Chapitre 3: Algorithme du simplexe - PL avec Pyhton

Dr. Cheikh GUEYE
Université Cheikh Anta Diop de Dakar (UCAD, Sénégal)
Université Claude Bernard Lyon 1 (UCBL, France)
Titulaire d'un Doctorat en Mathématiques appliquées (UCBL-Lyon)
Titulaire d'un Doctorat en Analyse, Statistiques et Applications (UCAD-Dakar)
Laboratoire de Mathématiques et Applications (LMA-UCAD)
cheikh.gueye1990@yahoo.fr, *cheikh39.gueye@ucad.edu.sn*

Institut Supérieur d'Informatique **Dép:** Réseaux et Systèmes

Niveau: Licence 2 RT/RI/CS/SEMI **Année:** 2025 - 2026

Plan

1- Présentation

Dans le chapitre précédent, nous avons présenté des programmes linéaires à deux variables. Toutefois, en pratique, les modèles d'optimisation linéaire peuvent impliquer plusieurs dizaines de variables et de contraintes. Cela met en évidence la nécessité de disposer d'une méthode algébrique permettant de résoudre efficacement les programmes linéaires de dimension supérieure à deux. Cette méthode est connue sous le nom de **méthode du Simplexe**.

La méthode du Simplexe est l'algorithme le plus couramment utilisé pour la résolution des programmes linéaires. Elle a été introduite en 1948 par George B. Dantzig. Depuis lors, cet algorithme a fait l'objet de nombreuses études scientifiques et a permis la résolution efficace de multiples problèmes d'optimisation dans divers domaines.

La méthode du Simplexe est une procédure algébrique itérative permettant de résoudre les programmes linéaires, notamment lorsque ceux-ci comportent plusieurs variables et contraintes.

Elle consiste à parcourir successivement les sommets de la région admissible, en améliorant à chaque itération la valeur de la fonction objectif, jusqu'à atteindre une solution optimale ou constater l'absence d'optimalité.

2- Principe

Le principe consiste à partir d'une solution admissible correspondant à un point extrême du polyèdre des contraintes, puis à chercher à améliorer la valeur de la fonction objectif en se déplaçant vers un point extrême voisin. En répétant ce processus de manière successive, il devient alors possible d'atteindre, de proche en proche, un sommet qui réalise la valeur optimale.

Il est possible de représenter le problème sous forme de tableau, ce qui facilite considérablement les calculs nécessaires à l'application de la méthode du Simplexe.

- **Étape 1 : Forme Standard** Il convient de réécrire le programme linéaire sous sa forme standard, c'est-à-dire en exprimant toutes les contraintes sous forme d'égalités et en s'assurant que toutes les variables sont non négatives. Cette transformation permet d'appliquer directement la méthode du Simplexe.
- **Étape 2 : Solution de base initiale :** Il s'agit de déterminer la première solution de base réalisable.

- **Étape 3 :** Il s'agit d'effectuer une série d'itérations sur les tableaux du Simplexe jusqu'à l'obtention de la solution optimale.

2.1. La forme standard d'un programme linéaire

La forme standard d'un programme linéaire consiste à exprimer toutes les contraintes sous forme d'égalités et à imposer que toutes les variables soient non négatives. Cette représentation est essentielle pour l'application de la méthode du Simplexe.

- Pour transformer une contrainte d'inégalité de type $\leq$ en une contrainte d'égalité, on ajoute au membre de gauche une variable supplémentaire, appelée variable d'écart ou variable slack, qui permet de compenser la différence entre le membre de gauche et le membre de droite de la contrainte.

$$3x_1 + 2x_2 \leq 2 \text{ se transforme en } 3x_1 + 2x_2 + e_1 = 2,\ e_1 \geq 0.$$

Les variables d'écart représentent la différence entre le membre de droite et le membre de gauche d'une contrainte d'inégalité de type $\leq$. Elles permettent de transformer cette inégalité en égalité tout en conservant l'interprétation physique ou économique de la contrainte. Dans le contexte de la programmation linéaire, ces variables mesurent donc la marge restante par rapport à la limite imposée par la contrainte.

- Pour transformer une contrainte d'inégalité de type $\geq$ en une contrainte d'égalité, on soustrait du membre de gauche une variable supplémentaire, appelée variable d'excès (ou surplus), qui compense la différence entre le membre de gauche et le membre de droite.

$$\mathbf{3x_1 + 2x_2 \geq 2 \text{ se transforme en } 3x_1 + 2x_2 - e_1 = 2,\ e_1 \geq 0.}$$

Les variables d'excès (ou surplus) représentent la quantité par laquelle le membre de gauche d'une contrainte de type $\geq$ dépasse le membre de droite. Elles permettent de transformer une inégalité en égalité pour l'application de la méthode du Simplexe, tout en conservant la signification de la contrainte. En d'autres termes, elles mesurent l'excès par rapport à la limite imposée par la contrainte.

La méthodologie proposée pour cette technique consiste à visiter tous les états possibles dans un système en partant d'un sommet vers un sommet adjacent de manière à réviser et améliorer la fonction objectif. Pour ce faire, nous procédons à l'exploration des différentes démarches selon l'ordre de priorité donné ci dessous :

- **Démarche 1 :** On démarre l'application de l'approche (méthode du simplexe) par la transformation des contraintes d'inégalité en contraintes d'égalité en ajoutant les variables d'écart.
- **Démarche 2 :** Dans un second temps nous sélectionnons les variables originales comme variables hors-base (VHB) et les variables d'écarts comme variable basique (VB). puis nous effectuerons une permutation entre une variable hors-base de notre choix qui sera remplacée par une variable de base (entrante). Le choix de la variable entrante repose sur la variable dont le coefficient est le plus élevé dans la fonction objectif (ou fonction économique).

- **Démarche 3 :** La variable sortante est la première à s'annuler. On répète le processus jusqu'à ce que tous les coefficients de fonction objectif soient négatifs ou nuls. Dans ce cas, on arrête et la solution optimale est trouvée.

