Solutions TP 5

Exercice 1 : *Dérivée d'une fonction*

La définition standard de la dérivée, celle que vous pouvez trouver dans les livres de mathématiques, est la suivante :

$$\frac{df}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

La méthode de base pour le calcul des dérivées numériques correspond à une implémentation de cette formule. Dans la pratique nous ne pouvons pas considérer la limite $h \to 0$, mais nous pouvons prendre h très petit, puis calculer :

$$\frac{df}{dx} \approx \frac{f(x+h) - f(x)}{h}$$

Cette approximation de la dérivée est appelée *forward approximation*, car elle est évaluée dans la direction des x croissantes, autour d'un point x. Il correspond à la pente de la courbe $f(x)$ mesurée sur un intervalle de largeur h dans la directions des x croissants. Il existe également une approximation appelée *backward approximation*, qui est définie de la façon suivante :

$$\frac{df}{dx} \approx \frac{f(x) - f(x-h)}{h}$$

Les deux approximations ne sont pas très efficaces pour des raisons différentes. Une amélioration simple pour calculer la dérivée d'une fonction consiste à utiliser une méthode plus précise appelée *central differences approximation* :

$$\frac{df}{dx} \approx \frac{f(x+h/2) - f(x-h/2)}{h}$$

Cette méthode est similaire aux deux autres, car la dérivée est calculée en utilisant la différence entre deux valeurs de $f(x)$ à une distance h. Ce qui a changé est que les deux points sont placés symétriquement autour de x, un à une distance $1/2h$ dans la direction des x positifs et l'autre à une distance $1/2h$ dans la direction des x négatifs. L'objectif de cet exercice est de créer une fonction $f(x)$ tel que :

$$f(x) = 1 + \frac{1}{2} \cdot tanh(2x)$$

puis d'utiliser la central differences approximation pour calculer la dérivée de la fonction dans l'intervalle $-2 \leqslant x \geqslant 2$. Calculez aussi analytiquement la dérivée et reportez dans un graphique votre résultat numérique et le résultat analytique. Il peut être utile de tracer la fonction analytique avec une ligne continue et l'approximation numérique avec des cercles. **Conseil : en Python la fonction** *tanh* **se trouve dans le paquet math, et elle s'appelle simplement tanh**.

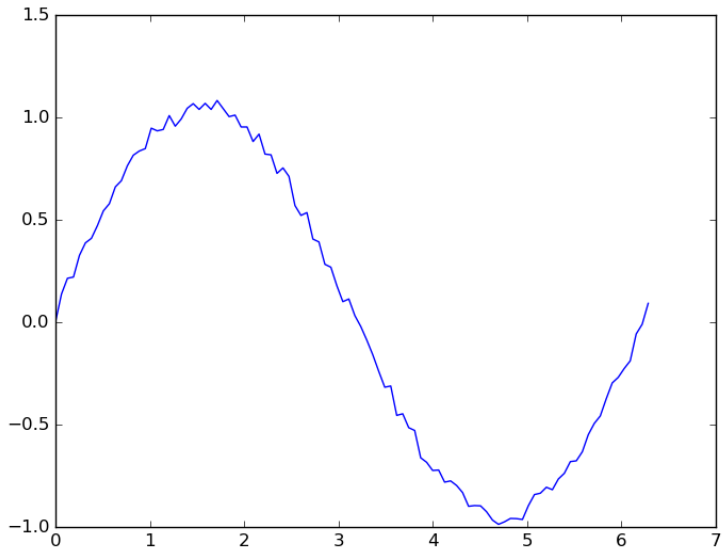
Solution 1:

```
from __future__ import division
from numpy import arange,empty,linspace
from pylab import plot,show
import math
import numpy as np
```

```
h=1e-10
def f(x):
    return 1+0.5*np.tanh(2*x)

x=linspace(-2,2,100)
N=len(x)
y_der=empty(len(x),float)

#Central differences
for i in range(len(x)):
    y_der[i]=(f(x[i]+h/2) - f(x[i]-h/2))*1/h

def f1(x):
    return 1 - (np.tanh(2*x))**2

plot(x,f1(x),"k")
plot(x,y_der,"ro")
show()
```

Exercice 2 : *Dérivée d'une fonction avec du bruit*

Supposons que nous avons des mesures d'une quantité qui, lorsqu'elle est représentée sur un graphique, ressemble à la figure suivante :



Ceci peut être par exemple le résultat d'une expérience en laboratoire. La forme générale de la courbe est claire à partir de la figure, mais il y a du bruit dans les données, de sorte que la courbe n'est pas complètement régulière.

