Cela revient à rechercher le zéro de la fonction f *définie sur* $[0; +infty[$ *par*

$$f(x) = 10log_{10}(1+x^4) - 3.$$

La lecture du diagramme de Bode du filtre nous permet d'affirmer que le zéro de f*, noté* x_c*, appartient à l'intervalle* $[0; 1, 2]$.

1. *Écrire une fonction* ***sol_dichotomie(precision)*** *d'unique argument un flottant* ***precision*** *et qui retourne une valeur approchée de* x_c*, calculée par dichotomie, avec une erreur inférieure à la valeur precision passée en paramètre.*

   Indication : *le logarithme la base 10 peut être calculer à partir de la fonction* ***log10()*** *de la bibliothèque* ***numpy****, prenant comme argument un flottant.*

```
# On définit la fonction f
def f(x):
    return 10.*np.log10(1.+x**4.)-3.

# On définit la fonction dichotomie

def sol_dichotomie(precision):

    # Initialisation des bornes
    a=0
    b=1.2

    # Construction de la suite d'intervalles
    while b-a > precision:
        m = (a+b)/2.     # Milieu de l'intervalle

        # Si on tombe directement sur un zéro, on arrête le programme
        if f(m)==0.:
            return m
        # Recherche de la moitié de l'intervalle contenant une solution à f (x) = 0
        elif f(a)*f(m)<0.:
            b=m
        else:
            a=m

    return a
```

2. *Écrire un script sous Python permettant de faire afficher la courbe représentative de la fonction f sur l'intervalle $[0;1,2]$ en construisant 1000 points.*
   Indication :

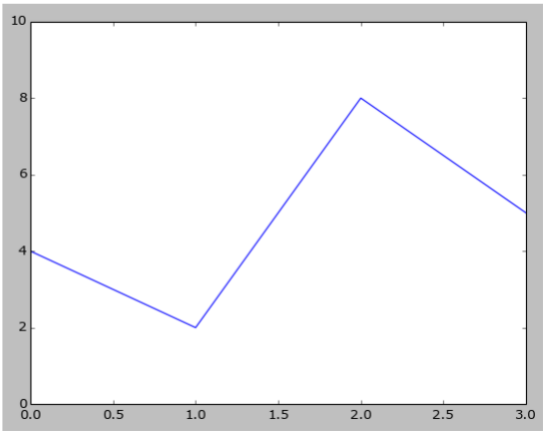
A l'aide de la bibliothèque Matplotlib, et de son instruction plot(), il est possible de tracer des courbes qui relient des points dont les abscisses et ordonnées sont fournies dans des listes. Pour cela, on utilise la suite d'instructions suivantes :

```
import matplotlib.pyplot as plt
plt.clf()                    # Efface le graphique précédent
plt.plot(list_x,list_y)      # Trace la courbe
plt.ylim(ymin,ymax)          # Définit la plage des ordonnées
plt.show()                   # Affiche la courbe
```

Par exemple pour :

```
list_x=[0,1,2,3]
list_y=[4,2,8,5]
ymin = 0
ymax = 10
```

On obtient la courbe ci-contre :

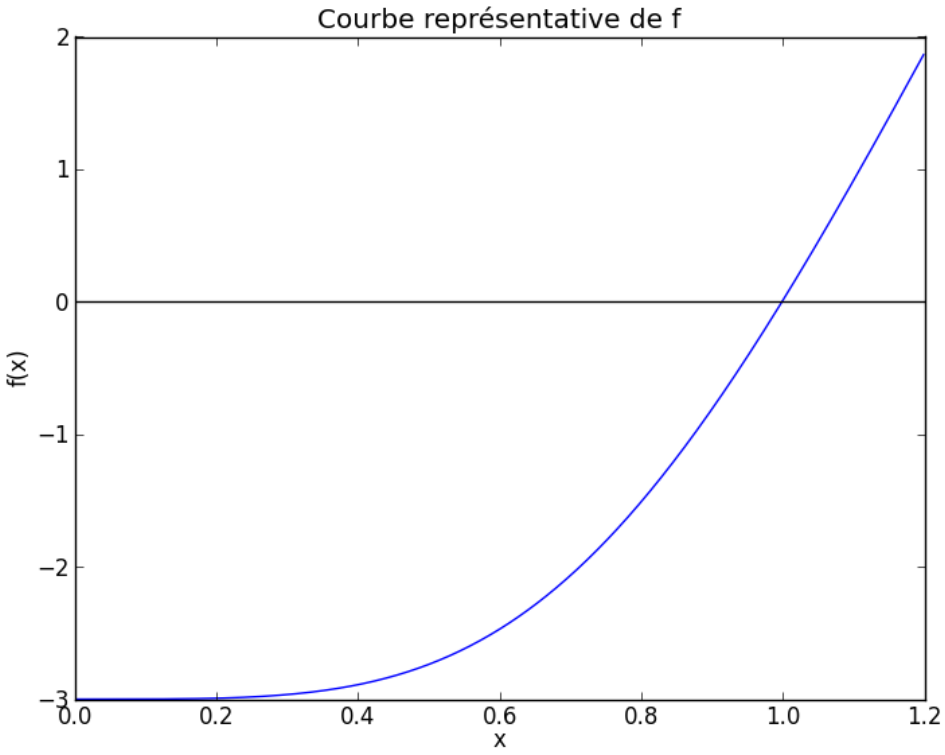


```
# On définit les vecteurs abscisse x et ordonnée y
import matplotlib.pyplot as plt
L_abs=[]
L_ord=[]
h=(1.2-0)/1000
for i in range(1000):
    x=i*h
    L_abs.append(x)
    L_ord.append(f(x))

# On affiche la courbe
plt.clf()
pl.plot(L_abs,L_ord)

# Un titre, des noms pour les axes...
plt.title('Courbe représentative de f')
plt.xlabel('x')
plt.ylabel('f(x)')

# Un axe des abscisses pour repérer l'intersection
pl.plot(L_abs,[0]*len(L_abs),'k-')
plt.show()
```



