TD 8 -
Tracé avec `pyplot`.
Recherche de racine,
Méthode des moindres carrés
appliqués à la cinétique chimique.

Informatique
MPSI - Lycée Thiers

2014/2015

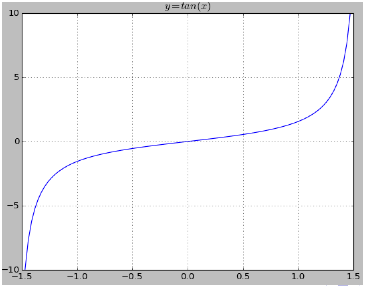
Exercice

Exercice. *Tracer à l'aide de* `matplotlib.pyplot` :

1. La courbe représentative de tan.
2. Les courbes représentatives de ln et de $\sin(x)/x$ pour $x \in]0, 10]$.
3. La courbe paramétrée $x(t) = t.\cos(t), y(t) = t.\sin(t)$ pour $t \in [0, 10]$.
4. L'ellipse de grand axe $[-2, 2] \times \{0\}$ et de petit axe $\{0\} \times [-1, 1]$.

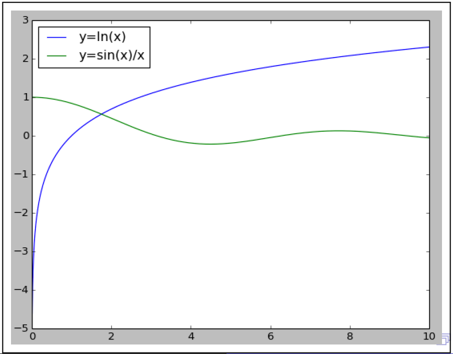
Exercice

```
import numpy as np
import matplotlib.pyplot as plt

# Courbe représentative de tan
ecart = 0.1
X = np.linspace(-np.pi/2 + ecart,np.pi/2 - ecart,100)
Y = np.tan(X)
plt.figure(1)
plt.plot(X,Y)
plt.grid()
plt.title("$y = tan(x)$")
plt.show()
```



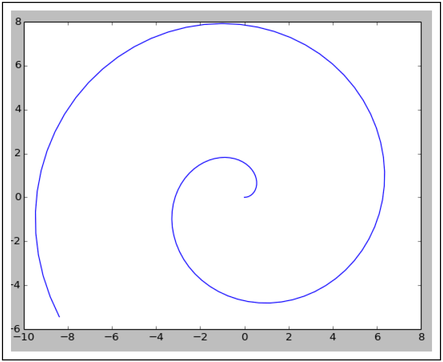
Exercice

```
# Courbes représentatives de ln et sin(x)/x
X = np.linspace(0,10,1000)
X = X[1:]
Y = np.log(X)
plt.figure(2)
plt.plot(X,Y)
Z = np.sin(X)/X
plt.plot(X,Z)
plt.legend(('y=ln(x)', 'y=sin(x)/x'), 'upper left')
plt.show()
```



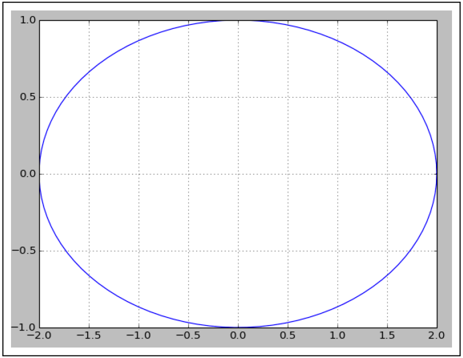
Exercice

```
# Spirale
T = np.linspace(0,10,100)
X = T*np.cos(T)
Y = T*np.sin(T)
plt.figure(3)
plt.plot(X,Y)
plt.show()
```



Exercice

```
# Ellipse
T = np.linspace(0,2*np.pi,100)
X = 2*np.cos(T)
Y = np.sin(T)
plt.figure(4)
plt.plot(X,Y)
plt.grid()
plt.show()
```



Problème

Problème. *Modélisation d'une réaction chimique par la méthode des moindres carrés*

Dans une réaction chimique on souhaite modéliser l'évolution de la concentration d'un réactif en fonction du temps.

On a mesuré expérimentalement :

Temps (s)	0	7	18	27	37	56	102
Concentration	34,83	32,14	28,47	25,74	23,14	18,54	11,04

Dans la suite on notera $(T_i)_{0\leqslant i\leqslant 6}$ la suite des temps considérés et $(C_i)_{0\leqslant i\leqslant 6}$ la suite des concentrations mesurées.

