Institut **P**réparatoire aux **E**tudes d'**I**ngénieurs de **N**abeul
Département : Mathématiques et Informatique
Informatique : TP Python
Niveau : $2^{ème}$ années SP, ST et SM

Corrigé TP 2. Simulation Numérique
Résolution des équations différentielles du preimer ordre

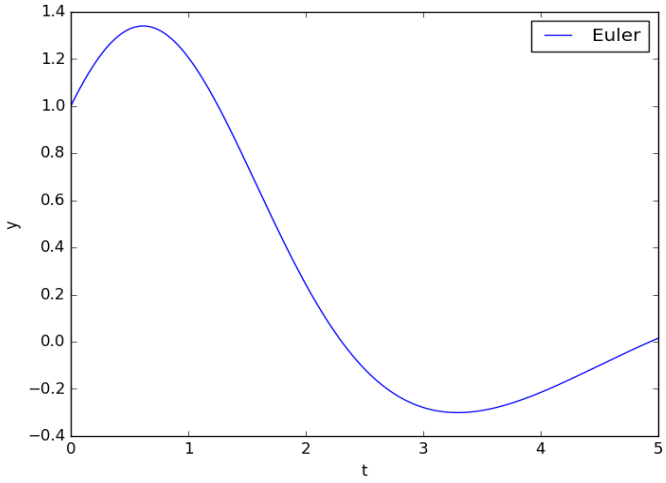
Exercice 1 :

Une équation différentielle d'ordre 1 : On considère le problème de Cauchy :

$$\begin{cases} \dfrac{dy}{dt} = -ty + cos(t) \\ y(0) = 1 \end{cases}$$

Représenter graphiquement la solution de ce problème sur l'intervalle [0, 5].

```
import numpy as np
import math as m
import matplotlib.pyplot as plt
def euler(F, a, b, y0, h=1e-2):
        """Fournit une solution approchée à l'équation :
        y'=F(y,t), y(a)=y0
        sur l'intervalle [a,b] par la méthode d'Euler
        le pas utilisé est h, par défaut 0.01
        La solution renvoyée est une liste de valeurs pour les_t,
        et une liste de valeurs pour les_y
        """
        y = y0
        t = a
        les_y = [y0] # la liste des valeurs renvoyées
        les_t = [a]
        while (t + h <= b):
                #les_y = [y0, . . . ,yk], les_t = [t0, . . . ,tk]
                y += h * F(y, t)
                les_y.append(y)
                t += h
                les_t.append(t)
        return les_t, les_y
def F(y,t):
        return -t * y + m.cos(t)
t,y = euler(F,0,5,1)
plt.plot(t,y,label='Euler')
plt.legend(loc='upper right')
plt.ylabel('y')
plt.xlabel('t')
plt.show()
```



Exercice 2 : Problème de cinétique chimique

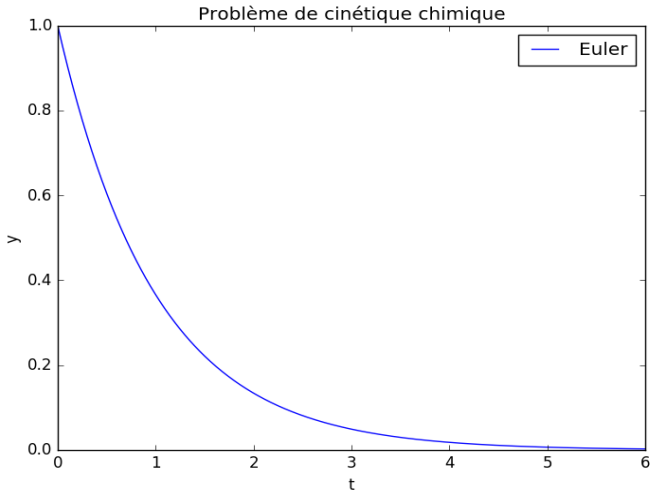
On s'intéresse à une réaction chimique faisant disparaître un réactif A d'une solution.
On suppose que la vitesse de disparition de A est proportionnelle à sa concentration (réaction d'ordre 1) :

$$\frac{d[A]}{dt} = -\alpha[A]$$

où α est une constante s'exprimant en s^{-1} (la valeur $\frac{ln2}{\alpha}$ est un temps appelé temps de demi-réaction).
L'équation est bien de la forme $[A]' = F(t,[A])$ où $F : (t,y) \longmapsto -\alpha y$
On suppose $\alpha = 1s^{-1}$ et au temps $t = 0$, $[A] = 1mol/l$.

