Séance 19 :

Dérivée $n^{ième}$: méthode de Simpson

1. Valeur approchée

Sachant que les alculs en Python se font à 10^{-16} ***près, prouver que pour calculer*** I ***à*** 10^{-12} ***près, il suffit de calculer à*** 10^{-12} ***près :***

$$J = \int_0^6 exp(-x^2)sin(x^3)dx$$

Calculons $|I - J|$:

$$\begin{array}{rrcl} & I - J & = & \int_6^{+\infty} exp(-x^2)sin(x^3)dx \\ \Rightarrow & |I - J| & \leq & \int_6^{+\infty} exp(-x^2)|sin(x^3)|dx \\ \Rightarrow & |I - J| & \leq & \int_6^{+\infty} exp(-x^2)\frac{2x}{12}dx \\ \Rightarrow & |I - J| & \leq & \frac{e^{-36}}{12} \\ \Rightarrow & |I - J| & \leq & 2.10^{-17} \end{array}$$

Ainsi, à 10^{-12} près, I et J sont égaux.

2. Dérivée $k^{ième}$

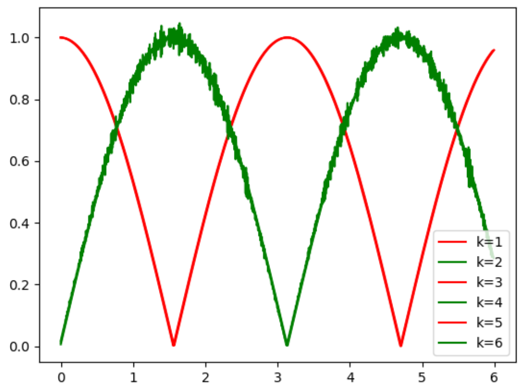
Soit `x` une liste d'abscisses de a à b inclus avec un `pas` constant. On pose $y = f(x)$.
Par récursivité, construire un fonction `deriveekieme(y, x, k)` ***qui calcule la dérivée*** $k^{ième}$ ***de*** `y` ***par rapport à*** `x`***. Elle retournera une liste de longueur*** `len(x)-k`***.***

```
> main.py
def deriveekieme(Y, X, k):
    if k == 0:
        return Y
    else:
        Z = [(Y[i+1]-Y[i])/(X[i+1]-X[i]) for i in range(len(Y)-1)]
        return deriveekieme(Z, X, k-1)
```

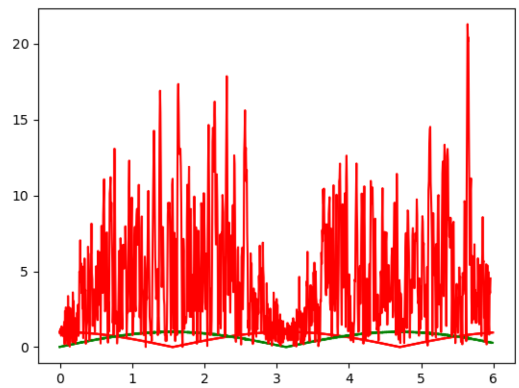
(a) On pose $f(x) = sin(x)$, $a = 0$, $b = 6$ et $n = 1000\times$`pas`.
Tracer les graphes en valeur absolue des dérivées $k^{ième}$ ***de*** f ***pour*** $k \in [\![1, 6]\!]$***.***

```
> main.py
def f(x):
    return np.sin(x)

def TraceDerivk(f, a, b, n, k):
    X = np.linspace(a, b, n+1)
    yk = deriveekieme(f(X), X, k)
    c = 'g' if k % 2 == 0 else 'r'
    plt.plot(X[:-k], np.abs(yk), color=c, label=f'k={k}')

def TraceDeriv(f, a, b, n):
    for k in range(1, 7):
        TraceDerivk(f, a, b, n, k)
    plt.legend()
    plt.show()
```

`TraceDeriv(f, 0, 6, 1000)` donne :



Peut-on le faire pour $k = 7$ ***?***



Oui, mais ça ne donne pas la dérivée $7^{ième}$!

