1 Interface

L'interface proposée à l'oral devrait être Pyzo ou Spyder. Bien penser à s'habituer à l'interface avant l'oral, en particulier l'éditeur et la console. Savoir comment interpréter le script complet, ou bien seulement une partie de son code. Ne pas oublier d'enregistrer régulièrement son script.

2 Les aide-mémoire

Il est important de se familiariser avec les aide-mémoire, et de ne pas les découvrir le jour de l'oral. Comprendre en particulier ce que sont les arguments et les valeurs renvoyées par les différentes fonctions. Ne pas avoir de doute sur la façon d'importer un module, ou simplement une fonction de module.

3 Utiliser l'aide intégrée

Savoir utiliser l'aide intégrée à l'aide de `help`. Indiquer la fonction entre guillemet permet de ne pas charger le module. S'habituer aux différentes rubriques de l'aide, et savoir ce qu'est un argument optionnel d'une fonction.

```
>>> help("numpy.random.randint")

Help on built-in function randint in numpy.random:

numpy.random.randint = randint(...) method of numpy.random.mtrand.RandomState instance
    randint(low, high=None, size=None, dtype=int)

    Return random integers from `low` (inclusive) to `high` (exclusive).

    Return random integers from the "discrete uniform" distribution of
    the specified dtype in the "half-open" interval [`low`, `high`). If
    `high` is None (the default), then results are from [0, `low`).

    Parameters
    ----------
    low : int or array-like of ints
        Lowest (signed) integers to be drawn from the distribution (unless ``high=None``, in which
        case this parameter is one above the *highest* such integer).
    high : int or array-like of ints, optional
        If provided, one above the largest (signed) integer to be  drawn from the distribution
        (see above for behavior if ``high=None``). If array-like, must contain integer values
    size : int or tuple of ints, optional
        Output shape.  If the given shape is, e.g., ``(m, n, k)``, then ``m * n * k`` samples are drawn.
        Default is None, in which case a single value is returned.
    dtype : dtype, optional
        Desired dtype of the result. Byteorder must be native. The default value is int.

    Returns
    -------
    out : int or ndarray of ints
        `size`-shaped array of random integers from the appropriate
        distribution, or a single such random int if `size` not provided.

    Examples
    --------
    >>> np.random.randint(2, size=10)
    array([1, 0, 0, 0, 1, 1, 0, 0, 1, 0]) # random

    Generate a 2 x 4 array of ints between 0 and 4, inclusive:
    >>> np.random.randint(5, size=(2, 4))
    array([[4, 0, 2, 1], # random
           [3, 2, 2, 0]])
```

Réalisation de tracés

- Import du module de tracé `import matplotlib.pyplot as plt`
- Tracé de la grille `plt.grid()`
- Repère orthonormal `plt.axis('equal')`
- Tracé de ligne brisé, de fonctions, de courbe paramétrée `plt.plot(X,Y)`
- Tracé de surface `numpy.meshgrid`, ~~`mpl_toolkits.mplot3d.Axes3D`~~
- Tracé de lignes de niveau `numpy.meshgrid`, `plt.contour`

La liste des opérations proposée ici est très incomplète, se référer à l'aide-mémoire

922.1

(a) Pour $p = \frac{1}{50}, \frac{2}{50}, \dots, \frac{49}{50}$, représenter les points de coordonnées : $\big(p, 1 - 0.15\sin(\pi p)\big)$.

(b) Sur le même graphe, pour $p \in]0,1[$, tracer la courbe de la fonction :

$$p \mapsto \frac{2 - 2p + p^2}{2 - p}$$

922.2

(a) Représenter dans un repère orthonormal, pour $x \in \left[0, \frac{\pi}{2}\right]$, le graphe de la fonction sin, ainsi que la première bissectrice.

