I Création et manipulations simples

Avant toute chose, on s'assurera de charger les bibliothèques `numpy` et `matplotlib.pyplot` pour traiter ce TD.

Exercice 1

1. Tracer la courbe représentative de la fonction $f : x \mapsto x^3 - 3x - 1$ sur $[-3, 3]$ en utilisant 350 points.
2. On observe que cette fonction s'annule 3 fois. En utilisant uniquement les données calculées à la question précédente, donner une valeur approchée (ou un encadrement) de chaque solution de l'équation $f(x) = 0$ appartenant à $[-3, 3]$.

Exercice 2

On peut créer facilement des vecteurs numpy (et donc des matrices) en utilisant les commandes suivantes :

```
np.array(L)  # où L est une liste déjà créée
np.zeros(n) # vecteur nul de taille n
np.zeros((n, p))  # matrice nulle de taille n,p. Noter la double parenthèse
```

1. Créer la matrice A nulle et de taille $5, 3$. Tester ensuite les commandes

   ```
   A.shape
   A.reshape((3,5))
   ```

   Que se passe-t-il si on veut donner la forme $5, 5$ à la matrice A ?
2. Créer la matrice carrée de taille 10 : $\begin{pmatrix} 0 & 1 & 2 & \dots & 9 \\ 10 & 11 & 12 & \dots & 19 \\ \vdots & & & & \vdots \\ 90 & & & \dots & 99 \end{pmatrix}$. On pourra commencer par créer une liste contenant les entiers voulus, puis la transformer en vecteurs puis en changer sa forme.

Exercice 3

Pour créer la matrice identité, on peut utiliser au choix

```
np.eye(n)  # n est la taille
np.identity(n)
```

1. Créer la matrice identité I carrée de taille 5. Expliquer la valeur de

   ```
            I[3:5, 0:2]

   ```

   On pourra changer la valeur des indices donnés ici pour mieux comprendre.
2. Créer la matrice que l'on peut représenter par $\mathrm{A} = \begin{pmatrix} \mathrm{I}_2 & \mathrm{I}_2 \\ \mathrm{I}_2 & \mathrm{I}_2 \end{pmatrix}$ où on a noté non pas des coefficients mais des matrices identité de taille 2. Ainsi A est carrée de taille 4.
   Pour créer A, on commencera par créer la matrice identité de taille 4, puis on remplacera des sous-matrices par la matrice identité de taille 2.

II Matrices

Exercice 4

Dans toute cette partie, on appelle norme d'une matrice $\mathrm{M} = (m_{ij}) \in \mathcal{M}_{n,p}(\mathbb{R})$ (carrée ou non) le nombre

$$\|\mathrm{M}\| = \max_{i,j \in [\![1,n]\!] \times [\![1,p]\!]} |m_{ij}|,$$

ie le plus grand coefficient de M, en valeur absolue.
Créer la fonction `norme` qui prend en entrée une matrice (vecteur numpy pour nous) et retourne sa norme. Tester cette fonction avec quelques matrices : carrée, colonne, ligne, et contenant également des coefficients négatifs.

Exercice 5

Dans la suite, on s'intéresse à la matrice M définie par $\mathrm{M} = \begin{pmatrix} \mathrm{A} & -\mathrm{I_N} & 0 & \dots & \dots & 0 \\ -\mathrm{I_N} & \mathrm{A} & -\mathrm{I_N} & 0 & \dots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & \dots & -\mathrm{I_N} & \mathrm{A} & -\mathrm{I_N} \\ 0 & \dots & & & -\mathrm{I_N} & \mathrm{A} \end{pmatrix}$ où A est carrée de taille N, et est décrite par : la diagonale de A est remplie de 4, la sous-diagonale ainsi que la sur-diagonale de A sont remplies de -1. La diagonale de M comporte N fois la matrice A.

1. Créer la fonction `construitA(N)` qui prend en entrée un entier naturel non nul N et renvoie la matrice A.
2. Quelle est la taille de la matrice M ? Créer la fonction `construitM(N)` où l'entrée est la même qu'à la question précédente.
   On testera le résultat avec N = 3.

Exercice 6
Si `A, B` sont des vecteurs numpy représentant des matrices que l'on peut multiplier (les tailles sont compatibles), le produit matriciel s'écrit

```
A.dot(B)  # pour calculer AB
```

On souhaite implémenter le calcul de la suite de vecteurs suivantes : F_0 est une colonne donnée, puis pour tout $p \in \mathbb{N}$ on pose $F_{p+1} = \frac{MF_p}{\|MF_p\|}$, où M est la matrice définie dans la question précédente.

1. Créer la fonction `calculeFp(M, p)` qui calcule F_p pour un entier p.
   Pour cela on choisira "au hasard" un vecteur F_0 de la bonne taille grâce à la commande

   ```
   np.random.rand(n, r)
   ```

   qui retourne une matrice de taille n, r et dont les coefficients sont choisis au hasard dans $[0, 1]$.
2. On admet que la suite crée à la question précédente se stabilise dans le sens où pour $\varepsilon > 0$ on peut toujours trouver un rang p tel que $\|F_{p+1} - F_p\| \leqslant \varepsilon$.
   En s'inspirant du code de la question précédente, créer la fonction `stabilise(M, eps)` qui prend en paramètre un flottant `eps` strictement positif et retourne le premier vecteur F_p vérifiant $\|F_{p+1} - F_p\| \leqslant eps$.
   On testera avec `eps` valant 10^{-4}.
3. En notant F le vecteur trouvé grâce au test de la question précédente, vérifier que l'on a $MF \approx \lambda F$ où λ est un réel à préciser.

On peut montrer que F est une valeur approchée d'un vecteur propre associé à la valeur propre λ qui est la plus grande (en valeur absolue) des valeurs propres de M.

Exercice 7
Nous avons déjà trouvé un vecteur propre ainsi qu'une valeur propre de M. En remarquant que notre matrice M est symétrique réelle, nous pouvons "assez facilement" poursuivre la réduction.
La première étape consiste à répéter l'opération de l'exercice précédent sur une matrice de taille $N^2 - 1$ bien construite.

1. Les autres vecteurs propres peuvent être chois orthogonaux à F car M est symétrique réelle. Grâce à la commande

   ```
   import scipy.linalg as scl
   scl.null_space(S)
   ```

   où S est la matrice d'un système homogène et qui retourne une base de l'ensemble des solutions, construire B une base de $\text{Vect}(F)^\perp$ (espace dont on connaît un vecteur normal).
2. La matrice de passage P est constituée de F comme première colonne et de B pour toutes les autres. Construire cette matrice puis la matrice M_1 qui est la matrice carrée de taille $N * *2 - 1$ extraite (en bas à droite) de $P^{-1}MP$ (et il faut trouver comment on inverse une matrice en numpy au passage).
3. On répète l'opération sur M_1, sachant que les coefficients de cette matrices sont des coordonnées dans la base représentée par B.