# 2TSI. TD PYTHON 05

# *Diagonalisation des matrices*

❑ **Méthode 0.1.— Comment diagonaliser avec Python**

- On peut ***utiliser le sous-module*** *numpy.linalg* et on tape :

  *import numpy.linalg as alg*

  La fonction *eigvals* renvoie les valeurs propres de la matrice $A$ sous la forme d'un tableau en ligne. On tape $alg.eigvals(A)$.
  La fonction *eig* renvoie un tableau $L$ formé par une liste $L[0]$ qui est la liste des valeurs propres et par un tableau $L[1]$ qui fournit la matrice de passage de la base canonique à la base de vecteur propres. On tape $alg.eig(A)$.
  Si l'on veut retrouver la matrice $A$, on tape
  $L = alg.eig(A)$
  $L[1].dot(np.diag(L[0])).dot(alg.inv(L[1]))$
  (Voir la fiche méthode pour le produit de trois matrices et l'inversion.)
  Enfin, $numpy.poly(A)$ donne le polynôme caractéristique de $A$ sous forme d'une liste des coefficients de $\chi_A$ dans l'ordre décroissant des degrés.

- On peut ***utiliser le module*** *sympy.* Dans ce cas, une matrice ne se définit plus par *np.array* mais par $Matrix$ (voir l'exemple). Si $A$ est une matrice ainsi définie, $A.eigenvals()$ donne la séquence des valeurs propres et $A.eigenvects()$ donne en plus des valeurs propres des vecteurs propres associés.
  ***Astuce :*** on peut avoir un meilleur affichage des résultats en tapant :

  *from sympy import init_printing*
  *init_printing()*

**Exercice 01 : diagonalisation d'une matrice**

1. À l'aide de fonctions du sous module *numpy.linalg*, déterminer les valeurs propres et une matrice de vecteurs propres pour :

$$A = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}.$$

2. Faire de même à l'aide de fonctions du module *sympy*. Comparer.

**T.S.V.P →**

## Exercice 02 : valeur approchée (ou exacte) de la valeur propre de plus grand module

***Comment déterminer une valeur approchée (ou exacte) de la valeur propre de plus grand module à l'aide du quotient des traces de deux puissances itérées consécutives avec Python***
Cet algorithme étant proposé dans le programme officiel de TSI2 en Mathématiques, nous nous devons d'en parler. Ici $A \in \mathcal{M}_n(\mathbf{K})$ et diagonalisable dans $\mathcal{M}_n(\mathbf{K})$.
Si $\mathsf{Sp}\,(A) = \{\lambda_1, ..., \lambda_n\}$, ces $n$ valeurs propres n'étant pas nécessairement distinctes mais on fait l'hypothèse que l'une de ces valeurs propres, $\lambda_n$ par exemple pour fixer les idées, est de module strictement plus grand que le module des autres : $\forall\, i \in [\![1, n-1]\!]$, $|\lambda_i| < |\lambda_n|$. Alors :

$$\boxed{\lim_{p \to +\infty} \frac{\mathrm{Tr}(A^{p+1})}{\mathrm{Tr}(A^p)} = \lambda_n.}$$

1. Sous les hypothèses précédentes, justifier mathématiquement le résultat théorique.
2. L'idée est maintenant de calculer informatiquement les éléments de la suite $(\mathrm{Tr}(A^p))_{p \in \mathbf{N}^*}$ puis de faire le rapport de deux éléments consécutifs de cette suite pour $p$ grand et on obtient une valeur approchée de la valeur propre de plus grand module de $A$.
   Écrire la fonction $VPGM$ d'argument la matrice $A$ et l'entier $n$ qui renvoie les $n$ premiers termes de la suite précédente.
3. Déterminer une valeur approchée de la valeur propre de plus grand module de la matrice :
   $$A = \begin{pmatrix} 4 & -3 & -3 \\ 3 & -2 & -3 \\ 3 & -3 & -2 \end{pmatrix}$$
   en utilisant la procédure Python $VPGM$ précédente et vérifier le résultat avec la fonction $eigvals$.
4. Faire de même avec $B = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$. Que constate-t-on ?