# Externat Des Enfants Nantais

## Devoir surveillé n° 5 - Mathématiques

### Bcpst 1

20 janvier 2024

Durée: 3 heures

*La rédaction et la présentation interviennent pour une part très importante dans la note ; il est recommandé de les soigner. Les exercices sont indépendants. Les calculatrices sont autorisées.*

**Exercice I** *(AV 2018)*

**Consignes**

— *Les programmes sont à rédiger en langage Python.*

— *Les extraits de codes matérialisés par* `__________` *correspondent à des portions à compléter.*

Dans cette partie, on suppose que le fichier Python débute par l'importation du module `matplotlib.pyplot` et par la définition des deux listes `Ls` et `Lv`, de la façon suivante :

```
import matplotlib.pyplot as plt

Ls = [0.333, 0.167, 0.0833, 0.0416, 0.0208, 0.0104, 0.0052]
Lv = [3.636, 3.636, 3.236, 2.666, 2.114, 1.466, 0.866]
```

**1. a.** Écrire une fonction `inv` qui prend en entrée une liste de nombres (supposés non nuls) `L` et qui renvoie la liste composée des inverses de ces nombres.

Exemple : `inv([0.25, 2, 1, 8])` renvoie `[4.0, 0.5, 1.0, 0.125]`.

**b.** Écrire une version améliorée `inv_ex` de la fonction `inv` qui prend en entrée une liste de nombres `L`, puis : si un de ces nombres est nul, alors elle renvoie le booléen `False`, sinon elle renvoie la liste composée des inverses de ces nombres.

**c.** On dispose de deux listes `Ls` et `Lv` de valeurs prises par deux grandeurs $s$ et $v$ au cours du temps comme rappelé en préambule. Compléter les lignes de codes suivantes afin qu'elles effectuent le tracé des points de coordonnées $(s^{-1}, v^{-1})$ (les points seront représentés par des petits cercles et ne seront pas reliés entre-eux) :

```
plt.plot(______________)
plt.show()
```

**2. Écriture de fonctions préliminaires.**

***Pour cette question, on s'interdit d'utiliser les commandes préprogrammées de Python qui renvoient la somme, la moyenne ou la variance. Avant chaque fonction, on écrira brièvement le raisonnement suivi et la formule qu'elle est censée calculer.***

**a.** Écrire une fonction `moyenne` qui prend en entrée une liste `X` (non vide) de nombres réels et qui renvoie la moyenne des éléments de la liste.

**b.** Écrire une fonction `variance` qui prend en entrée une liste `X` (non vide) de nombres réels et qui renvoie la variance des éléments de la liste.

**3. a.** Compléter le programme suivant afin qu'il renvoie la valeur de la covariance de `X` et `Y` si elle existe et le booléen `False` sinon. (*Même consigne qu'à la question 2.*)

```
1  def cov(X: list, Y: list):
2      nx, ny = len(X), len(Y)
3      if __________ or nx == 0:
4          return False
5      else:
6          __________
7          __________
8          for k in range(__________):
9              S = ____________
10         return ____________
```

**b.** Parmi les quatre valeurs suivantes, lesquelles ne peuvent pas être renvoyées par la fonction `cov` ? (*On justifiera les réponses*).

**i.** `True`

**ii.** `[1, 2]`

**iii.** `"False"`

**iv.** `-0.5`

**c.** On considère les fonctions `Coef` et `Trace` suivantes :

```
1  def Coef(X, Y):
2      a = cov(X, Y)/variance(X)
3      b = moyenne(Y)-cov(X, Y)/variance(X)*moyenne(X)
4      return([a, b])
```

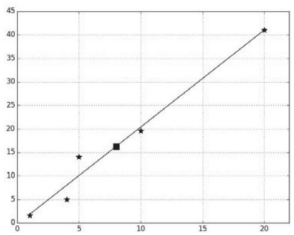
```
1  def Trace(X, Y):
2      [a, b] = Coef(X,Y)
3      xmin = min(X); xmax = max(X)
4      plt.plot(X, Y, "*")
5      plt.plot(__________)
6      plt.plot([moyenne(X)], [moyenne(Y)], "s")
7      plt.grid()
8      plt.show()
```

**i.** Compléter la ligne 5 de la fonction `Trace` afin de tracer le segment d'extrémités : `(xmin, a*xmin + b)` et `(xmax, a*xmax + b)`.