Résolution par Tableaux de simplexe-Problème de maximisation

Nous commençons par transformer les contraintes d'inégalité en contraintes d'égalité en introduisant les variables d'écart appropriées. Cette étape permet d'obtenir la forme standard du programme linéaire.

- **Étape 1 (Tableau initial).** On construit un tableau à deux dimensions de taille $r \times s$, où le nombre de colonnes correspond au nombre total de variables du système (variables de décision et variables d'écart), auquel s'ajoute la colonne des solutions C et la colonne K.

  Le nombre de lignes du tableau est égal au nombre d'équations du système, sans tenir compte des contraintes de positivité.

Lors de la première itération, on sélectionne comme variable entrante celle dont le coefficient est le plus élevé (positif) dans la ligne de la fonction objectif. On encadre alors la colonne correspondante, appelée **colonne pivot**.

On calcule ensuite le minimum des rapports entre le terme du membre de droite de chaque contrainte et le coefficient correspondant dans la colonne pivot. Les lignes dont le coefficient dans la colonne entrante est négatif, nul ou non défini ne sont pas prises en compte dans ce calcul. On encadre alors la ligne où ce minimum est obtenu ; cette ligne est appelée **ligne pivot**.

Le coefficient situé à l'intersection de la colonne pivot et de la ligne pivot est appelé **élément pivot**.

- **Étape 2 (Tableau 2).** On reconstruit ensuite le tableau du simplexe, en veillant à conserver la même dimension que le tableau initial.

  –> On commence par construire la nouvelle ligne pivot, laquelle se calcule de la manière suivante :

$$\text{Ligne (nouveau tableau)} = \frac{\textit{Ancienne ligne pivot}}{\textit{element pivot}}$$

  C'est-à-dire que l'on divise toute la ligne pivot par l'élément pivot.

  –> Tous les autres coefficients de la colonne pivot, à l'exception de l'élément pivot, sont ramenés à zéro.

–> Si un élément de la ligne du pivot est nul, on recopie simplement la valeur correspondante de la colonne du pivot.

–> Ensuite, on met à jour les autres lignes du tableau (case vide) à l'aide de la formule suivante, selon la méthode du rectangle :

$$L_i = \text{Position (ligne actuelle)} - \frac{\textit{Produit croise dans l'autre sens du pivot}}{\textit{pivot}}$$

Où L_i désigne la position dans le nouveau tableau et l_i la position correspondante dans l'ancien tableau.

On construit le rectangle en partant de la position (l_i, pivot).

$$L_i = l_i - \frac{a \times b}{pivot}$$

Conditions d'arrêt. La question suivante se pose : l'optimum a-t-il été atteint ?

Rappel : l'optimum est atteint lorsque tous les coefficients de la ligne de la fonction objectif Z sont négatifs ou nuls.

Résumé

- **Étape 1 : Forme standard.** on introduit des variables d'écart afin de transformer les contraintes d'inégalité en contraintes d'égalité.
- **Étape 2 : Détermination de la solution de base initiale.** Pour déterminer la solution de base initiale, on considère les variables d'écart comme variables de base (VB) et on met les variables de décision à zéro (VHB). Les valeurs des variables de base sont alors directement données par les termes constants des contraintes.
- **Étape 3 : Créer le tableau initial (ou table de simplexe).** on construit le tableau du simplexe initial en y inscrivant les coefficients de la fonction objectif ainsi que ceux des contraintes.
- **Étape 4 : Itérations.** Appliquer des itérations pour améliorer la solution en sélectionnant une variable d'entrée et une variable de sortie.
- **Conditions d'arrêt.** Vérifier les conditions d'arrêt pour déterminer si la solution optimale a été trouvée.

TD 3 : Algorithme des tableaux de simplexe

Dans les exercices suivants, appliquer l'algorithme tableaux de simplexe pour trouver la solution optimale de programmation linéaire.

Exercice 1 : Problème de maximisation

Soit le problème d'optimisation suivant :

$$(P1)\begin{cases} \textit{Maximiser } Z = 240x_1 + 160x_2 \\ \textbf{sc} \\ x_1 + 2x_2 \leq 150 \\ 4x_1 + 2x_2 \leq 400 \\ \textit{avec } x_1, x_2 \geq 0 \end{cases}$$

Solution optimale : $x_1 = 83,33$, $x_2 = 33,33$ et la valeur optimale $Z = 2533,33$.

Exercice 2 : Soit le problème d'optimisation suivant :

$$(P2)\begin{cases} \text{Maximiser } Z = 3x_1 + 5x_2 \\ \textbf{sc} \\ x_1 + 2x_2 \leq 10000 \\ 2x_1 + 3x_2 \leq 12000 \\ x_1 + 4x_2 \leq 15000 \\ \text{avec } x_1, x_2 \geq 0 \end{cases}$$

Solution optimale : $x_1 = 600$, $x_2 = 3600$ et la valeur optimale $Z = 19800$.

Exercice 3 : Soit le problème d'optimisation suivant :

$$(P3)\begin{cases} \text{Maximiser } Z = 3x_1 + 4x_2 + 2x_3 \\ \textbf{sc} \\ x_1 + x_2 - x_3 \leq 10 \\ x_1 - 2x_2 + 3x_3 \leq 14 \\ \text{avec } x_1, x_2, x_3 \geq 0 \end{cases}$$

La solution optimale est : $x_1 = 0,\ x_2 = 44,\ x_3 = 34$ et la valeur optimale $Z = 244$.

Exercice 4 : Soit le problème d'optimisation suivant :

$$(P4)\begin{cases} \textit{Maximiser } Z = 4x_1 + 6x_2 + 3x_3 \\ \textbf{sc} \\ x_1 + 6x_2 + 2x_3 \leq 24 \\ x_1 + 3x_2 - 2x_3 \leq 9 \\ \textit{avec } x_1, x_2, x_3 \geq 0 \end{cases}$$

La solution optimale est : $x_1 = 0,\ x_2 = 3,83,\ x_3 = 16,74$ et la valeur optimale $Z = 78,10$.