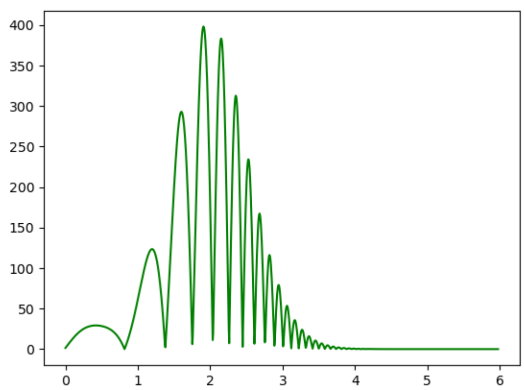
(b) ***Que faut-il contrôler pour que la fonction*** `deriveekieme` ***donne un résultat correct ?***
Il faut que l'odre de la dérivée ne soit pas trop grand...

3. Dérivée $4^{ième}$ de I

Tracer le graphe de la dérivée $4^{ième}$ ***de*** $f(x) = e^{-x^2} sin(x^3)$ ***sur*** $\mathbb{R}$ ***en valeur absolue et donner le maximum*** M***.***
Montrer que $0 \leq M \leq 400$

```
> main.py
```

```python
def g(x):
    return np.exp(-x**2)*np.sin(x**3)

TraceDerivk(g, 0, 6, 1000, 4)
plt.show()
X = np.linspace(0, 6, 1001)
print(max(deriveekieme(g(X), X, 4)))
```



Et $M =$ `398.1138002274256`

4. Methode de Simpson

(a) ***Donner une fonction*** `simpson(f,n,a,b,M)` ***qui, lorsque qu'on divise l'intervalle*** $[a; b]$ ***en*** n `pas` ***renvoie dans un tuple le calcul de*** $\int_a^b f(t)dt$ ***par la méthode de simpson ainsi que l'erreur du calcul.***

> main.py

```python
def simpson(f, n, a, b, M):
    pas = (b-a)/n
    S = 0
    for k in range(n):
        S += pas*(f(a+k*pas)+4*f(a+(k+1/2)*pas)+f(a+(k+1)*pas))/6
    return int(S*10**16)/10**16, (b-a)*pas**4*M/2880
```

(b) Donner alors J pour $n = 182$ et vérifier que l'erreur est inférieure à 10^{-6}.
`print(simpson(g, 182, 0, 6, M))` donne

$$\begin{cases} J = 0.2001380266856075 \\ \varepsilon = 9.796827123004573\text{e-}07 \end{cases}$$

(c) Donner une valeur optimale de n pour que l'erreur soit inférieure à 10^{-12}.

> main.py

```python
def optim(M, p):
    x = 6*(6*M/2880)**(1/4)*10**(p/4)
    return int(x)+1
```

`print(optim(M, 12))` donne $n =$ `5726`
`print(simpson(g, 5726, 0, 6, M))` donne en effet

$$\begin{cases} J = 0.2001380291480081 \\ \varepsilon = 9.999208440734312\text{e-}13 \end{cases}$$

5. Meilleur que les meilleurs ?

Montrer que notre fonction `simpson` ***donne le calcul de*** J ***un meilleur résultat que la fonction*** `quad` ***de la bibliothèque*** `scipy.integrate` ***de python. Comment cela est-il possible d'après vous ? Quelles peuvent-être les limites de la méthode de Simpson ?***

> main.py

```python
from scipy.integrate import quad

res, err = quad(g, 0, 6)
print(res, err)
```

Ce qui donne :

$$\begin{cases} J = 0.20013802915686038 \\ \varepsilon = 9.167238679541269\text{e-}10 \end{cases}$$

Ainsi, la méthode Simpson est plus précise que la méthode de résolution de scipy (qui est précise à 10^{-9} près). La limite de la méthode de Simpson est une précision de 10^{-12} max.