Nous souhaitons effectuer une modélisation de la réaction par une réaction chimique à l'ordre 1, c'est à dire, si $C(t)$ désigne la concentration en fonction du temps :

$$\frac{dC(t)}{dt} = -\lambda \cdot C(t)$$

pour λ une constante réelle.
Elle admet pour solution : $C(t) = C_0 \cdot \exp(-\lambda t)$ où C_0 est une constante réelle.

Problème

1. Justifier que $\lambda > 0$; quelle valeur peut-on prendre pour C_0 ?

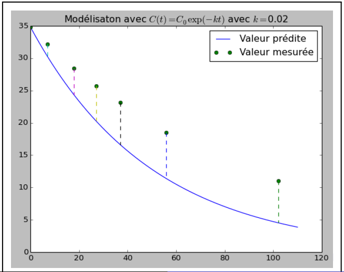
Réponse :

$$C(t) = C_0 \exp(-\lambda t)$$

Pour $t = 0$, $C(0) = C_0 = 34,83$. La fonction exp est strictement croissante. Or la concentration mesurée décroît strictement en fonction du temps. Donc il faut prendre $\lambda > 0$.

• Une fois choisie la constante C_0 on souhaite déterminer, s'il existe, le réel $\lambda > 0$ pour laquelle la solution trouvée approchée le mieux le nuage de points $(T_i, C_i)_{0 \leqslant i \leqslant 6}$ au sens des moindres carrés, c'est à dire le minimum de l'application :

$$m : \lambda \longmapsto \sum_{i=0}^{6} (C_i - C_0 \exp(-\lambda T_i))^2$$



Problème

2. Ecrire le code de cette fonction $m()$ qui avec pour paramètre un nombre λ retourne la valeur de $m(\lambda)$.
3. Tracer le graphe de la fonction m sur plusieurs intervalles bien choisis ; sur quel intervalle I de $\mathbb{R}^*$ la fonction m semble-t-elle présenter un minimum ?

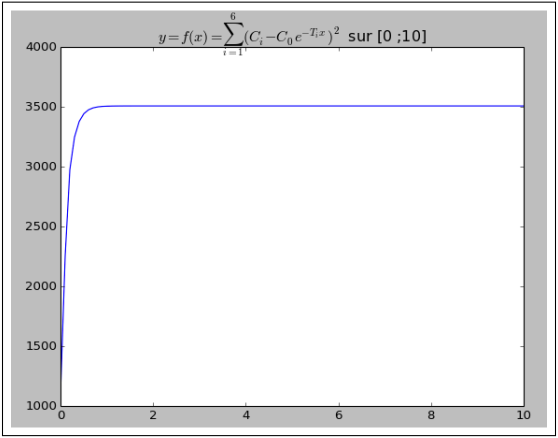
$$m : \lambda \longmapsto \sum_{i=0}^{6} \left(C_i - C_0 \exp(-\lambda T_i)\right)^2$$

```
T = [0, 7, 18, 27, 37, 56, 102]
C = [34.83, 32.14, 28.47, 25.74, 23.14, 18.54, 11.04]

import numpy as np

def m(x):
    S = 0
    for i in range(1,7):
        S += (C[i] - C[0]*np.exp(-x*T[i]))**2
    return S
```

```
import matplotlib.pyplot as plt

A = 10 # 1 0.1
X = np.linspace(0,A,1000)
Y = m(X)
plt.plot(X,Y)
plt.show()
```

Problème

Sur $[0, 10]$ (`A = 10`)

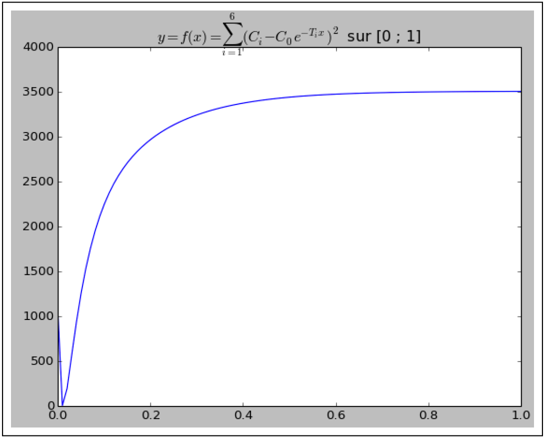


La courbe de la fonction admet l'asymptôte horizontale :