Exercice 0.4. *Considérons un parachutiste de masse $m = 70kg$ sautant depuis un point. Au cours de sa chute, il est soumis à son poids ainsi qu'à des frottements fluides qu'on modélisera par une force proportionnelle à la vitesse de chute au carré :$\overrightarrow{F} = -kv^2\overrightarrow{u_z}$(avec un axe z choisi descendant) avec $k = 0,2N.m^{-2}.s^2$.*
On montre alors que l'équation différentielle vérifiée par la vitesse s'écrit :

$$\frac{\mathrm{d}v}{\mathrm{d}t} + \frac{k}{m}v^2 = g$$

1. *Quelle est la fonction $f(t, y)$ correspondant à cette équation différentielle ?*
   Écrire en python la fonction $f(t, y)$ associée. On définira les constantes physiques comme des variables globales placées en début de programme.
2. *Écrire une fonction python* ***euler(f,t0,y0,tmax,h)*** *prenant en arguments :*
   - *la fonction f ;*
   - *les conditions initiales t0 et y0 ;*
   - *la valeur maximale tmax du temps pour lequel on souhaite calculer $y(t)$*

- *le pas h*

et retournant deux listes contenant les valeurs de t_n et y_n pour $t_0 \leq t_n \leq tmax$.

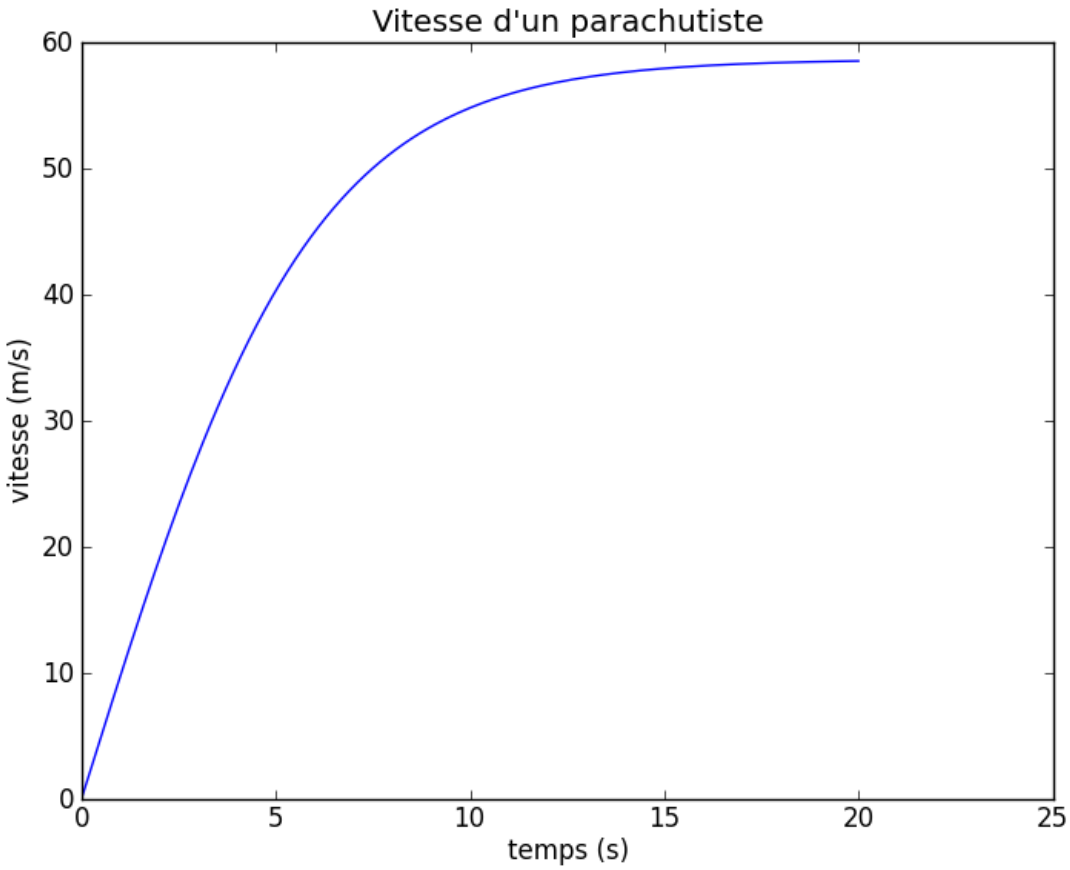
```
# Constantes physiques
k = 0.2
m = 70.
g = 9.81
t0 = 0.
v0 = 0.
tmax = 20.

# Question 1
def f(t,y):
    return (-k/m*y**2.+g)

# Question 2
def euler1(f,t0,y0,tmax,h):
    # Initialisations
    L_abs=[t0]
    L_ord=[y0]
    t=t0
    y=y0
    while t<= tmax:
        t=t+h
        y=y + h*f(t,y)
        L_ord.append(y)
        L_abs.append(t)

    return (L_abs,L_ord)
```

Si on trace la courbe en fonction du temps, on obtient :


Vitesse d'un parachutiste
vitesse (m/s)
temps (s)
0
10
20
30
40
50
60
0
5
10
15
20
25