L'objective de cet exercice est de calculer la dérivée première de cette courbe. Voici les étapes à suivre :

— Écrivez un programme pour calculer la dérivée de ces données, en utilisant la *forward approximation* (voir la page 1 du TP).
— Calculez à nouveau la dérivée, mais en utilisant la formule suivante :

$$f'(x) = \frac{-f(x+2h) + 8f(x+h) - 8f(x-h) + f(x-2h)}{12h}$$

— Comparez et commentez les résultats que vous avez obtenus avec les deux méthodes.

Conseil : en Python vous pouvez créer et reporter dans un graphique un ensemble des données comme reporté dans la figure en utilisant les lignes de code suivantes :

```
x = np.linspace(0,2*np.pi,100)
y = np.sin(x) + 0.1*np.random.random(size=x.shape)
plot(x,y,"b",label='noisy data')
```

Solution 2:

```
#This program calculates the derivative of noisy data
#with two different methods: 2-points central differences and
#4-points central differences
import numpy as np
from pylab import *

#Definition of the noisy function data
x = np.linspace(0,2*np.pi,100)
y = np.sin(x) + 0.1*np.random.random(size=x.shape)
plot(x,y,"b",label='noisy data')#plot of the function

#Calculation of the derivative of this function
dy_2p=empty(len(x),float)
#2 points formula, forward differences
for i in range(len(y)-1):
    dy_2p[i]=(y[i+1]-y[i])/(x[i+1]-x[i])
#for the last element of the y array we use a backward approximation
dy_2p[-1] = (y[-1] - y[-2])/(x[-1] - x[-2])
#Single element indexing accepts negative indices
#for indexing from the end of the array.

#Calculation of the derivative with a four point central difference
dy_4p=empty(len(x),float)
h=x[1]-x[0]
dy_4p[2:-2] = (y[0:-4] - 8*y[1:-3] + 8*y[3:-1] - y[4:])/(12.*h)

#For the last elements of the y array we use a low order approximation
dy_4p[0] = (y[1]-y[0])/(x[1]-x[0])
dy_4p[1] = (y[2]-y[1])/(x[2]-x[1])
dy_4p[-2] = (y[-2] - y[-3])/(x[-2] - x[-3])
dy_4p[-1] = (y[-1] - y[-2])/(x[-1] - x[-2])

#Definition of the analytical derivative
def f(x):
    return cos(x)

plot(x,f(x),"k",label='analytical derivative')
plot(x,dy_2p,"g--",label='2pt-forward diff')
plot(x,dy_4p,"r--",label='4pt-centered diff')
legend(loc='lower left')
show()
```

Exercice 3 : *Capacité thermique d'un solide*

Selon la théorie des solides de Debye, la capacité thermique d'un solide à la température T peut s'exprimer comme :

$$C_V(T) = 9V\rho k_B \left(\frac{T}{\theta_D}\right)^3 \int_0^{\theta_D/T} \frac{x^4 e^x}{(e^x-1)^2} dx$$

où V est le volume du solide, ρ est la densité en numéro d'atomes par unité de volume, k_b est la constante de Boltzmann et θ_D est la température de Debye, une propriété des solides qui dépende de leurs densité et de la vitesse du son. L'intégrale qui apparaît dans l'expression de C_V ne peut pas être résolu analytiquement et un traitement numérique s'impose.

Plusieurs méthodes de résolution numérique d'intégrales existent, dans cet exercice vous utiliserez une simple somme de Riemann. Pour une intégrale du type

$$I(a,b) = \int_a^b f(x)dx$$

le domaine d'intégration [a,b] est divisé en un nombre d'intervalles N de largeur $\Delta x = (a-b)/N$. En considérant la fonction $f(x)$ dans le point de milieu de l'intervalle k, l'intégrale peut être approximé par la somme :

$$I(a,b) \simeq \sum_{k=0}^{N-1} \Delta x \cdot f[a + \Delta x(k+1/2)]$$

La figure suivante donne une représentation graphique de cette méthode.

