Résolution numérique d'équations

I Capacité numérique :

déterminer, à l'aide d'un langage de programmation, l'état final d'un système, siège d'une transformation, modélisée par une réaction à partir des conditions initiales et valeur de la constante d'équilibre.

II Fonction fsolve

La fonction `fsolve` du module `scipy` permet d'obtenir des solutions numériques approchées d'équations ou de systèmes d'équations.

Elle prend pour cela pour argument :

- une expression `func` d'une variable `x`
- une estimation initiale de `x` (à choisir dans l'intervalle où on pense qu'une solution existe)

Dans l'exemple suivant, on recherche la solution de l'équation :

$$x^5 = 10^{-2}\sqrt{2-x}$$

avec comme estimation initiale $x = 0{,}2$.

```
%matplotlib notebook
```

La ligne précédente ne doit apparaître que dans les notebooks `Jupyter`, pas dans un fichier python.

```
import numpy as np
from scipy.optimize import fsolve

def equation(x):
    return x**5 - 1e-2*np.sqrt(2-x)

root = fsolve(equation,.2)
print(f'La solution est : {root}')
```

On peut également lui faire résoudre un système d'équations en utilisant comme argument de `equation` un tableau (voir la documentation officielle).

III Questions du DM04

III.1 Exo 1, question 3a

L'équation à résoudre est ici :
\[

$$\frac{\tau^{5/2}}{(1-\tau)^3/2} = \frac{\sqrt{K_3}}{K_1 c}$$

\]

On calcule alors, en utilisant comme valeur initiale celle obtenue par l'approximation de la question 2.b.

```
c = 1e-2 # en mol.L
K1 = 10**(8.3)
K3 = 10**(15.2)

def equation3(x):
    return x**5 - np.sqrt(K3)/(c*K1)*(1-x)**(3/2)

root = fsolve(equation3,.87)
print(f'Pour c = {c}, la solution est du 3a est: tau={root}')
```

III.2 Exo 1, question 3b

Il suffit de changer la valeur de la concentration c

```
c = 1e-1 # en mol.L
root = fsolve(equation3,.87)
print(f'Pour c = {c}, la solution est du 3a est: tau={root}')
```