(b) Sur le même graphe, représenter les 10 premiers termes de la suite définie par :

$$\begin{cases} u_0 = \dfrac{\pi}{2} \\ u_{n+1} = \sin(u_n) \quad \forall n \in \mathbb{N} \end{cases}$$

Zoom sur la 3d

```python
import matplotlib.pyplot as plt
import numpy as np

def f(x,y):
    return ...

X = np.linspace(1, 49, 4)   # [ 1. 17. 33. 49.]
Y = np.linspace(0, 50, 4)   # [ 0. 25. 50.]

X, Y = np.meshgrid(X, Y)
                            # [[ 1. 17. 33. 49.]   [[ 0.  0.  0.  0.]
                            #  [ 1. 17. 33. 49.]    [25.  25. 25. 25.]
                            #  [ 1. 17. 33. 49.]]   [50. 50. 50. 50.]]
Z = f(X, Y)
                            # [[1.49494949 1.39759036 1.25373134 1.01960784]
                            #  [0.7006237  0.97372742 0.97372742 0.7006237 ]
                            #  [1.01960784 1.25373134 1.39759036 1.49494949]]

fig = plt.figure()
ax = fig.add_subplot(projection='3d')

ax.plot_surface(X, Y, Z)
# ax.set_aspect('equal')

plt.savefig("...")          # ou plt.show()

plt.figure()
plt.contour(X, Y, Z, [1.0, 1.1, 1.2])
plt.show()
```

922.3

Soit $D = [1,49] \times [0,50]$ et $f : (x,y) \in D \mapsto \dfrac{x}{x+y} + \dfrac{50-x}{100-(x+y)}$.
Représenter la surface d'équation $z = f(x,y)$

Probabilités

- Simuler 20 lancers de dé à 6 faces `numpy.random.randint(1,7,20)`
- Simuler 20 valeurs d'une va suivant $\mathscr{U}(\llbracket 1,7\llbracket)$ `numpy.random.randint(1,7,20)`
- Simuler une épreuve de Bernoulli de paramètre .2 `numpy.random.binomial(1,.2)`
- Simuler 42 valeurs d'une va suivant $\mathscr{G}(.7)$ `numpy.random.geometric(.7,42)`
- Obtenir un réel « au hasard » dans $[0,1[$ `numpy.random.random()`

La liste des opérations proposée ici est très incomplète, se référer à l'aide-mémoire
Savoir dire : « La loi faible des grands nombres nous permet de dire que si l'on calcule la moyenne d'un grand nombre d'épreuves, la probabilité qu'on soit loin de l'espérance est faible. »

922.4

Au rez-de-chaussée d'un immeuble, n personnes entrent dans un ascenseur et choisissent de façon aléatoire un étage parmi p possibles. On note X le nombre d'arrêts.

(a) Écrire une fonction simulant X.

(b) Estimer $\mathrm{E}(X)$ lorsque $n = 10$ et $p \in \{15, 30, 100\}$.

922.5

On considère N joueurs qui jouent à un jeu. Lors d'une manche, ils lancent simultanément chacun une pièce équilibrée. Si l'un d'entre eux obtient un résultat différent de tous les autres (F et tous les autres P, ou P et tous les autres F), il gagne. Sinon, on rejoue une manche. Le jeu s'arrête lorsqu'un joueur gagne.

(a) Écrire une fonction `manche(N)` qui renvoie un booléen indiquant si la manche est gagnante ou non.

(b) Écrire une fonction `partie(N)` qui renvoie le nombre de manches jouées à l'issue de la partie.

(c) Tester avec 10 joueurs, puis 20 joueurs.

922.6

Soit $p \in]0,1[$, $(X_n)_{n\in\mathbb{N}^*}$ des v.a. indépendantes suivant une loi de probabilité définie de la façon suivante :

$$P(X_n = 1) = p \text{ et } P(X_n = 2) = 1 - p$$

On pose $S_n = \sum_{i=1}^{n} X_i$ et $Y_k = \mathrm{Inf}\{n \in \mathbb{N}^*,\ S_n \geqslant k\}$. On admet qu'il s'agit d'une v.a.