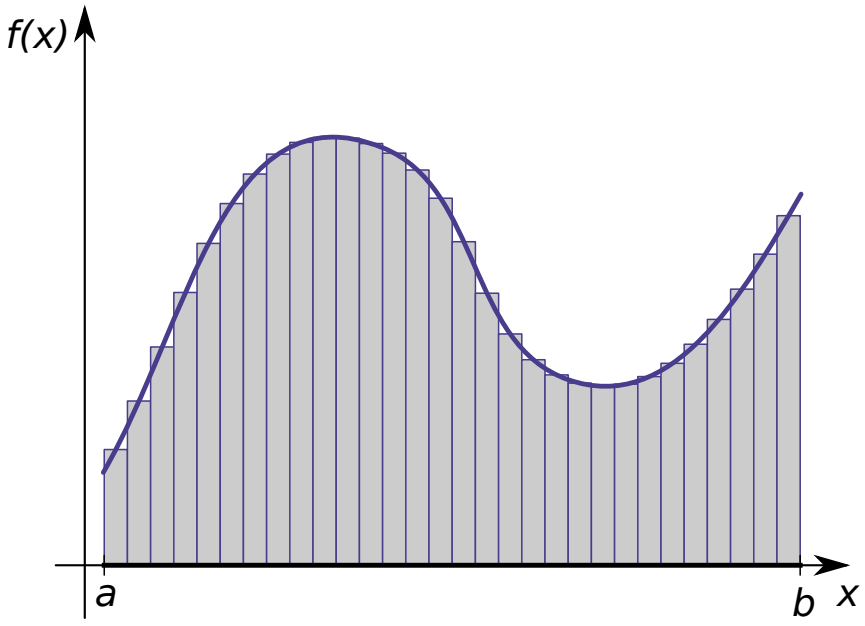


FIGURE 1 – Integrale de Riemann

Dans cet exercice vous allez considérer un cristal d'aluminium de 1000 cm^3. La densité de l'aluminium est $\rho = 6.022 \times 10^{28}$ m^{-3} et sa température de Debye $\theta_D = 428$ K. On vous rappelle que $k_B = 1.30 \times 10^{-23}$ m^{2}Kg s^{-2}K^{-1}. Pour la première partie de cet exercice la température du systéme est constante et égale à T=300 K.

1. Définissez une fonction :

$$f(x) = \frac{x^4 e^x}{(e^x - 1)^2}$$

2. Définissez la fonction `cv(T,N)` qui calcule la capacité thermique en fonction de la température T et du nombre d'intervalles d'intégration N.
3. La précision d'un intégrale numérique I depende du nombre d'intervalles d'intégration N et elle peut être définie comme :

$$\varepsilon = \left| \frac{I(N) - I(N-1)}{I(N)} \right|$$

Trouvez le nombre de points N nécessaires pour avoir une précision sur le calcul de la fonction `cv(T,N)` égale à 10^{-4}, 10^{-6} et 10^{-8}. Pour faire ceci vous pouvez écrire un bloc d'instructions qui utilise l'instruction `while` et une condition sur la précision. A l'intérieur du bloc vous augmentez progressivement le nombre de points d'intégration utilisés pour la fonction `cv(N,T)`.

Exemple :

```
while precision >= 1E-2:
    N += 1
    [...]
```

4. Vous fixez maintenant le nombre d'intervalles d'intégration au valeur qui vous donne une précision de 10^{-6} à $T = 300$ K et vous faites varier T. Calculez la capacité thermique dans la plage de température 10-500 K par pas de 10 K.
5. Tracez la courbe de $C_V(T)$.

Solution 3:

```
from numpy import linspace
from math import exp, sqrt, fabs,log
from pylab import plot,show,xlabel,ylabel

V = 1E-3
rho = 6.022E28
debye = 428.0
kB = 1.38E-23
N=75

def f(x):
    return x**4*exp(x)/(exp(x)-1)**2

def cv(T,N):
    prefactor = 9*V*rho*kB*(T/debye)**3
    delta_x=(debye/T)/N
    integrale = 0
    for k in range(N):
        integrale += delta_x*f(delta_x*(k+0.5))
    return prefactor*integrale

capacita=[]
temperature=[]

for T in range(10,500,10):
    capacita.append(cv(T,N))
    temperature.append(T)

plot(temperature,capacita)
xlabel("Temperature")
```

```
ylabel("Cv")
show()


N=int()
cvold=float()
precision=1

while precision >= 1E-6:
    N += 1
    precision = (cv(T,N)-cvold)/cv(T,N)
    cvold = cv(T,N)

print (N, cv(T,N),precision)
```