$$y = \sum_{i=1}^{6} C_i^2$$

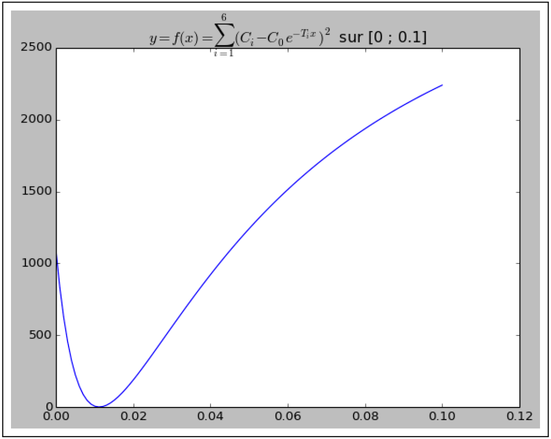
Problème

Sur $[0, 1]$ (`A = 1`)



Problème

Sur $[0, 0, 1]$ (`A = 0,1`)



La fonction semble présenter un minimum entre 0 et $0,02$

Problème

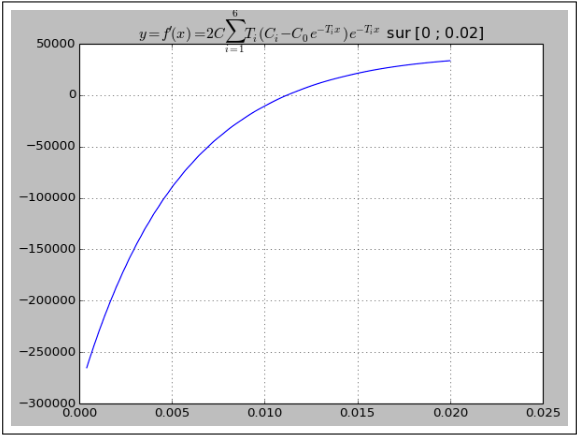
4. Ecrire le code de la fonction dm() qui pour paramètre un nombre λ retourne le nombre dérivée $m'(\lambda)$. Le calcul donne :

$$m'(\lambda) = 2C_0 \sum_{i=0}^{6} T_i \left(C_i - C_0 \exp(-\lambda T_i)\right) \exp(-\lambda T_i)$$

5. Tracer le graphe sur I de la fonction dérivée m'.

```
def dm(x):
    S = 0
    for i in range(7):
        S += T[i] * (C[i]-C[0]*np.exp(-x*T[i]))*np.exp(-x*T[i])
    return 2*C[0] * S


X = np.linspace(0,0.02,100)
Y = dm(X)
plt.plot(X,Y)
plt.grid()
plt.show()
```

Problème



La fonction dérivée m' admet une racine entre $0,01$ et $0,015$.

Problème

6. Proposer une méthode pour déterminer une valeur approchée du minimum λ_{min} de m et l'implémenter. Si besoin, le calcul donne :

$$m''(\lambda) = 2C_0 \sum_{i=0}^{6} T_i^2 \left(2C_0 \exp(-\lambda T_i) - C_i\right) \exp(-\lambda T_i)$$

7. Quelle est la valeur de m obtenue pour cette valeur λ_{min} de λ ?
8. Pour cette valeur effectuer le tracé sur un même graphique du nuage de points et de la solution théorique obtenue. Quel est leur écart au sens des moindres carrés ?
9. L'hypothèse d'une réaction chimique à l'ordre 1 est-elle vraisemblable ?

6. On recherche une racine de la dérivée m'. On applique la méthode de Newton,

$$u_{n+1} = u_n - \frac{f(u_n)}{f'(u_n)}$$

Pour cela il faut calculer la dérivée de m', c'est à dire la dérivée seconde m''.

Il faudra aussi prendre garde que la suite (u_n) converge en cherchant "à tatons" la valeur de u_0 à choisir qui soit suffisamment proche de la racine recherchée.