Problématique 1 : Tous les coefficients de la fonction objectif sont égaux.

1. **Problème :** Lorsque deux ou plusieurs coefficients de la fonction objectif sont égaux et sont les plus positifs (pour un problème de maximisation), il existe un choix multiple de la variable entrante.
2. **Implications :**
   - **Choix arbitraire de la variable entrante** : Si plusieurs variables ont le même coefficient maximal, n'importe laquelle peut être choisie pour entrer dans la base.
   - **Solutions multiples :** Cette situation peut conduire à des solutions optimales alternatives, car différentes variables entrant dans la base peuvent produire le même optimum.
   - **Faisabilité** : Il faut s'assurer que la solution reste faisable à chaque itération, en vérifiant les ratios pour choisir la ligne pivot.

Conclusion : L'égalité des coefficients dans la fonction objectif ne bloque pas la méthode du simplexe mais nécessite de prendre une décision judicieuse lors du choix de la variable entrante et peut révéler l'existence de solutions multiples optimales.

Problématique 2 : Égalité de plusieurs rapports positifs lors du choix de la ligne pivot

1. **Analyse de la Problématique 2 : Égalité de plusieurs rapports positifs.** Lorsqu'on calcule le minimum des rapports entre les termes du membre de droite et les coefficients correspondants dans la colonne pivot, il peut arriver que plusieurs lignes donnent le même rapport minimal positif. Cette situation indique que plusieurs variables candidates peuvent sortir de la base simultanément.
2. **Implications dans la méthode du simplexe :**
   - Le choix de la ligne pivot doit être effectué de manière arbitraire parmi celles présentant le rapport minimal.
   - Cette décision peut conduire à des solutions alternatives optimales, car le tableau final peut dépendre de la ligne pivot choisie.
   - Il est important de vérifier après chaque itération que la solution reste faisable et que l'optimalité est atteinte.

Conclusion : L'égalité de plusieurs rapports positifs ne bloque pas la méthode du simplexe, mais elle nécessite une attention particulière dans le choix de la ligne pivot et peut conduire à des solutions multiples optimales.

Application

Trouver la solution optimale des problèmes linéaires suivants en appliquant la méthode du simplexe :

1. **Les coefficients de la fonction objectif sont égaux :**

$$(P1)\begin{cases} Max\ Z_1 = 12x_1 + 12x_2 \\ x_1 + x_2 \leq 7 \\ 2x_1 + x_2 \leq 9 \\ x_1, x_2 \geq 0 \end{cases} \quad et \quad (P2)\begin{cases} Max\ Z_2 = 3x + 3y \\ x + 2y \leq 8 \\ 3x + y \leq 9 \\ x, y \geq 0 \end{cases}$$

2. **Égalité des plus petits rapports positifs :**

$$(P3)\begin{cases} Max\ Z_1 = 8x_1 + 5x_2 \\ x_1 + x_2 \leq 7 \\ 2x_1 + x_2 \leq 14 \\ x_1, x_2 \geq 0 \end{cases} \quad et \quad (P4)\begin{cases} Max\ Z_2 = 3x + 2y \\ x + 2y \leq 4 \\ 3x + y \leq 6 \\ x, y \geq 0 \end{cases}$$

Solution : Méthode de résolution par le simplexe (cas d'égalité des coef de la fonction objectif)

$$(P1)\begin{cases} Max\ Z_1 = 12x_1 + 12x_2 \\ x_1 + x_2 \leq 7 \\ 2x_1 + x_2 \leq 9 \\ x_1, x_2 \geq 0 \end{cases}$$

Étape 1 : Forme standard

- Transformer toutes les contraintes en égalités en introduisant les variables d'écart (et d'excès si nécessaire).
- S'assurer que toutes les variables sont non négatives.
- Vérifier que le problème est bien sous forme maximisation (ou convertir si besoin).

$$(P1)\begin{cases} Max\ Z_1 = 12x_1 + 12x_2 \\ x_1 + x_2 + e_1 = 7 \\ 2x_1 + x_2 + e_2 = 9 \\ x_1, x_2, e_1, e_2 \geq 0 \end{cases}$$

- Les variables x_1 et x_2 sont appelées des variables initiales.
- Les variables e_1 et e_2 sont appelées des variables d'écarts.
- Les coefficients des variables d'écarts dans le fonction objectif sont nuls c'est-à-dire

$$Max\ Z_1 = 12x_1 + 12x_2 + 0e_1 + 0e_2$$

Étape 2 : Solution de base initiale

La solution de base est donnée à l'origine $x_1 = x_2 = 0$ et $e_1 = 7$, $e_2 = 9$.
La valeur de la fonction objectif $z = 0$.

- Les variables x_1 et x_2 sont appelées des variables hors bases (V.H.B).
- Les variables e_1 et e_2 sont appelées des variables de bases (V.B).

Étape 3 : Construction du tableau initial (premier tableau)

Écrire le premier tableau du simplexe contenant :

- Les variables de base (première colonne),
- Les variables hors base (première ligne puis ajouter les variables de base)
- Les coefficients des contraintes,
- Les valeurs du second membre (C),
- Une colonne supplémentaire pour calculer les rapports,
- Les coefficients de la fonction objectif (la dernière ligne).

	x_1	x_2	e_1	e_2	C	K
e_1	**1**	**1**	1	0	7	K_1 K'_1
e_2	**2**	**1**	0	1	9	K_2 K'_2
Coef de Z_1	**12**	**12**	0	0	0	

On constate que les coefficients de la fonction objectif sont égaux. Dans ce cas, nous avons deux possibilités pour choisir la colonne du pivot, car les

Lorsque les coefficients de la fonction objectif sont identiques, le choix de la variable entrante n'est pas unique. Dans cette situation, plusieurs colonnes peuvent être candidates au pivot. En particulier, les variables x_1 et x_2 ont le même coefficient dans la fonction objectif ; elles constituent donc toutes les deux des options valides pour la variable entrante au cours de l'itération du simplexe.