(a) Implanter une fonction simulant Y_k en fonction de k et de p.

(b) Définir une fonction permettant de calculer une approximation m_k de $E(Y_k)$, pour un p donné.

(c) Tracer m_k en fonction de k, pour $p \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$.

Calcul matriciel

- Créer des matrices, des vecteurs comme $\begin{pmatrix}1&2&3\\2&3&1\\3&1&2\end{pmatrix}$ `numpy.array`
- Faire des produits de matrices `A.dot(B)`
- Utiliser la transposée d'une matrice `A.T`
- Obtenir les éléments propres d'une matrice `numpy.linalg.eig`

La liste des opérations proposée ici est très incomplète, se référer à l'aide-mémoire

922.7

On considère la matrice $A = \begin{pmatrix}1&2&3\\4&5&6\\7&8&9\end{pmatrix}$.

(a) Calculer et afficher A^2.

(b) Déterminer le spectre de A. La matrice A est-elle diagonalisable ? est-elle inversible ?

(c) Définir X vecteur propre associé à la plus petite valeur propre (en valeur absolue) et calculer AX.

922.8

Soit $n \geqslant 1$ un entier et a un réel non nul. On considère la matrice carrée d'ordre n à coefficients réels :

$$A_{n,a} = \begin{pmatrix}0&1/a&0&\dots&0\\a&0&1/a&\ddots&\vdots\\0&a&\ddots&\ddots&0\\\vdots&\ddots&\ddots&0&1/a\\0&\dots&0&a&0\end{pmatrix}$$

(a) Écrire une fonction qui, étant donnés un entier $n \geqslant 1$ et un réel non nul a, renvoie la matrice $A_{n,a}$.

(b) Donner les valeurs approchées décimales des valeurs propres de $A_{n,a}$ pour $3 \leqslant n \leqslant 8$ et a dans $\{-2,-1,1,2,3\}$.

922.9

Pour une matrice $A \in \mathcal{M}_{n,p}(\mathbb{R})$ (où n et $p \in \mathbb{N}^*$), on note $A^\top$ la matrice transposée de A. On note également $\mathrm{Im}(A) = \{AX \mid X \in \mathcal{M}_{p,1}(\mathbb{R})$ et $\mathrm{Ker}(A) = \{X \in \mathcal{M}_{p,1}(\mathbb{R}) | AX = 0\}$, il s'agit respectivement de l'image et du noyau de l'application linéaire : $u_A : X \in \mathcal{M}_{p,1}(\mathbb{R}) \mapsto AX \in \mathcal{M}_{n,1}(\mathbb{R})$.
On note aussi pour $D \in \mathcal{M}_{n,1}(\mathbb{R})$, les systèmes linéaires $\Sigma_{A,D} : AX = D$ et $\Sigma'_{A,D} : A^\top AX = A^\top D$; il s'agit en fait d'équations matricielles d'inconnue $X \in \mathcal{M}_{p,1}(\mathbb{R})$.
On considère $A_1 = \begin{pmatrix}1&0&1\\-1&1&0\\0&0&1\\1&1&0\end{pmatrix}$, $D_1 = \begin{pmatrix}0\\5\\10\\5\end{pmatrix}$, $A_2 = \begin{pmatrix}0&2\\0&1\end{pmatrix}$, $D_2 = \begin{pmatrix}10\\5\end{pmatrix}$ et $A_3 = \begin{pmatrix}1&0&0\\0&0&-1\\0&1&0\end{pmatrix}$, $D_3 = \begin{pmatrix}10\\0\\5\end{pmatrix}$.
Afin d'alléger les notations, on note respectivement Σ_i et Σ'_i les systèmes Σ_{A_i,D_i} et Σ'_{A_i,D_i}, pour $i \in \{1,2,3\}$.

(a) Résoudre avec Python les systèmes Σ_i et Σ'_i pour $i \in \{1,2,3\}$.