Tracer la courbe représentative de l'évolution de la vitesse de disparition de A jusqu'à $t = 6s$.

```
import numpy as np
import math as m
import matplotlib.pyplot as plt
def euler(F, a, b, y0, h=1e-2):
        y = y0; t = a
        les_y = [y0] # la liste des valeurs renvoyées
        les_t = [a]
        while (t + h <= b):
                #les_y = [y0, . . . ,yk], les_t = [t0, . . . ,tk]
                y += h * F(y, t)
                les_y.append(y)
                t += h
                les_t.append(t)
        return les_t, les_y
def F(y,t):
        return -y
t,y = euler(F,0,6,1)
plt.plot(t,y,label='Euler')
plt.legend(loc='upper right')
plt.title('Problème de cinétique chimique')
plt.ylabel('y')
plt.xlabel('t')
plt.show()
```



Exercice 3 : Chute libre d'un corps

On lâche une bille de masse m. Cette bille est soumise à l'action de la pesanteur et à un effort de freinage proportionnel au carré de la vitesse. L'équation de résultante dynamique projetée dans la direction $\vec{e_z}$ conduit à :

$$m\frac{dv}{dt}(t) = mg - \mu m v^2 \qquad \Leftrightarrow \qquad \frac{dv}{dt}(t) = g - \mu v^2$$

où v est la vitesse de la bille, intiallement nulle : $v(0) = 0$.
La vitesse $v(t)$ est alors solution du problème de Cauchy :

$$\begin{cases} \dfrac{dv}{dt} = f(t, v(t)) \\ v(0) = 0 \end{cases}$$

avec f la fonction de deux variables :

$$f : (t, x) \longmapsto g - \mu x^2$$

Initialisez votre code avec la définition des constantes du problème :

```
g=9.81 # accélération de la pesanteur, m/s^2
mu=2.5e-4 # indice de viscosité, 1/m
```

Question 1.
Définir une fonction f qui prend comme arguments une valeur de x et un instant t et qui renvoie la valeur de $f(t, x)$ correspondant au problème.

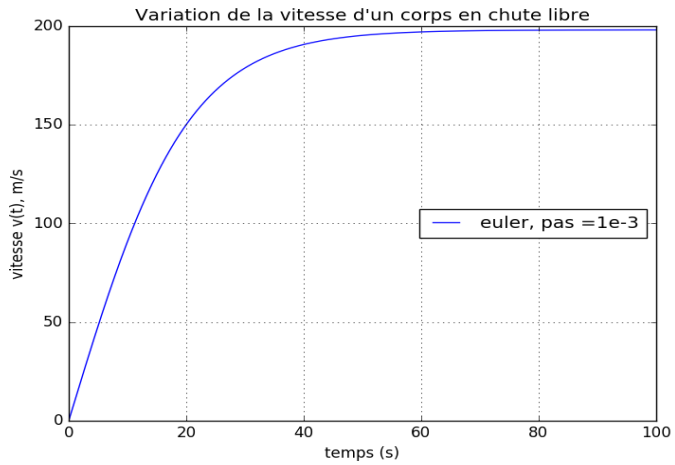
```
#Initialisation des constantes
g = 9.81 #accelération de la pésenteur, m/s²
mu = 2.5e-4 #indice de viscosité, l/m

def F(v,t):
        return g - mu * v**2
```

Question 2.
Écrire une fonction $euler(f, x0, a, b, h)$ qui prend comme arguments une fonction f, une valeur intiale $x0$, une valeur initiale de temps a, une valeur finale de temps b, un pas de temps h et qui renvoie deux listes $liste_t$ et $liste_x$ contenant respectivement les instants $(ti)_{i\geq 0}$ avec $t_i = a + h \times i \leq b$ et les approximations de la solution correspondantes.

```
def euler(F, v0, a, b, h):
        t=a
        liste_t=[t]
        v=v0
        liste_v=[v]
        while t+h<b:
                v=v+h*F(v,t)
                liste_v.append(v)
                t+=h
                liste_t.append(t)
        return liste_t, liste_v
```

Question 3.
Tracer l'évolution de la vitesse de la bille v en fonction du temps, en utilisant la fonction précédente pour avoir une simulation sur l'intervalle de temps $[0, 100]$ (en secondes) et un pas de temps $h = 10^{-3}$s. Que peut-on remarquer pour tout instant $t > 60s$?

```
import numpy as np
import matplotlib.pyplot as plt
t,v = euler(F,0,0,100,1e-3)
plt.plot(t,v, label="euler, pas =1e-3")
plt.legend(loc="center right")
plt.title("Variation de la vitesse d'un corps en chute libre")
plt.xlabel("temps (s)")
plt.ylabel("vitesse v(t), m/s")
plt.grid(True); plt.show()
```



Interprétation :
Le mouvement de la bille comporte deux phases :

- Un régime transitoire, au cours duquel la vitesse augmente d'abord fortement
- et un régime permanent pendant lequel la vitesse diminue de plus en plus faiblement pour atteindre une valeur limite constante à partir de t = 60 s . Donc, on considère qu'au bout de 60 secondes, le régime permanent est atteint.