Voici la méthode à suivre

- **On suppose que x_1 est la variable entrante, puis on calcule les rapports afin de déterminer le pivot.**

$$K_1 = \frac{7}{1} = 7 \quad et \quad K_2 = \frac{9}{2} = 4,5$$

$min(K_1; K_2) = K_2 = \frac{9}{2}$. Ainsi, K_2 représente le plus petit rapport positif, ce qui détermine la variable sortante et identifie l'élément pivot dans le tableau du simplexe.

- **On suppose que x_2 est la variable entrante, puis on calcule les rapports afin de déterminer le pivot.**

$$K_1' = \frac{7}{1} = 7 \quad et \quad K_2' = \frac{9}{1} = 9.$$

$min(K_1'; K_2') = K_1' = 7$. Ainsi, K_1' représente le plus petit rapport positif, ce qui détermine la variable sortante et identifie l'élément pivot dans le tableau du simplexe.

Dans le cas où plusieurs choix de variables entrantes sont possibles, on calcule les rapports positifs associés à chaque hypothèse. On compare ensuite les deux plus petits rapports positifs obtenus. La règle consiste alors à retenir le plus grand de ces deux rapports, afin de déterminer la variable sortante selon la stratégie retenue pour éviter les ambiguïtés ou les cycles.

Conclusion. $max(K_2; K_1') = max\left(7; \frac{9}{2}\right) = 7$. Ainsi, la variable x_2 est choisie comme variable entrante, tandis que la variable e_1, devient la variable sortante. L'élément situé à l'intersection de la colonne de x_2 et de la ligne de e_1 constitue le pivot, dont la valeur est égale à 1.

Étape 4 : Présentation du deuxième tableau du simplexe

	x_1	x_2	e_1	e_2	C	K
x_2	1	1	1	0	7	
e_2	$L_1 = 1$	0	$L_2 = -1$	1	$L_3 = 2$	
Coef de Z_1	$L_4 = 0$	0	$L_5 = -12$	0	$L_6 = -84$	

L'optimalité est atteinte dès lors que tous les coefficients de la fonction objectif (ligne de Z_1) sont devenus négatifs ou nuls. Cela signifie qu'aucune amélioration supplémentaire de la valeur de Z_1 n'est possible, et que la solution courante constitue donc la solution optimale.

Conclusion. La résolution du programme linéaire par la méthode du simplexe conduit à la solution optimale suivante :

$$x_1^* = 0, \quad x_2^* = 7, \quad e_1 = 0, \quad e_2 = 2.$$

La valeur optimale de la fonction objectif s'élève à : $Z_1^* = 84$.

Cette solution respecte l'ensemble des contraintes du modèle et maximise la fonction objectif. Elle correspond donc à l'unique solution optimale obtenue au terme de l'algorithme du simplexe.

Solution : Égalité des rapports

$$(P3)\begin{cases} Max\ Z_1 = 8x_1 + 5x_2 \\ x_1 + x_2 \leq 7 \\ 2x_1 + x_2 \leq 14 \\ x_1, x_2 \geq 0 \end{cases}$$

Étape 1 : Forme standard

$$(P3)\begin{cases} Max\ Z_1 = 8x_1 + 5x_2 \\ x_1 + x_2 + e_1 = 7 \\ 2x_1 + x_2 + e_2 = 14 \\ x_1, x_2, e_1, e_2 \geq 0 \end{cases}$$

Étape 2 : Solution de base initiale

La solution de base est donnée à l'origine $x_1 = x_2 = 0$ et $e_1 = 7$, $e_2 = 14$.
La valeur de la fonction objectif $z = 0$.

- Les variables x_1 et x_2 sont appelées des variables hors bases (V.H.B).
- Les variables e_1 et e_2 sont appelées des variables de bases (V.B).

Étape 3 : Construction du tableau initial (premier tableau)

	x_1	x_2	e_1	e_2	C	K
e_1	1	1	1	0	7	$K_1 = 7$
e_2	2	1	0	1	14	$K_2 = 7$
Coef de Z_1	8	5	0	0	0	

- L'élément 8 identifié dans le tableau constitue le pivot, c'est-à-dire l'élément utilisé pour réaliser le changement de base. Par conséquent, la variable x_1 est sélectionnée comme variable entrante pour l'itération du simplexe.

Comment choisir le plus petit rapport positif ?

1. On fait la somme des coefficients de la ligne considérée sauf les contraintes C
2. On divise cette somme par le coefficient de la variable entrante dans cette ligne (la colonne du pivot).

Ce calcul permet de déterminer le rapport positif, qui est utilisé pour identifier la variable sortante et l'élément pivot du tableau. Cette étape garantit que la nouvelle solution reste faisable après le pivotage.

- 1+1+1+0=3 et on divise cette somme par le coefficient correspondant de la colonne du pivot $\frac{3}{1} = 3$.
- 2+1+0+1=4 et on divise cette somme par le coefficient correspondant de la colonne du pivot $\frac{4}{2} = 2$.

Lorsque plusieurs rapports positifs sont calculés, la règle consiste à sélectionner le plus grand rapport parmi eux. Donc $max(3; 2) = 3$.

Le rapport correspondant à cette valeur détermine la variable sortante et l'élément pivot à utiliser pour la prochaine itération du simplexe.

1 est le pivot et e_1 est la variable sortante.

Étape 4 : Deuxième tableau du simplexe

	x_1	x_2	e_1	e_2	C	K
e_1	1	1	1	0	7	
e_2	0	-1	-2	1	0	
Coef de Z_1	0	-3	-8	0	-56	

La condition d'optimalité dans la méthode du simplexe est satisfaite lorsque tous les coefficients de la ligne de la fonction objectif Z_1 sont négatifs ou nuls.

Dans ce cas :

- Aucune variable entrante ne peut améliorer la valeur de Z_1.
- La solution courante est donc optimale.

Cette règle garantit que l'on a atteint la valeur maximale de la fonction objectif, sous les contraintes données.