(b) Écrire une fonction `rangs(A)` qui prend en argument un tableau `numpy` représentant une matrice A et qui renvoie la liste de longueur 2 contenant les rangs des matrices A et $A^\top A$.

(c) Tester la fonction précédente avec les matrices A_i pour $i \in \{1,2,3\}$. Que peut-on conjecturer ?

(d) Pour $i \in \{1,2,3\}$, comparer $\mathrm{Ker}(A_i)$ et $\mathrm{Ker}({A_i}^\top A_i)$. Conjecture ?

Polynômes

- Définir un polynôme comme $1+X+X^2$

```
X = numpy.polynomial.Polynomial([0,1])
P=1+X+X**2
```

La liste des opérations proposée ici est très incomplète, se référer à l'aide-mémoire

922.10

Soit $(P_n)_{n\geqslant 1}$ la suite de polynômes définie par :

$$P_1 = X,\ P_2 = X^2 - 1 P_{n+2} = XP_{n+1} - p_n \quad \forall n \in \mathbb{N}^*$$

(a) Calculer les coefficients de $P_3, \ldots, P_8$.

(b) Donner des valeurs approchées des racines de $P_3, \ldots, P_8$.

922.11

Centrale II

Soit $G \in \mathbb{R}[X]$ un polynôme de degré $n+1$, scindé sur $\mathbb{R}$ et à racines simples. On définit g l'application de $\mathbb{R}_n[X]$ dans $\mathbb{R}[X]$ qui associe à un polynôme P le reste de la division euclidienne de PX par G.

(a) Montrer que g est un endomorphisme de $\mathbb{R}_n[X]$.

(b) Dans cette question, $n=3$ et $G = X^4 + 2X^3 - X^2 - 2X = X(X-1)(X+1)(X+2)$.

b1. Donner la matrice de g dans la base canonique.

b2. g est-elle diagonalisable ?

b3. Tracer sur $[-3, 2]$ les fonctions polynomiales associées aux vecteurs propres de g.

(c) On revient au cas général. L'application g est-elle diagonalisable ?

Analyse numérique

- Résoudre (numériquement) une équation comme $x^5 + x^2 + 1 = 0$ `scipy.optimize.fsolve`
- Résoudre (numériquement) un système comme $\begin{cases} x^2 - y^2 = 1 \\ x + 2y - 3 = 0 \end{cases}$ `scipy.optimize.root`
- Calculer (numériquement) une intégrale comme $\displaystyle\int_0^{+\infty} \mathrm{e}^{-t^2}\,\mathrm{d}t$ `scipy.integrate.quad`
- Résoudre (numériquement) une éq. diff. comme $y' = x + y^2$ `scipy.integrate.odeint`
- Résoudre (numériquement) une éq. diff. comme $y'' + xy' - y^3 = \cos x$ `scipy.integrate.odeint`
- Résoudre (numériquement) un syst. diff. comme $\begin{cases} x' = -x - y \\ y' = x - y \end{cases}$ `scipy.integrate.odeint`
- Calculer avec des nombres complexes comme $1 + \mathrm{i}$ `1+1j`

922.12

On pose $P_0 = 1$, $P_1 = 2X$, $P_{n+2} = 2XP_{n+1} - P_n$.

(a) Calculer $(P_i)_{0 \leqslant i \leqslant 8}$.

On pose $\langle P, Q\rangle = \displaystyle\int_{-1}^{1} P(x)Q(x)\sqrt{1-x^2}\,\mathrm{d}x$.

(b) Calculer $\langle P_i, P_j\rangle$ pour $i, j \in \{0, \dots, 8\}$. Conjecture ?

922.13

On note (E) l'équation différentielle :

$$\mathrm{sh}^5(x)\ y' + 2\,\mathrm{ch}(x)\,\mathrm{sh}^2(x)\ y = \mathrm{ch}^3(x)$$

(a) Tracer avec Python la solution vérifiant $y(1) = 1$, sur $[1, 10]$, en utilisant un pas de 0.01. Conjecturer le comportement de la solution au voisinage de 0.

