Module Numpy et tracé de courbes

Comme on l'a dit dans le TP précédent, il existe un module Python qui permet de mieux gérer les matrices : le module `numpy`. On l'importe avec la commande `import numpy as np` : toutes les commandes de `numpy` seront précédées de `np.`.

On peut définir une matrice comme liste de listes avec la commande `np.array`. Par exemple, `A = np.array([[1,2,3][4,5,6],[7,8,9]])`.

On peut récupérer la taille d'une matrice avec `nomdelamatrice.shape`, et accéder au coefficient (i, j) avec `nomdelamatrice[i,j]`. `nomdelamatrice[i,:]` récupère la $i-1$ième ligne de la matrice, et `nomdelamatrice[:,j]` la $j-1$ième colonne (attention, on renvoie une liste contenant la colonne, en ligne donc).

Les fonctions `np.zero([n,p]),np.ones([n,p]),np.eye(n)` renvoient respectivement la matrice nulle de taille $n \times p$, la matrice remplie de 1 de taille $n \times p$ et l'identité de taille $n \times n$.

Le symbole `+` permet d'ajouter deux matrices, et `*` de multiplier une matrice par un scalaire (ou de multiplier deux matrices coefficients par coefficients). On multipliera les matrices avec `np.dot`.

On peut créer des vecteurs avec les commandes `np.arange`, qui est l'équivalent de `range` et `np.linspace(debut,fin,pas)` qui crée le vecteur de premier terme `debut`, de dernier terme `fin` et ayant 6 termes identiquement espacés.

La fonction `np.reshape(vecteur,(n,p))` transforme un vecteur en matrice à n lignes et p colonnes.

Le module peut utiliser les opérations booléennes entre tableaux, qui renvoient des tableaux de booléens. Il peut aussi, en utilisant les fonctions mathématiques du module, appliquer une fonction à tous les termes d'un tableau.

Exercice 1. Tester toutes les fonctions précédentes, jusqu'à vous sentir à l'aise.

En important le module `matplotlib.pyplot` (importé en tant que `plt`), on peut tracer des courbes. On utilisera pour l'instant les fonctions `plt.plot(x,y)`, où x et y sont des tableaux de même dimension, et `plt.show()` qui affiche le dessin.

Exercice 2. Tracer les courbes des fonctions ln, exp, cos et sin.

Exercice 3. Dessiner la courbe de trajectoire

$$\begin{cases} x(t) = \sin^3(t) \\ y(t) = \cos(t) - \cos^4(t) \end{cases}.$$

Exercice 4. On veut programmer la méthode d'Euler pour la résolution d'équations différentielles.

Étant donnée une équation différentielle $y' = F(t, y)$, on découpe l'intervalle de résolution $[t_0, t_f]$ en n morceaux $t_0, t_1, \ldots, t_n = t_f$. On note le pas $h = \frac{f_f - t_0}{n}$. On a donc $t_i = t_0 + ih$. On approximera alors

$$y(t_{i+1}) \simeq y(t_i) + h * y'(t_i) = y(t_i) + h * F(t_i, y(t_i)).$$

Programmer la méthode d'Euler, et résoudre vos équations préférées : on affichera sur le même graphique la solution obtenue par méthode d'Euler, et la solution théorique.

Exercice 5. Implémenter la méthode d'Euler pour les systèmes de deux équations différntielles à deux inconnues.

Résoudre le système

$$\begin{cases} x'(t) = x(t)(3 - 2y(t)) \\ y'(t) = y(t)(-4 + x(t)) \end{cases}$$

(On tracera soit y en fonction de x, soit x et y en fonction de t.)

L'équation ci-dessus modélise l'évolution de lapins et de renards dans une région. Commenter les courbes obtenues.