Conclusion. La résolution du programme linéaire par la méthode du simplexe conduit à la solution optimale suivante :

$$x_1^* = 7, \quad x_2^* = 0, \quad e_1 = 0, \quad e_2 = 0.$$

La valeur optimale de la fonction objectif s'élève à : $Z_1^* = 56$.

Résumé : Principe de l'algorithme du simplexe

Voici l'idée principale de l'algorithme du simplexe :

1. Pour commencer, on sélectionne un sommet initial ;
2. On teste si ce sommet est l'optimum ;
3. Si le sommet que l'on vient d'examiner n'est pas optimal, on se déplace sur un sommet voisin pour lequel la fonction objectif s'améliore et on repasse à l'étape précédente.

Le sommet optimal est atteint lorsqu'aucun des sommets voisins ne permet plus l'amélioration de la fonction objectif.

En s'appuyant sur le principe précédent, voici l'aspect algorithmique de la méthode du simplexe :

Début :

- Saisir la fonction objectif.
- Saisir le type initial.
- Standardiser le programme.
- Convertir le programme en tableau ;

Tant que : Les coefficients de la fonction objectif ne sont pas tous nuls ou négatifs.

- **Sélection de la variable entrante (VE) :** Choisir le coefficient le plus grand (positif) de la fonction objectif.
- **Sélection de la variable sortante (VS) :** Choisir le rapport minimum (le plus petit rapport positif).
- **Définir le pivot.**

Multiplier la ligne du pivot par 1/pivot.

Étendre l'opération aux autres lignes.

Fin Tant que.

Fin.

Dualité

La méthode du simplexe permet d'obtenir une solution optimale à tout programme linéaire en variables continues (non entières). La théorie de la dualité donne d'autres informations caractérisant une solution optimale.

Programme Dual (D)

A tout programme linéaire, on peut associer un autre programme linéaire appelé programme dual. Considérons un programme linéaire sous forme canonique, ce programme sera appelé programme primal (P).

Si un modèle de programmation linéaire possède une solution optimale, il en est de même pour son dual, et les valeurs optimales des deux modèles sont égales.

La solution optimale du dual correspond aux multiplicateurs optimaux. Nous pouvons les lire directement dans le tableau optimal du simplexe : ce sont les coefficients dans la ligne correspondant à l'objectif.

Le coût réduit (correspondant à l'opposé du coefficient dans la ligne de l'objectif) mesure la variation de l'objectif entraîné par une augmentation d'une unité de la valeur de la variable hors-base associée.

Analyse de sensibilité

En général, le coût réduit d'une variable hors-base indique le changement dans l'objectif apporté par une augmentation d'une unité de la valeur de cette variable. Pour les variables d'écart, ce principe peut se formuler ainsi : le coût réduit d'une variable d'écart hors-base indique le changement dans l'objectif apporté par une diminution d'une unité du terme de droite associé.

Ceci est un exemple d'analyse de sensibilité : un paramètre (ici, un terme de droite) est modifié et nous mesurons la sensibilité de la solution optimale à ce changement.

Nous pouvons aussi mesurer la sensibilité de la solution optimale à un changement d'un terme de droite ou d'un coefficient dans l'objectif en résolvant à nouveau le modèle modifié.

Primal	**Dual**
Variable	Contrainte
Contrainte	Variable
Minimisation	Maximisation
Profit unitaire	Terme de droite
Terme de droite	Coût unitaire
Ligne	Colonne
Colonne	Ligne
Contrainte $\geq$	Contrainte $\leq$

Exemple : Soit le programme primal (P) suivant

$$(P)\begin{cases} Min\ Z = 2x_1 + 3x_2 \\ 4x_1 + x_2 \geq 5 \\ 3x_1 + 2x_2 \geq 6 \\ x_1 + 2x_2 \geq 3 \\ x_1, x_2 \geq 0 \end{cases}$$

Le programme dual (D)

Pour simplifier le traitement des problèmes linéaires, il est courant de transformer la forme canonique en forme matricielle.

Cette transformation consiste à :

- Représenter les coefficients des variables dans une matrice A.
- Regrouper les variables de décision et les variables d'écart dans un vecteur colonne x.
- Mettre les second membres des contraintes dans un vecteur colonne b.
- Écrire la fonction objectif sous forme vectorielle $Z = c^T x$, où c contient les coefficients de la fonction objectif.

La forme matricielle permet ainsi d'appliquer plus facilement les méthodes algébriques, comme le simplexe, et de manipuler les systèmes de contraintes de manière structurée.

Étape 1 : Forme matricielle

On donne

$$A = \begin{pmatrix} 4 & 1 \\ 3 & 2 \\ 1 & 2 \end{pmatrix}, \quad x = \begin{pmatrix} x_1 \\ \\ x_2 \end{pmatrix}, \quad b = \begin{pmatrix} 5 \\ 6 \\ 3 \end{pmatrix} \quad \text{et} \quad c = \begin{pmatrix} 2 \\ \\ 3 \end{pmatrix}.$$

On cherche la transposée de la matrice c afin que le produit avec x soit possible c'est-à-dire $c^T = \begin{pmatrix} 2 & & 3 \end{pmatrix}$. On obtient la forme matricielle suivante :

$$(P) \begin{cases} Max \;\; Z = c^T x \\ Ax \leq b \\ x \geq 0 \end{cases}$$

Étape 2 : Programme dual

- On cherche la transposée de la matrice A.

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

- L'inégalité supérieure ou égale ($\geq$) devient inférieure ou égale ($\leq$).
- Les coefficients de la première ligne de la transposée de A représentent les coefficients de la première contrainte du programme dual.
- Les coefficients de la deuxième ligne de la transposée de A représentent les coefficients de la deuxième contrainte du programme dual.
- Les seconds membres du programme primal (P) deviennent les coefficients de la fonction objectif du programme dual (D).
- Les coefficients de la fonction objectif du programme primal (P) deviennent les seconds membres du programme dual (D).
- Si le programme primal (P) est un problème de minimisation, le programme dual (D) sera un problème de maximisation, et vice versa.