(b) On souhaite vérifier la conjecture. Tracer à rebours cette solution sur $[0.1, 1]$, toujours avec un pas de 0.01.

922.14

On considère sur $\mathbb{R}_+$ l'équation, pour tout entier $n \geqslant 1$:

$$x^n + x^{n-1} + \cdots + x - 1 = 0 \qquad (\mathcal{E}_n)$$

qui admet une unique solution notée u_n pour tout entier $n \geqslant 1$.

(a) Afficher les 100 premières valeurs approchées de la suite $(u_n)_{n \geqslant 1}$.

922.15

Concours Centrale Supélec - Math II

Pour les simulations informatiques sous Python, on importera les bibliothèques scientifiques à l'aide des instructions suivantes :

```
import numpy as np
import matplotlib.pyplot as plt
import scipy.integrate as integr
```

On considère les fonctions f et g définies sous condition de convergence par :

$$f\ :\ x\mapsto\int_0^{+\infty}\frac{\sin t}{x+t}\,\mathrm{d}t \qquad \text{et} \qquad g\ :\ x\mapsto\int_0^{+\infty}\frac{\mathrm{e}^{-xt}}{1+t^2}\,\mathrm{d}t$$

(a) Montrer que f est définie sur $\mathbb{R}_+$.

(b) Que donne l'outil informatique pour le calcul de $f(0)$, $f(1)$ et $f(10)$? Commenter.

(c) Soit $A\in\mathbb{R}_+$, on pose $f_A\ :\ x\mapsto\displaystyle\int_0^A\frac{\sin t}{x+t}\,\mathrm{d}t$.

c1. Écrire une fonction Python de paramètres A et x qui calcule $f_A(x)$.

c2. Tracer simultanément les courbes représentatives de f_{50}, f_{100}, f_{200} et f_{500} sur l'intervalle $[0,5]$. Commenter.

(d) Montrer que la fonction g est définie et continue sur $\mathbb{R}_+$ et de classe $\mathcal{C}^2$ sur $\mathbb{R}_+^*$.

(e) Tracer simultanément les courbes représentatives de g et de f_{200}. Que peut-on conjecturer ?

(f) Après avoir changé son expression à l'aide d'un changement de variable, montrer que la fonction f est de classe $\mathcal{C}^2$ sur $\mathbb{R}_+^*$.

(g) Donner une équation différentielle d'ordre 2 vérifiée par f et g. Conclure.

922.16

Centrale II

On définit, pour $n\in\mathbb{N}^*$: $u_n=\displaystyle\prod_{k=1}^{n}\left(1-\frac{1}{k^2\pi^2}\right)$.

(a) Donner une approximation de u_n pour $n\leq 10$ puis de $\dfrac{1}{u_{10^k}}$ pour $k\in\{1,..,4\}$. Conjecture ?

(b) Soient f et g définies sur $]0,\pi[$ par :

$$\forall t\in]0,\pi[\qquad f(t)=\sum_{n=1}^{+\infty}\frac{2t}{t^2-n^2\pi^2}\qquad\text{et}\qquad g(t)=\frac{\cos t}{\sin t}-\frac{1}{t}.$$

Montrer que f et g sont prolongeables en des fonctions continues sur $[0,\pi[$.

(c) Représenter graphiquement f et g sur $]0,\pi[$. On admettra dans la suite l'égalité de ces deux fonctions.

(d) Calculer, pour $x\in]0,\pi[$, la limite lorsque n tend vers $+\infty$ de $\displaystyle\sum_{k=1}^{n}\ln\left(1-\frac{x^2}{k^2\pi^2}\right)$.

On pourra calculer de deux façons $\displaystyle\int_0^x g(t)dt$.

(e) En déduire la limite de $(u_n)_{n\in\mathbb{N}}$.