Problème

```
def ddm(x):
    S = 0
    for i in range(7):
        S += T[i]**2 * (2*C[0]*np.exp(-x*T[i])-C[i])*np.exp(-x*T[i])
    return 2*C[0] * S

def newton(f,g,u,n):
    for k in range(n):
        u = u-f(u)/g(u)
    print("m' = ", dm(u))
    return u

def solution(u,n):
    return newton(dm,ddm,u,n)
```

Problème

- Utilisation :

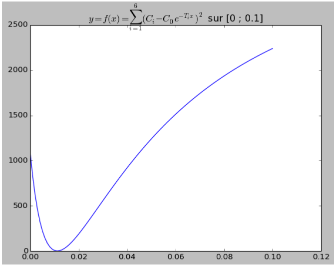
```
In [1]: solutionNewton(1,10)
m' = 0.000647849931581
Out[1]: 2.4287841721441406
In [2]: solutionNewton(1,100)
m' = 5.30849037946e-43
Out[2]: 15.285927039409478
```

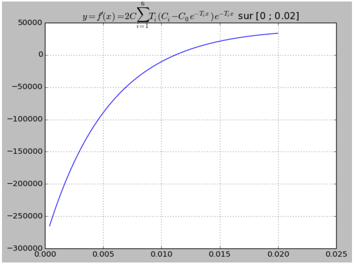
La méthode de Newton diverge pour le point initial : $u_0 = 1$. Ainsi que pour $u_0 = 0.1$:

```
In [3]: solutionNewton(0.1,10)
m' = 0.868069492978
Out[4]: 1.4001516737694022
In [4]: solutionNewton(0.1,100)
m' = 7.11272712686e-40
Out[4]: 14.25730796594908
```

Problème

Sur $[0, 0.1]$ (`A = 0.1`)





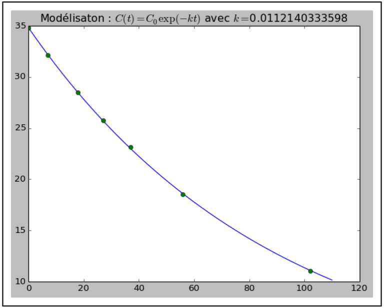
• Obtention de la solution à l'aide de la méthode de Newton : convergence pour $u_0 = 0.02$:

```
In [5]: solutionNewton(0.02,10)
m' = 1.29000010674e-11
Out[5]: 0.011214033359765131

In [6]: solutionNewton(0.02,100)
m' = 1.29000010674e-11
Out[6]: 0.011214033359765131
```

• Solution trouvée : $\boxed{\lambda_{min} = 0,011214033359765131}$ (toutes les décimales semblent exactes).

Problème

```
u = solutionNewton(0.02,10)
X = np.linspace(0,110,1000)    # Subdivision régulière de [0,110]
Y = C[0] * np.exp(-u*X)
plt.figure(2)
ch = "Modélisaton : $C(t) = C_0 \exp(-kt)$ avec $k =$" + str(u)
plt.title(ch)
plt.plot(X,Y,'b',T,C,'o')    # Tracés
plt.show()
```

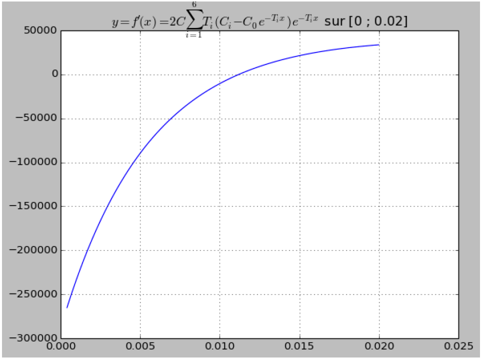



```
In [7]: dm(u), m(u)
Out[7]: (1.2900001067350785e-11, 0.02843347519783374)
```

Problème

- Recherche par dichotomie :

En s'aidant du graphe de la dérivée m', on peut appliquer une dichotomie sur $[0.01; 0.02]$ car m' y change de signe :



Problème

```
def dichotomie(f,a,b,tol):
    assert f(a)*f(b) < 0
    m = (a+b)/2
    if b-a < tol:
        print("m' = " + str(dm(m)))
        return m
    if f(a)*f(m) < 0:
        return dichotomie(f,a,m,tol)
    else:
        return dichotomie(f,m,b,tol)

def solutionDich(a,b,tol):
    return dichotomie(dm,ddm,a,b,tol)
```

```
In [2]: solutionDich(0.01,0.02,1e-10)
m' = -0.000247138216798
Out[2]: 0.011214033327996732

In [3]: solutionDich(0.01,0.02,1e-15)
m' = -2.21156133406e-10
Out[3]: 0.011214033359765101
```

• Remarquons que les 15 premiers chiffres après la virgule obtenus par la méthode de Newton : $\lambda_{min} = 0,\underline{011214033359765}131$ sont tous exacts après 10 itérations (ainsi que les suivants)...