# DS n° 03

Faites les deux exercices sur deux feuilles séparées.

## Exercice 1 – Résolution d'équations

Soit $f$ une fonction donnée. On cherche à calculer une approximation d'une solution de l'équation : $f(x) = 0$.

1. Avec la méthode de la dichotomie :
   Écrire une fonction `dicho` qui prend comme entrée la fonction $f$ à étudier, les bornes initiales $a$ et $b$, la précision $\epsilon$ et qui renvoie $\frac{a_n + b_n}{2}$, approximation d'une solution à $\epsilon$ près.

   **Solution 1.**

```python
def dicho(f,a,b,eps):
    while (b-a)>2*eps:
        c=(float(a)+b)/2
        if f(a)*f(c)<0:
            b=c
        else:
            a=c
    return((a+b)/2.)
```

2. Avec la méthode de Newton :
   Écrire une fonction `Newton` qui prend comme entrée la fonction à étudier, sa dérivée, une valeur initiale de la suite $x_0$, la précision $\epsilon$ et qui renvoie $x_n$ approximation de la solution.

   **Solution 2.**

```python
def Newton(f,fprime,x0,epsilon):
    x=float(x0)
    compteur=0
    # l ecart entre xn et x(n+1) est f(xn)/fprime(xn)
    while abs(f(x)/fprime(x))>epsilon and compteur<1000 :
        x=x-f(x)/fprime(x)
        compteur=compteur+1
    return(u)
```

3. Donner un avantage et un inconvénient de la méthode de Newton par rapport à la méthode de la dichotomie.

   **Solution 3.** La méthode de Newton converge plus rapidement que la dichotomie. Mais elle ne converge pas forcément. Alors que si $f(a)$ et $f(b)$ sont de signes opposés, la dichotomie converge nécessairement vers l'une des solutions.

## Exercice 2 – Approximation d'une intégrale

Soit $f$ une fonction donnée. On souhaite approximer la valeur de l'intégrale $\int_a^b f$ avec différentes méthodes.

### .1 Cas où f est connue

Dans cette partie, on travaille avec une fonction $f$ connue sur tout l'intervalle $[a, b]$. On subdivise l'intervalle $[a, b]$ en $n$ intervalles du type $[x_i, x_{i+1}]$, de même longueur avec :
$a = x_0 < x_1 < \cdots < x_n = b$.

1. Avec la méthode des rectangles :
   sur chaque intervalle $[x_i, x_{i+1}]$, on approxime $f$ par une constante.

   Coder un algorithme qui calcule une approximation de $\int_0^{\frac{3\pi}{2}} \cos(x)dx$ avec 2019 intervalles.

   **Solution 4.**

```
x=linspace(0,3*pi/2,2020)
integrale=0
for i in range(2019):
   integrale=integrale+cos(x[i])*(x[i+1]-x[i])
print integrale
```

2. Avec la méthode de Simpson :
sur chaque intervalle $[x_i, x_{i+1}]$, on approxime $f$ par un polynôme de degré deux. Dans ce cas, l'intégrale est approximée par :

$$\int_a^b f(x)dx \approx \frac{b-a}{6n}\left(\sum_{i=0}^{n-1} f(x_i) + 4f\left(\frac{x_i+x_{i+1}}{2}\right) + f(x_{i+1})\right)$$

Ecrire un programme qui calcule une approximation de $\int_a^b f$ par la méthode de Simpson.
(On supposera que `f`, `n`, `a` et `b` sont déjà définis dans le programme.)

**Solution 5.**

```
x=linspace(a,b,n+1)
integrale=0
for i in range(n):
   integrale=integrale+(b-a)*1./(6*n)*(f(x[i])+4*f((x[i+1]+x[i])/2.)+f(x[i+1]))
print integrale
```

## .2 Avec lecture de fichiers

Dans cette partie, la fonction $f$ n'est connue qu'en certains points, dont les coordonnées sont situées dans un fichier. Ce fichier `coordonnee.csv`, situé dans le répertoire de travail, contient une quinzaine de lignes selon le modèle suivant :

$$0.0; 1.00988282142$$

$$0.1; 1.0722126449$$

Chaque ligne contient deux valeurs flottantes séparées par un point-virgule, représentant respectivement l'abscisse et l'ordonnée d'un point. Les points sont ordonnés par abscisses croissantes.

1. Rappeler quels sont les différents modes d'ouverture d'un fichier texte à l'aide de Python, et les différences entre ces modes.

   **Solution 6.** Il est possible d'ouvrir un fichier dans plusieurs *modes* : lire (`'r'`), ajouter à la fin (`'a'`), écrire (`'w'`).

2. Écrire une séquence d'instructions en Python permettant d'effectuer les opérations suivantes sur le fichier `coordonnee.csv` :
Ouvrir le fichier en lecture, le lire et construire deux listes de flottants : la liste `LX` des abscisses et la liste `LY` des ordonnées contenues dans ce fichier.

   **Solution 7.**

```
fichier=open('coordonnee.csv','r')

LX=[]
LY=[]

for ligne in fichier.readlines():
    LX.append(float(ligne.strip().split(";")[0]))
    LY.append(float(ligne.strip().split(";")[1]))
```

3. Avec la méthode des trapèzes :
sur chaque intervalle, on approxime $f$ par un polynôme de degré 1.
Ecrire une fonction `trapeze` qui prend comme entrée deux listes de taille $n+1$ : $(x_0, \cdots x_n)$ et $(y_0, \cdots, y_n)$ et qui renvoie :

$$\sum_{i=0}^{n-1}(x_{i+1}-x_i)\frac{y_{i+1}+y_i}{2}$$

**Solution 8.**

```
def trapeze(LX,LY):
    integrale=0
    for i in range(len(LX)-1) :
        integrale=integrale+(LX[i+1]-LX[i])*(Y[i+1]+Y[i])*1./2
    return integrale
```

4. Valeur moyenne et écart-type :
Par définition, la valeur moyenne et l'écart-type d'une fonction $f$ sur $[x_0, x_n]$ sont donnés par :
— valeur moyenne : $M_f = \dfrac{1}{x_n - x_0}\displaystyle\int_{x_0}^{x_n} f(x)dx$
— écart-type : $\sigma_f = \sqrt{\dfrac{1}{x_n - x_0}\displaystyle\int_{x_0}^{x_n} (f(x) - M_f)^2 dx}$
Ecrire un programme qui calcule une valeur approximative de $\sigma_f$. (On pourra faire appel à la fonction `trapeze`)

**Solution 9.**

```
Mf=1./(LX[-1]-LX[0])*trapeze(LX,LY)

LY_sigma=[]
for y in LY:
        LY_sigma.append((y-Mf)**2)

variance=1./(LX[-1]-LX[0])*trapeze(LX,LY_sigma)
sigma=sqrt(variance)
```

Méthode 2 :

```
Mf=1./(LX[-1]-LX[0])*trapeze(LX,LY)

LY_sigma=(LY-Mf)**2

variance=1./(LX[-1]-LX[0])*trapeze(LX,LY_sigma)
sigma=sqrt(variance)
```