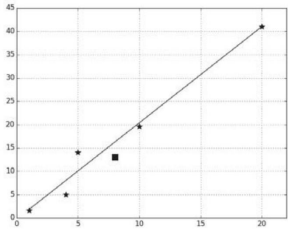
**ii.** Donner l'équation de la droite qui passe par ces deux points.

**iii.** Quel nom porte cette droite ?

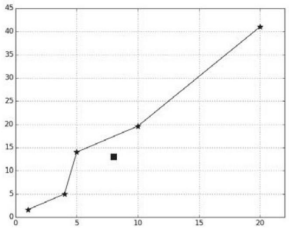
**iv.** On exécute la fonction `Trace` pour des listes `X` et `Y` quelconques de taille 5. Pour chacun des trois tracés suivants, indiquer avec justification s'il peut être ou non le résultat de `Trace` ?



(a) Tracé n°1



(b) Tracé n°2



(c) Tracé n°3

## Annexe : Rappels Python

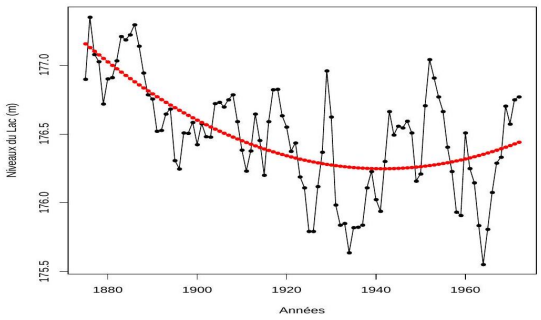
On suppose que le module `matplotlib.pyplot`, qui permet de tracer des graphiques, est importé via `import matplotlib.pyplot as plt`.
Les variables `X` et `Y` sont ici deux listes de réels, de même longueur.

| Python | Interprétation |
|---|---|
| `plt.plot (X,Y)` | Place les points dont les abscisses sont contenues dans X et les ordonnées dans Y et les relie entre eux par des segments. Si cette fonction n'est pas suivie de `plt.show ()`, le graphique n'est pas affiché |
| `plt.grid()` | Dessine en arrière plan du graphique un quadrillage. |
| `plt.show ( )` | Affiche le(s) tracé(s) précédemment créé(s) par `plt.plot` |
| `plt.plot(X,Y,"o")` | Même effet que `plt.plot (X,Y)` à la différence près que les points sont représentés par un symbole en forme de cercle et ne sont pas reliés. En remplaçant `o` par `s` (respectivement par `*`), le symbole est un carré (respectivement une étoile) |
| `min(X)` (resp. `max(X)`) | Renvoie le minimum (resp. le maximum) de `X` |

**Exercice II** *(AV 2019)*

Dans ce sujet, on va s'intéresser à la modélisation des niveaux annuels du lac Huron.

**1.** Dans cette question, on note $n$ un entier naturel supérieur ou égal à 3 . Pour tout entier $i$ dans $\{1, 2, \ldots, n\}$, on note $y_i$ le niveau du lac Huron de la $i$ ème observation à l'instant $t_i$. Dans la figure, les points de coordonnées $(t_i, y_i)$ sont représentés par des points $(\bullet)$. Dans la suite, les $t_i$ sont supposés distincts. On s'intéresse à l'ajustement d'un polynôme de degré 2 au nuage de points $((t_1, y_1), (t_2, y_2), \ldots, (t_n, y_n))$. Un tel ajustement est représenté dans la figure ci-dessous :




Pour réaliser cet ajustement on utilise le critère des moindres carrés défini ci-dessous. On définit la fonction $F : \mathbb{R}^3 \longrightarrow \mathbb{R}$ par :

$$\forall (a, b, c) \in \mathbb{R}^3, \quad F(a, b, c) = \sum_{i=1}^{n} \left( y_i - a - bt_i - ct_i^2 \right)^2$$

**a.** Expliquer en quelques lignes ou à l'aide d'un dessin ce que l'on cherche à faire lorsque l'on minimise la fonction $F$ ci-dessus par rapport à $a, b$ et $c$.

