TD 12 : Calcul scientifique et Visualisation

L'objectif de ce TD est d'utiliser les ndarray pour du calcul scientifique. En plus de celles vues en cours, nous utiliserons les fonctions suivantes :

np.allclose() : retourne vrai si 2 éléments sont égaux à une tolérance près.
np.cumsum (A) : retourne un ndarray correspondant à la somme cumulée sur les composantes de A
np.unique(A, return.count= True) qui retourne deux tableaux : un tableau contenant les différents éléments de A et un tableau qui contient le nombre d'occurrence de ces éléments.

Algèbre linéaire :

T : Transposée d'une matrice A.T (A^T)
Produit matriciel : dot() : exemple A.dot(B)

De nombreuses fonctions d'algèbre linéaire sont définies dans le module *linalg*. On retrouve les calculs de déterminants, de valeurs et vecteurs propres, d'inversion de matrices etc…

np.linalg.det(A) : calcule le déterminant de A
np.linalg.inv(A) : calcule l'inverse de A (A^{-1}) . Une matrice est inversible si son déterminant est différent de 0. Sinon on peut calculer la pseudo-inverse
np.linalg.pinv(A) : pseudo-inverse de A
np.linalg.eig(A) : calcul des valeurs propres et des vecteurs propres d'une matrice (eighen values) . Cette fonction retourne 2 tableaux : un contient les valeurs propres de la matrice, l'autre les vecteurs propres.
np.linalg.lstsq(A, b, cond=None) : calcule la résolution selon les moindres carrés de l'équation Ax=b

Affichage :

Image : plt.imshow(Z) ajouter cmap='gray' pour niveaux de gris, ou autres valeurs, Z tableau 2D
Histogramme : plt.hist(A, bins=n), A tableau 1D n= nombre de barres
Courbe : plt.plot(x,y)

Exercice 1

Ecrire une fonction qui prend en paramètres n et m et qui retourne une matrice aléatoire de dimension (n, m+1) avec une colonne biais remplie de 1 tout à droite.

Exercice 2

Créer une matrice carrée de nombres réels aléatoires respectant une distribution normale centrée en 0. Calculer la matrice $M = R.R^T$, l'inverser et vérifier que $M \cdot M^{-1} = M^{-1} \cdot M = I$ à la précision numérique près (numpy.allclose()).

Exercice 3

Soit le code suivant qui charge une image (d'abord en couleur) puis en niveaux de gris. C'est cette image que nous utiliserons dans l'exercice.

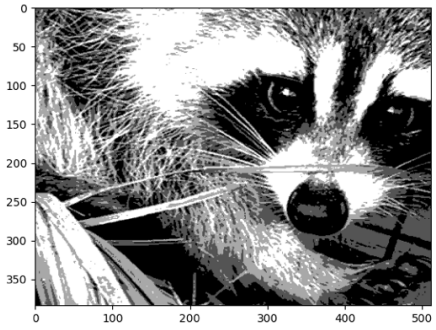
```
import numpy as np
from scipy import misc
import matplotlib.pyplot as plt

 #image couleur
face_c= misc.face()
plt.imshow(face_c)
plt.show()
```

```
print(type(face_c), face_c.shape) # tableau 3D

face= misc.face(gray=True)       #niveau de gris
plt.imshow(face, cmap=plt.cm.gray)  # affiche en niveau de gris
plt.show()
print(type(face), face.shape)     # tableau 2D
```

1) Appliquer un zoom de ¼ vers le milieu et augmenter les contrastes (augmente la luminosité des pixels clairs et baisse celle des pixels foncés)
2) Revenir à l'image zoomée sans changement de contraste. Calculer l'histogramme des niveaux de gris et l'afficher.
3) A l'aide de la fonction np.cumsum() qui calcule la somme cumulée, calculer et afficher une nouvelle image en 4 niveaux de gris (0, 84, 168, 255) ayant un histogramme équilibré. Résultat attendu :



Exercice 4

Soit la matrice 3x3 suivante :

A [[1, 5, 2],
[2, 4, 1],
[3, 6, 2]]

Calculer les valeurs propres λ_i et vecteurs propres v_i pour $i \in [1..3]$
Vérifier que ces derniers sont de norme = 1
Vérifier la relation

$$\mathbf{A}\mathbf{v} = \lambda\mathbf{v}.$$