Feuille de TP n°3
Solutions de base

Objectif : Implémenter le calcul des solutions de base sous Python.

1 Remarques préliminaires

Ce TP est noté sur 10 points. Les deux parties sont relativement indépendantes, mais au sein de chaque partie, les exercices ne le sont pas.

Pour utiliser la bibliothèque `numpy`, on commencera par ajouter la ligne de code suivante en début de script :

```
1 import numpy as np
```

2 Solution de base et base réalisable

Soit $A \in \mathcal{M}_{m,n}(\mathbb{R})$ une matrice (avec m lignes et n colonnes, $n \geq m$) et $b \in \mathbb{R}^m$ un vecteur. On considère un opérateur d'extraction de colonnes

$$\gamma : \{1, \cdots, m\} \to \{1, \cdots, n\} \text{ croissant}$$

auquel on associe l'unique application croissante $\hat{\gamma}$ qui a pour image les indices non atteints par γ, c'est-à-dire

$$\hat{\gamma} : \{1, \cdots, n-m\} \to \{1, \cdots, n\} \text{ avec } \hat{\gamma}\big(\{1, \cdots, n-m\}\big) = \{1, \cdots, n\} \setminus \gamma\big(\{1, \cdots, m\}\big)$$

On note $B = A_\gamma$ la matrice constituée des colonnes $\gamma(1)$, $\cdots$, $\gamma(m)$ de A et $N = A_{\hat{\gamma}}$ celle constituée des colonnes $\hat{\gamma}(1)$, $\cdots$, $\hat{\gamma}(n-m)$ de A.

On dit que γ définit une **base** de $\mathbb{R}^m$ si la matrice B est inversible.

Exercice 1 – Extraction de colonnes On considère la matrice suivante :

$$A = \begin{pmatrix} 1 & 1 & 3 & 4 \\ 3 & 2 & 1 & 1 \\ 2 & 3 & 2 & 3 \end{pmatrix}$$

et l'opérateur d'extraction de colonnes $\gamma(1) = 1$, $\gamma(2) = 2$ et $\gamma(3) = 4$.

1. Définir la matrice A en Python.
2. Écrire une fonction `extraction()` qui prend en argument une matrice A de taille $m \times n$ et un extracteur de colonnes défini par le vecteur $\gamma = (\gamma(1), \cdots, \gamma(m))$ de taille m et qui renvoie la matrice $B = A_\gamma$ de taille $m \times m$. La tester sur la matrice A définie à la question précédente.
3. Vérifier en Python que B est inversible. *On pourra utiliser la fonction* `np.linalg.matrix_rank()`.

4. Écrire une fonction `testBase()` qui prend en argument une matrice A de taille $m \times n$ et un extracteur de colonnes défini par le vecteur $\gamma = (\gamma(1), \cdots, \gamma(m))$ de taille m et qui renvoie `1` si γ définit une base ou `0` sinon. La tester sur la matrice définie à la première question.

Soit γ définissant une base et $x \in \mathbb{R}^m$. On dit alors que les composantes $x_{\gamma(i)}$ pour $i \in \{1, \ldots, m\}$ du vecteur x sont **les variables en base** pour la base γ, tandis que les autres composantes sont **les variables hors-base** pour la base γ. Le sous-vecteur de x composé de ses variables en base est noté x_B et celui composé de ses variables hors-base est noté x_N.

Pour parcourir l'ensemble des extracteurs de colonnes, on va utiliser le code suivant

```
from itertools import combinations

def ensembleExtracteur(A):
    m, n = A.shape
    colonnes = np.arange(1, n + 1)
    for gamma in combinations(colonnes, m):
        print(gamma)
```

Exercice 2 – Base

1. Tester la focntion `ensembleExtracteur()` sur la matrice A de l'**Exercice 1**.
2. À partir de la fonction `ensembleExtracteur()`, écrire une fonction `ensembleBase()` qui prend en argument une matrice A de taille $m \times n$ et qui, pour tout extracteur de colonnes γ, extrait la matrice $B = A_\gamma$ et affiche le vecteur $\gamma = (\gamma(1), \cdots, \gamma(m))$ puis le texte `' est une base'` si γ définit une base, ou le texte `' nest pas une base'` si γ ne définit pas une base. *On pourra utiliser la fonction* `testBase()` *de l'***Exercice 1**.
3. Tester cette fonction avec la matrice A de l'**Exercice 1**.

Exercice 3 – Variables en base, variables hors base

On considère le vecteur

$$x = \begin{pmatrix} -1 \\ 1 \\ 2 \\ -2 \end{pmatrix}$$

et la base γ définie dans l'**Exercice 1**.

1. Définir le vecteur x en Python.
2. À partir du vecteur x, extraire les sous-vecteurs x_B et x_N associés à la base γ.
3. Soit y le vecteur défini par

$$y = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$$

Écrire une fonction `ecritureBase()` qui prend en argument un vecteur x de taille n, une base définie par le vecteur $\gamma = (\gamma(1), \cdots, \gamma(m))$ et un vecteur y de taille m, et qui affecte au sous-vecteur x_B les coefficients du vecteur y et annule les variables hors-base de x. La tester sur les vecteurs ci-dessus.

Si γ définit une base, alors on introduit la solution de base du système linéaire $A\,x = b$ associée à γ comme étant l'unique vecteur $x^* \in \mathbb{R}^m$ tel que

$$x^*_B = B^{-1}b \qquad \text{et} \qquad x^*_N = 0$$

Enfin, si $x^* \geq 0$, alors on dit que x^* est une solution de base **admissible** pour tout problème d'optimisation linéaire de contraintes

$$A\,x = b \qquad \text{et} \qquad x \geq 0$$

Dans ce cas, γ est une **base réalisable** pour ce même problème.

Exercice 4 – Solution de base On reprend la matrice A et la base γ de l'**Exercice 1** et on considère le vecteur

$$b = \begin{pmatrix} 18 \\ 10 \\ 16 \end{pmatrix}$$

1. Résoudre le système linéaire $B\,x_B = b$ avec Pÿthon. *On pourra utiliser la fonction* `np.linalg.solve()`.
2. À partir de la fonction `ecritureBase()` de l'**Exercice 3**, écrire une fonction `solutionBase()` qui prend en argument une matrice A de taille $m \times n$, un vecteur b de taille m et une base définie par le vecteur $\gamma = (\gamma(1), \cdots, \gamma(m))$, et qui renvoie x^* la solution de base du système $A\,x = b$ associée à γ.
3. Vérifier que x^* est bien une solution du système linéaire $A\,x = b$.
4. En reprenant la fonction `ensembleBase()` de l'**Exercice 2**, écrire une fonction `ensembleSolutionBase()` qui prend en argument une matrice A de taille $m \times n$, un vecteur b de taille m, et qui pour tout extracteur de colonnes γ teste si γ est une base, et si c'est le cas, calcule et affiche la solution de base associée.

Exercice 5 – Solution de base admissible On reprend la matrice A et la base γ de l'exercice 1 et le vecteur b de l'exercice 4.

1. Écrire une fonction `testPositif()` qui prend en argument un vecteur x et qui renvoie 1 si x n'a que des composantes positives ou nulles, et 0 sinon.
2. En reprenant la fonction `ensembleSolutionBase()` de l'exercice 2, écrire une fonction `ensembleSolutionBaseAdmissible()` qui prend en argument une matrice A, un vecteur b, et qui pour tout extracteur γ teste si γ est une base réalisable et si oui affiche la solution de base associée.