**b.** Calculer les dérivées partielles de $F$ par rapport à $a, b$ et $c$ que l'on notera $\frac{\partial F}{\partial a}(a, b, c)$, $\frac{\partial F}{\partial b}(a, b, c)$ et $\frac{\partial F}{\partial c}(a, b, c)$, respectivement.

**2.** Des éléments de syntaxe Python, et en particulier l'usage du module numpy, sont donnés en annexe à la fin de la partie 2. Dans tout ce qui suit, les variables $n, p, A, M, i, j$ et $c$ vérifient les conditions suivantes qui ne seront pas rappelées à chaque question :

- $n$ et $p$ sont des entiers naturels tels que $p \geq n \geq 2$ ;
- $A$ est une matrice carrée à $n$ lignes inversible ;
- $M$ est une matrice à $n$ lignes et $p$ colonnes telle que la sous-matrice carrée constituée des $n$ premières colonnes de $M$ est inversible ;
- $i$ et $j$ sont des entiers tels que $0 \leq i \leq n-1$ et $0 \leq j \leq n-1$ ;
- $c$ est un réel non nul.

On note $L_i \leftarrow L_i + cL_j$ l'opération qui ajoute à la ligne $i$ d'une matrice la ligne $j$ multipliée par $c$.

**a.** Soit la fonction `initialisation`

```
def initialisation(A):
    n = np.shape(A)[0]
    mat = np.zeros((n,2*n))
    for i in range(0, n):
        for j in range(0, n):
            mat[i,j] = A[i,j]
    return(mat)
```

Pour chacune des affirmations suivantes, indiquer si elle est vraie ou fausse, en justifiant. L'appel initialisation(A) renvoie :

**i.** une matrice rectangulaire à $n$ lignes et $2n$ colonnes remplie de zéros ;

**ii.** une matrice de même taille que $A$ ;

**iii.** une erreur au niveau d'un range ;

**iv.** une matrice rectangulaire telle que les $n$ premières colonnes correspondent aux $n$ colonnes de $A$, et les autres colonnes sont nulles.

**b.** Les trois fonctions `multip`, `ajout` et `permut` suivantes ne renvoient rien : elles modifient les matrices auxquelles elles s'appliquent.

**i.** Que réalise la fonction multipt ?

```
def multip(M, i, c):
    p = np.shape(M)[1]
    for k in range(0, p):
        M[i,k] = c*M[i,k]
```

**ii.** Compléter la fonction `ajout`, afin qu'elle effectue l'opération $L_i \leftarrow L_i + cL_j$.

```
def ajout(M, i, j, c):
    p = np.shape(M)[1]
    for k in range(0, p):
        _____ ligne(s) à compléter _____
```

**iii.** Écrire une fonction `permut` prenant pour argument `M`, `i` et `j`, et qui modifie `M` en échangeant les valeurs des lignes `i` et `j`.

Dans la suite du sujet, l'expression « opération élémentaire sur les lignes » fera référence à l'utilisation de `permut`, `multip` ou `ajout`.

**c.** Soit la colonne numéro `j` dans la matrice `M`. On cherche le numéro `r` d'une ligne où est situé le plus grand coefficient (en valeur absolue) de cette colonne parmi les lignes `j` à `n-1`. Autrement dit, `r` vérifie :

$$\left|\texttt{A[r,j]}\right| = \max\left\{\left|\texttt{A[i,j]}\right| \text{ pour } \texttt{i} \text{ tel que } \texttt{j} \leq \texttt{i} \leq \texttt{n}-1\right\}$$