On obtient le programme dual (D) suivant :

$$(D)\begin{cases} Max\ Z' = 5y_1 + 6y_2 + 3y_3 \\ 4y_1 + 3y_2 + y_3 \leq 2 \\ y_1 + 2y_2 + 2y_3 \leq 3 \\ y_1, y_2, y_3 \geq 0 \end{cases}$$

Exemple : Résoudre le programme linéaire (P) en utilisant la méthode du simplexe

$$(P)\begin{cases} Min\ Z = 2x_1 + 3x_2 \\ 4x_1 + x_2 \geq 5 \\ 3x_1 + 2x_2 \geq 6 \\ x_1 + 2x_2 \geq 3 \\ x_1, x_2 \geq 0 \end{cases}$$

x_1, x_2 et x_3 sont des variables du modèle.

Étape 1 : Forme standard

$$(P)\begin{cases} Min\ Z = 2x_1 + 3x_2 \\ 4x_1 + x_2 - e_1 = 5 \\ 3x_1 + 2x_2 - e_2 = 6 \\ x_1 + 2x_2 - e_3 = 3 \\ x_1, x_2, e_1, e_2, e_3 \geq 0 \end{cases}$$

Étape 2 : Solution de base initiale

La solution de base est donnée à l'origine $x_1 = x_2 = 0$ et $e_1 = -5$, $e_2 = -6$ et $e_3 = -3$. La valeur de la fonction objectif $z = 0$.

Problème. Les variables de base obtenues sont négatives : $e_1 = -5$, $e_2 = -6$ et $e_3 = -3$, ce qui n'est pas faisable, car toutes les variables doivent être ≥ 0.

Cela montre que ce problème n'admet pas de solution de base faisable avec les seules variables d'excès.

- On peut utiliser le programme dual
- On peut introduire des variables artificielles (méthode du simplexe en deux phases) pour obtenir une solution de base faisable.

- Avantages de passer par le dual :

- Le programme primal (P) n'a pas de solution de base faisable immédiate à cause des variables d'excès négatives.
- Le programme dual (D) peut être formulé de manière à trouver directement une solution faisable, car les contraintes du dual deviennent souvent plus simples à gérer.
- Après avoir résolu le programme dual (D), la solution optimale du programme primal (P) peut être retrouvée grâce à la relation de dualité forte.

- Conclusion.

- Oui, utiliser le programme dual (D) est possible et souvent conseillé lorsque le programme primal n'a pas de solution de base faisable.

- Il suffit de :
  - Écrire le programme dual du problème.
  - Résoudre le programme dual par le simplexe (les contraintes sont maintenant toutes « inférieures ou égales » et faisables).
  - Récupérer la solution optimale du programme primal à partir de celle du dual.

Programme dual (D)

$$(D)\begin{cases} Max\ Z' = 5y_1 + 6y_2 + 3y_3 \\ 4y_1 + 3y_2 + y_3 \leq 2 \\ y_1 + 2y_2 + 2y_3 \leq 3 \\ y_1, y_2, y_3 \geq 0 \end{cases}$$

y_1, y_2 et y_3 sont les variables initiales du programme dual.

Étape 1 : Forme standard

$$(D)\begin{cases} Max\ Z' = 5y_1 + 6y_2 + 3y_3 \\ 4y_1 + 3y_2 + y_3 + u_1 = 2 \\ y_1 + 2y_2 + 2y_3 + u_2 = 3 \\ y_1, y_2, y_3, u_1, u_2 \geq 0 \end{cases}$$

- y_1, y_2 et y_3 sont les variables initiales du programme dual.
- u_1 et u_2 sont les variables de base du programme dual.

Exercice : Formuler le problème dual (D) de chacun des programmes linéaires suivants :

$$(P1)\begin{cases} Max\ Z = 2x_1 + 4x_2 + 3x_3 \\ 3x_1 + 4x_2 + 2x_3 \leq 60 \\ 2x_1 + x_2 + 2x_3 \leq 40 \\ x_1 + 3x_2 + 2x_3 \leq 80 \\ x_1, x_2, x_3 \geq 0 \end{cases}$$

$$(P2)\begin{cases} Max\ Z = 3x_1 + x_2 - 2x_3 \\ x_1 + 2x_2 \geq 10 \\ 3x_1 - x_2 + x_3 = 7 \\ x_1 + 3x_3 \leq 8 \\ x_1, x_2, x_3 \geq 0 \end{cases}$$

$$(P3)\begin{cases} Max\ Z = 10x_1 + 14x_2 \\ x_1 + x_2 \geq 12 \\ x_1 \geq 8 \\ x_2 \leq 6 \\ x_1, x_2 \geq 0 \end{cases} \quad et \quad (P4)\begin{cases} Max\ Z = 400x_1 + 350x_2 + 450x_3 \\ 2x_1 - 3x_2 + 2x_3 \leq 120 \\ 4x_1 + 3x_2 = 160 \\ 3x_1 - 2x_2 + 4x_3 \geq 100 \\ x_1, x_2, x_3 \geq 0 \end{cases}$$

Introduction à la programmation linéaire avec Pyhon (PuLP/SciPy)

La programmation linéaire est un ensemble de techniques utilisées en programmation mathématique, parfois appelée optimisation mathématique, pour résoudre des systèmes d'équations linéaires et d'inéquations tout en maximisant ou en minimisant une fonction linéaire. C'est important dans des domaines tels que le calcul scientifique, l'économie, les sciences techniques, la fabrication, les transports, l'armée, la gestion, l'énergie, etc.

L'écosystème Python propose plusieurs outils complets et puissants pour la programmation linéaire. Vous pouvez choisir entre des outils simples et complexes ainsi qu'entre des outils gratuits et commerciaux. Tout dépend de vos besoins.

- Qu'est-ce que la programmation linéaire et pourquoi c'est important ?
- Quels outils Python conviennent à la programmation linéaire ?
- Comment créer un modèle de programmation linéaire en Python ?
- Comment résoudre un problème de programmation linéaire avec Python ?