Question 4.
Vous allez à présent observer l'influence du pas de temps sur la solution calculée.
La solution explicite de l'équation

$$\frac{dv}{dt}(t) = g - \mu v^2 \quad avec \quad v(0) = 0$$

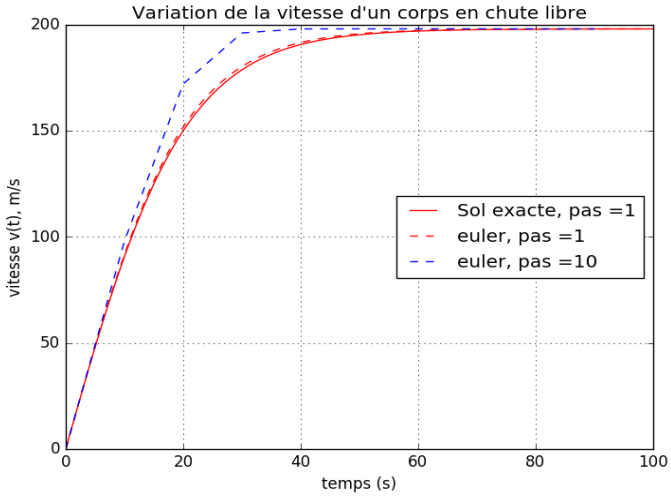
est :

$$v(t) = \sqrt{\frac{g}{\mu}} \times \frac{e^{\frac{2gt}{\sqrt{\frac{g}{\mu}}}} - 1}{e^{\frac{2gt}{\sqrt{\frac{g}{\mu}}}} + 1} \quad pour tout \quad t > 0$$

Tracer sur un même graphe la solution exacte, et les solutions calculées pour les deux pas de temps $h = 1$ et $h = 10$. Observez alors l'influence du pas de temps sur la qualité de la solution.

```
def v(t):
        assert t>0
        a1 = 2*g*t
        a2 =np.sqrt(g/mu)
        a=np.exp(a1/a2)-1
        b1 = 2*g*t
        b2 =np.sqrt(g/mu)
        b=np.exp(b1/b2)+1
        return np.sqrt(g/mu) * (a/b)

#solution exacte, pas = 1
t1=np.arange(0,101,1) # temps =[0,100]secondes; pas : h=1
v1=[0]+[v(t) for t in range(1,101,1)] # v(0)=0; vitesse à t=0
plt.plot(t1,v1,'r-', label="Sol exacte, pas =1")

#Solution calculée, pas = 1
t,v = euler(F,0,0,100,1)
plt.plot(t,v,'r--', label="euler, pas =1")

#solution exacte, pas = 10
t2=np.arange(0,110,10) # temps =[0,100]secondes; pas : h=10
v2=[0]+[v(t) for t in range(10,110,10)] # v(0)=0; vitesse à t=0
plt.plot(t2,v2,'b-', label="Sol exacte, pas =10")

#Solution calculée, pas = 10
t,v = euler(F,0,0,100,10)
plt.plot(t,v,'b--', label="euler, pas =10")
```

```
plt.legend(loc="center right")
plt.title("Variation de la vitesse d'un corps en chute libre")
plt.xlabel("temps (s)")
plt.ylabel("vitesse v(t), m/s")
plt.grid(True)
plt.show()
```



Question 5.
La fonction `odeint` de la bibliothèque scientifique `scipy.integrate` de Python est déjà programmée et permet la résolution d'équation différentielle. Attention, elle s'utilise en donnant une fonction de deux variables de la forme $(x,t) \longmapsto f(x,t)$. Vous pouvez redéfinir facilement une fonction à donner à `odeint` ainsi :

```
def f2(a,b):
        return f(b,a)
```

En lisant l'aide de la fonction `odeint` (`help(odeint)`, après importation), résoudre l'équation différentielle précédente. Tracer sur un même graphe la solution exacte et la solution obtenue par cette fonction. On pourra utiliser `np.linspace` pour générer des tableaux `Numpy` contenant des temps.

```
from scipy.integrate import odeint

#Solution calculée avec 'odeint'
lest=np.linspace(0,100)
lesv=odeint(F,0,lest)
plt.plot(lest,lesv,'g*', label="Sol odeint")

#solution exacte
v1=[0]+[v(t) for t in lest[1:]] # v(0)=0; vitesse à t=0
plt.plot(lest,v1,'r-', label="Sol exacte")

plt.legend(loc="center right")
plt.title("Variation de la vitesse d'un corps en chute libre")
```

```
plt.xlabel("temps (s)")
plt.ylabel("vitesse v(t), m/s")
plt.grid(True)
plt.show()
```