TP N°18 : Le calcul matriciel

On importera la bibliothèque `Numpy` au moyen de la commande : `import numpy as np`

Introduction

Le calcul matriciel ...

Produits matriciels

Pour deux vecteurs $a = [a_1,...,a_n]$ et $b = [b_1,...,b_n]$, on calcule ici le produit scalaire réel $\sum_{k=1}^{n} a_k b_k$

Un vecteur v est considéré comme une ligne à gauche, et une colonne à droite.
Pour effectuer le produit matriciel : c'est la fonction `np.dot(a,b)` ou la méthode `a.dot(b)`.

```
A=np.array([[1, 2], [3, 4]])
B=np.array([[1, 1, 1], [2, 2, 2]])
np.dot(A,B) # array([[ 5,  5,  5],[11, 11, 11]])
np.dot(A,B).dot(np.ones((3,2))) # calcul du produit de plusieurs matrices
```

La fonction `vdot` permet de calculer le produit scalaire de deux vecteurs de R^n.

```
u=np.array([1, 2])
v=np.array([3, 4])
np.vdot(u, v) # 11
```

La fonction `cross` permet de calculer le produit vectoriel de deux vecteurs de R^3.

```
u=np.array([1, 0, 0])
v=np.array([0, 1, 0])
np.cross(u, v) # array([0, 0, 1])
```

Inversion de matrices, résolution de systèmes

```
A=np.array([[1, 1, 1], [2, 2, 2]])
np.transpose(A) # renvoie la transposée de A
A.T # renvoie aussi la transposée de A

import numpy.linalg as alg

alg.det(A) # le déterminant de A
a = np.array([[1,-2,3],[2,1,4],[5,-1,2]]);
b = np.array([14,12,13]);
x = alg.solve(a,b) # résout le système A·x = B. x = array([ 1., -2.,  3.]))
```

Normes matricielles et vectorielles

`linalg.norm(x)` calcule une norme d'un vecteur ou d'une matrice. $M = \sqrt{\sum |m_{i,j}|^2}$

```
m = np.arange(-10,10).reshape(5,4) # m une matrice de taille 5×4
x = np.linalg.norm(m) # x = 25.88435821108957
```

Exercice 1

1. Implémenter le pseudo-code précédent en python et vérifier son fonctionnement.