Écrire une fonction `rang_pivot` prenant pour argument `M` et `j`, et qui renvoie cette valeur de `r`. Lorsqu'il y a plusieurs réponses possibles pour `r`, dire (avec justification) si l'algorithme renvoie le plus petit `r`, le plus grand `r` ou un autre choix. (*L'utilisation d'une commande max déjà programmée dans Python est bien sûr proscrite.*)

**d.** Soit la fonction `mystere` :

```
def mystere(M):
    n = np.shape(M)[0]
    for j in range(0, n):
        r = rang_pivot(M, j)
        permut(M, r, j)
        for k in range(j+1, n):
            ajout(M, k, j, -M[k,j]/M[j,j])
        print(M)
```

**i.** On considère dans cette question l'algorithme `mystere` appliqué à la matrice $M_1 = \begin{bmatrix} 3 & 2 & 2 \\ -6 & 0 & 12 \\ 1 & 1 & -3 \end{bmatrix}$

Indiquer combien de fois la ligne `print(M)` est exécutée ainsi que les différentes valeurs qu'elle affiche.

**ii.** De façon générale, que réalise cet algorithme ?

**e.** Soit la fonction `reduire`, qui modifie $M$ :

```
def reduire(M):
    n = np.shape(M)[0]
    mystere(M)
    for i in range(0, n):
        multip(M, i, 1/M[i,i])
    #Les lignes suivantes sont à compléter :
    ________________________
```

**i.** Compléter la fonction afin que la portion de code manquante effectue les opérations élémentaires suivantes sur les lignes :

```
pour j prenant les valeurs n-1, n-2, ..., 1, faire :
    pour k prenant les valeurs j-1, j-2, ..., 0, faire :
        Lk reçoit Lk - M[k, j] Lj
```

**ii.** Indiquer ce que réalise cette fonction.

**f. Inversion de A**

**i.** Écrire une fonction `augmenter` prenant pour argument `A` et qui renvoie la matrice de taille $(n, 2n)$ définie ainsi :

- dans la partie gauche (composée des $n$ lignes et $n$ premières colonnes), elle contient les coefficients de $A$ ;
- dans la partie droite (composée des $n$ lignes et $n$ dernières colonnes), elle contient les coefficients de la matrice identité de taille $n$.

Par exemple,

$$\text{pour } \begin{bmatrix} 1 & 2 \\ -1 & 3 \end{bmatrix} \text{ la fonction renvoie } \begin{bmatrix} 1 & 2 & 1 & 0 \\ -1 & 3 & 0 & 1 \end{bmatrix}$$

**ii.** À l'aide des fonctions précédentes, proposer un raisonnement permettant d'inverser A.

**iii.** Écrire une fonction `inverser` prenant pour argument $A$ et qui renvoie la matrice inverse de $A$, en suivant le raisonnement décrit en question précédente.

**iv.** Quelle méthode connue venez-vous d'implémenter ?

## Annexe : Rappels Python

On considère que le module `numpy`, permettant de manipuler des tableaux à deux dimensions, est importé via `import numpy as np`.
Pour une matrice $M$ à $n$ lignes et $p$ colonnes, les indices vont de 0 à $n-1$ pour les lignes et de 0 à $p-1$ pour les colonnes.

| Python | Interprétation |
|---|---|
| `abs(x)` | Valeur absolue du nombre $x$ |
| `M[i,j]` | Coefficient d'indice $(i,j)$ de la matrice $M$ |
| `np.zeros((n,p))` | Matrice à $n$ lignes et $p$ colonnes remplie de zéros |
| `T = np.shape(M)` | Dimensions de la matrice $M$ |
| `T[0]` | Nombre de lignes |
| `T[1]` | Nombre de colonnes |
| `M[a:b, c:d]` | Matrice extraite de $M$ constituée des lignes $a$ à $b-1$ et des colonnes $c$ à $d-1$ :<br>si $a$ (resp. $c$) n'est pas précisé, l'extraction commence à la première ligne (resp. colonne)<br>si $b$ (resp. $d$) n'est pas précisé, l'extraction finit à la dernière ligne (resp. colonne) incluse |