TD 1 : La bibliothèque numpy - les tableaux numpy

Exercice 1 : Traitement photo

Les images au format JPG sont constituées de pixels et chaque pixel est associé à un triplet (R,V,B) de nombres compris entre 0 et 255 (uint8) avec un nombre pour chaque couleur primaire (Rouge, Vert et Bleu). Le triplet (0,0,0) correspond à un pixel noir alors que le triplet (255,255,255) correspond à un pixel blanc.

La séquence Python ci-dessous indique comment récupérer le tableau tri-dimensionnel d'une image « pomme.jpg » puis comment créer une image à partir d'un tableau numpy.

```
import numpy as np
from PIL import Image

"""création d'un tableau numpy associée à une image PNG"""
photo=Image.open("pomme.jpg")
tab=np.array(photo)#tableau tridimensionnel (en uint8 par défaut)

"""création d'une image PNG à partir d'un tableau numpy"""
photo2=Image.fromarray(tableau)
photo2.save("photo_nouvelle.jpg") # pour l'enregistrer en .jpg dans le dossier de travail
```

1) Ecrire, sans utiliser de boucle for, une fonction image_rouge prenant comme argument le tableau de valeurs tab et modifiant ce tableau en ne conservant que la composante rouge de chaque pixel de l'image 'pomme.jpg' (cette image est accessible dans le dossier 1_chapitre1).
2) Afficher la nouvelle image et la commenter.

Exercice 2 : Intégration, erreur de quadrature, erreur d'arrondi

Soit une fonction $Y(X)$ échantillonnée. Les échantillons sont notés $Y_i(X_i)$. (X,Y) sont deux tableaux de dimension 1 contenant les valeurs X_i et Y_i.

1) Ecrire la fonction integration1(X,Y) renvoyant l'intégration de $Y(X)$ en utilisant la méthode des rectangles à gauche et sans aucune boucle for !
2) Ecrire la fonction integration2(X,Y) renvoyant l'intégration de $Y(X)$ en utilisant la méthode des trapèzes mais sans aucune boucle for !

On cherche à tester les deux fonctions précédentes afin d'évaluer $\int_0^{10} XdX$. Pour cela, on génère un tableau de N valeurs de X à l'aide de la fonction linspace.

3) Calculer puis commenter l'erreur de ce calcul d'intégration pour $N = 100$ et $N = 1000$ en testant les deux méthodes d'intégration précédentes.

Exercice 3 : Traitement photo (suite)

Dans cet exercice, nous allons manipuler le tableau numpy Tab associé à la photo « tache.jpg » (image en noir et blanc). On note $Tab[i,j,k]$ le tableau indiquant l'intensité affectée au pixel de la ligne $i+1$, de la colonne $j+1$ d'une image « noir et blanc » ($Tab[i,j,k]$ renvoie donc un triplet comportant 3 valeurs identiques car « R=V=B »). On souhaite appliquer l'opérateur Laplacien à chaque pixel de cette image ce qui revient à redéfnir l'intensité de chaque pixel suivant la formule :

$$Tab[i,j,k] = |Tab[i-1,j,k] + Tab[i+1,j,k] + Tab[i,j-1,k] + Tab[i,j+1,k] - 4Tab[i,j,k]|$$

1) Appliquer le filtre Laplacien à la photo « tache.jpg » en utilisant la vectorisation des tableaux numpy et en vous insiprant du programme ci-dessous (qui applique ce filtre Laplacien mais sans utiliser la vectorisation).
2) Analyser la photo obtenue.

```
import numpy as np
from PIL import Image
"""importation de l'image et création d'un tableau"""
photo=Image.open("tache.jpg")
tab=np.array(photo)#tableau tridimensionnel
H,l,p=np.shape((tab))
"""création de deux tableaux afin d'éviter les effets de bords et
l'overflow"""
tab2=np.array(tab,dtype="float")#permet d'éviter l'overflow
tab3=np.copy(tab2)#permet d'éviter les effets de bords
"""méthode non vectorisée"""
for i in range(1,H-1):
    for j in range(1,l-1):
        tab2[i,j]=abs((tab3[i-1,j]+tab3[i+1,j]+tab3[i,j-
1]+tab3[i,j+1]-4*tab3[i,j]))
"""normalisation"""
maxi=np.max(tab2)
tab2=tab2*255/maxi
"""tracés"""
tab4=np.array(tab2,dtype="uint8")
photo_exo3=Image.fromarray(tab4)
photo_exo3.save("photo_exo3.jpg")
```