- La méthode de base pour résoudre les problèmes de programmation linéaire est appelée la méthode du simplexe, qui comporte plusieurs variantes. Une autre approche populaire est la méthode du point intérieur.
- Les problèmes de PL en nombres entiers mixtes sont résolus à l'aide de méthodes plus complexes et gourmandes en calcul, comme la méthode de branchement et de liaison, qui utilise la programmation linéaire sous le capot. Certaines variantes de cette méthode sont la méthode de branchement et de coupe, qui implique l'utilisation de plans de coupe, et la méthode de branchement et de prix.
- Il existe plusieurs outils Python adaptés et bien connus pour la PL et la programmation linéaire en nombres entiers mixtes. Certains d'entre eux sont open source, tandis que d'autres sont propriétaires. Le fait que vous ayez besoin d'un outil gratuit ou payant dépend de la taille et de la complexité de votre problème ainsi que du besoin de rapidité et de flexibilité.

- Il convient de mentionner que presque toutes les bibliothèques de programmation linéaire (PL) et de programmation linéaire à nombres entiers (PLNE) mixtes largement utilisées sont natives et écrites en Fortran ou C ou C++. En effet, la programmation linéaire nécessite un travail de calcul intensif avec des matrices (souvent volumineuses). De telles bibliothèques sont appelées solveurs. Les outils Python ne sont que des enveloppes autour des solveurs.
- Python convient à la création de wrappers (ou "enveloppe" en français) autour de bibliothèques natives car il fonctionne bien avec C/C++. Nous n'aurons pas besoin de C/C++ (ou Fortran) pour cette séquence, mais si nous souhaitons en savoir plus sur cette fonctionnalité intéressante, consultons les ressources suivantes :
    - Construire un module d'extension Python C.
    - Composants internes de CPython.
    - Extension de Python avec C ou C++

Fondamentalement, lorsque nous définissons et résolvons un modèle, nous utilisons des fonctions ou des méthodes Python pour appeler une bibliothèque de bas niveau qui effectue le travail d'optimisation proprement dit et renvoie la solution à votre objet Python.

- **Plusieurs bibliothèques Python gratuites sont spécialisées pour interagir avec des solveurs de programmation linéaire ou linéaire à nombres entiers mixtes :**
  - Optimisation SciPy et recherche de racine.
  - Pulp
  - Pyomo
  - CVXOPT

Dans cette séquence, nous utiliserons les deux bibliothèques SciPy et PuLP pour définir et résoudre des problèmes de programmation linéaire.

Implémentation en Python d'un PL

Dans cette séquence, nous utiliserons deux packages Python pour résoudre le problème de programmation linéaire décrit ci-dessous :

- **PuLP** est une API (Application Programming Interface) de programmation linéaire en Python permettant de formuler des problèmes et d'appeler des solveurs externes.
- **SciPy** est un package à usage général destiné au calcul scientifique avec Python.

où **API :** C'est une interface qui permet à deux systèmes (ou applications) de communiquer entre eux de manière standardisée et sécurisée.

En résumé : Une API est un pont entre nous (ou notre application) et un service.

- **PuLP** permet de choisir différents solveurs et de formuler des problèmes de manière plus naturelle. Le solveur par défaut utilisé par PuLP est le COIN-OR Branch and Cut Solver (CBC). Il s'appuie sur le solveur de programmation linéaire COIN-OR (CLP) pour les relaxations linéaires et sur la bibliothèque de génération de coupes COIN-OR (CGL) pour la production des coupes.

  PuLP est une bibliothèque open source pour la programmation linéaire en Python. Elle met à disposition tous les outils pour modéliser un problème et les résoudre en faisant appel à différents types de solveurs standards utilisant des algorithmes de résolution différents. C'est en fait un wrapper qui permet la formulation du problème en Python.
- **SciPy** est simple à configurer. Une fois installé, il fournit tout le nécessaire pour commencer. Son sous-package scipy.optimize peut être utilisé pour l'optimisation linéaire et non linéaire.

- Un autre excellent solveur open source est le kit de programmation linéaire GNU (GLPK). Certaines solutions commerciales et propriétaires bien connues et très puissantes sont Gurobi, CPLEX et XPRESS.
- En plus d'offrir une flexibilité dans la définition des problèmes et la possibilité d'exécuter différents solveurs, PuLP est moins compliqué à utiliser que des alternatives comme Pyomo ou CVXOPT, qui nécessitent plus de temps et d'efforts pour être maîtrisées.

Installation de SciPy et PuLP

Deux outils Python pour l'optimisation et le calcul scientifique.

- **PuLP (Python Linear Programming) :**
  - Utilité : Modélisation et résolution de problèmes de programmation linéaire (LP, MILP).
  - Installation : !pip install pulp
  - Vérification : import pulp
  - Importation : from pulp import LpProblem, LpMaximize/LpMinimize, LpVariable
- **SciPy (Scientific Python) :**
  - Utilité : Optimisation, algèbre linéaire, statistiques, traitement du signal, etc.
  - Installation : !pip install scipy
  - Vérification : import scipy
  - Dépendance : Nécessite numpy (installé automatiquement avec scipy).
  - Importation : Pour définir et résoudre des problèmes d'optimisation avec SciPy, nous devon importer scipy.optimize.linprog() : from scipy.optimize import linprog

Maintenant que nous avons importé linprog(), nous pouvons commencer à l'optimiser.

linprog() ne résout que les problèmes de minimisation (pas de maximisation) et n'autorise pas les contraintes d'inégalité avec le signe supérieur ou égal à ($\geq$). Pour contourner ces problèmes, nous devons modifier notre problème avant de lancer l'optimisation :

- Au lieu de maximiser $z = 2x_1 + 3x_2$ (par exemple), nous pouvons minimiser son négatif ($-z = 2x_1 - 3x_2$).
- Au lieu d'avoir le signe supérieur ou égal à, nous pouvons multiplier l'inégalité par -1 et obtenir le signe opposé inférieur ou égal à ($\leq$).
- **Exemple.** On considère le problème de programmation linéaire suivant :

$$\begin{cases} Max \;\; Z = 3x_1 + 2x_2 \\ 2x_1 + x_2 \leq 18 \\ 2x_1 + 3x_2 \geq 42 \\ 3x_1 + x_2 \leq 24 \\ -x_1 + 5x_2 = 15 \\ x_1, x_2 \geq 0 \end{cases}$$

Après avoir introduit ces changements, vous obtenez un nouveau système :

$$\begin{cases} Min \quad -Z = -3x_1 - 2x_2 \\ 2x_1 + x_2 \leq 18 \\ -2x_1 - 3x_2 \leq -42 \\ 3x_1 + x_2 \leq 24 \\ -x_1 + 5x_2 = 15 \\ x_1, x_2 \geq 0 \end{cases}$$

Ce système est équivalent à l'original et aura la même solution. La seule raison d'appliquer ces changements est de surmonter les limitations de SciPy liées à la formulation du problème.

Étape 2 : L'étape suivante consiste à définir les valeurs d'entrée

- $obj = [-3, -2]$: cette variable obj contient les coefficients de la fonction objectif.

- $lhs\_ineq = \Big[[2,1],[-2,-3],[3,1]\Big]$: cette variable contient les coefficients du côté gauche des contraintes d'inégalité
- $rhs\_ineq = [18,-42,24]$: cette variable contient les coefficients du côté droit des contraintes d'inégalité.
- $lhs\_eq = \Big[[-1,5]\Big]$: cette variable contient les coefficients du côté gauche de la contrainte d'égalité.
- $rhs\_eq = [15]$: cette variable contient les coefficients du côté droit de la contrainte d'égalité.

Remarques : Attention à l'ordre des lignes et des colonnes !

- L'ordre des lignes pour les côtés gauche et droit des contraintes doit être le même. Chaque ligne représente une contrainte.
- L'ordre des coefficients de la fonction objectif et des côtés gauches des contraintes doit correspondre. Chaque colonne correspond à une seule variable de décision

Étape 3 : L'étape suivante consiste à définir les limites de chaque variable dans le même ordre que les coefficients. Dans ce cas, ils sont tous deux compris entre zéro et l'infini positif :

- bnd = $\left[(0, \text{float("inf")}), (0, \text{float("inf")})\right]$: Ce sont les contraintes sur les valeurs que peuvent prendre les variables de décision x dans un problème d'optimisation.
- Chaque variable x_i a souvent des bornes inférieures (lower bound) et supérieures (upper bound).
- Variables non négatives (cas courant en optimisation) :
  - $x_1 \geq 0 \longrightarrow$ Bounds (bnd) : $[0, \infty]$
  - $x_2 \geq 0 \longrightarrow$ Bounds (bnd) : $[0, \infty]$

  Cette instruction est redondante car linprog() prend ces limites (de zéro à l'infini positif) par défaut.

En résumé : Bounds (bnd) of x = Limites (min/max) autorisées pour chaque variable.

Remarque :

Remarque : Au lieu de float("inf"), nous pouvons utiliser math.inf, numpy.inf, ou scipy.inf.

Étape 4 : Enfin, il est temps d'optimiser et de résoudre notre problème qui nous intéresse. nous pouvons le faire avec linprog() :

- opt (ou res) = linprog (c=obj, $A\_ub = lhs\_ineq$, $b\_ub = rhs\_ineq$, $A\_eq = lhs\_eq$, $b\_eq = rhs\_eq$, bounds=bnd, method="revised simplex")

Le paramètre c fait référence aux coefficients de la fonction objectif. A_ub et b_ub sont liés respectivement aux coefficients des côtés gauche et droit des contraintes d'inégalité. De même, A_eq et b_eq font référence à des contraintes d'égalité. Nous pouvons utiliser des limites pour fournir les limites inférieure et supérieure des variables de décision.

Nous pouvons utiliser le paramètre **method** pour définir la méthode de programmation linéaire que nous souhaitons utiliser. Il existe trois options :

- **method="interior-point"** sélectionne la méthode du point intérieur. Cette option est définie par défaut.
- **method="revised simplex"** sélectionne la méthode révisée du simplexe en deux phases.
- **method="simplex"** sélectionne l'ancienne méthode du simplexe en deux phases.

linprog() renvoie une structure de données avec ces attributs :

- .con correspond aux résidus des contraintes d'égalité.
- .fun est la valeur de la fonction objectif à l'optimum (si trouvée).
- .message est le statut de la solution.
- .nit est le nombre d'itérations nécessaires pour terminer le calcul.
- .slack correspond aux valeurs des variables slack, ou aux différences entre les valeurs des côtés gauche et droit des contraintes.

- .status est un entier compris entre 0 et 4 qui indique l'état de la solution, tel que 0 lorsque la solution optimale a été trouvée.
- .success est un booléen qui indique si la solution optimale a été trouvée.
- .x est un tableau NumPy contenant les valeurs optimales des variables de décision.

Nous pouvons accéder à ces valeurs séparément :

- opt.fun
- opt.success
- opt.x

C'est ainsi que nous obtenons les résultats de l'optimisation.

Les capacités de programmation linéaire de SciPy sont principalement utiles pour les petits problèmes. Pour des problèmes plus importants et plus complexes, nous pourrions trouver d'autres bibliothèques plus adaptées pour les raisons suivantes :

- SciPy ne peut pas exécuter divers solveurs externes.
- SciPy ne peut pas fonctionner avec des variables de décision entières.
- SciPy ne fournit pas de classes ou de fonctions facilitant la création de modèles. Nous devons définir des tableaux et des matrices, ce qui peut s'avérer une tâche fastidieuse et sujette aux erreurs en cas de problèmes importants.
- SciPy ne nous permet pas de définir directement les problèmes de maximisation. Nous devons les convertir en problèmes de minimisation.
- SciPy ne nous permet pas de définir des contraintes en utilisant directement le signe supérieur ou égal à. Nous devons plutôt utiliser la valeur inférieure ou égale à.

Heureusement, l'écosystème Python propose plusieurs solutions alternatives pour la programmation linéaire qui sont très utiles pour des problèmes plus importants. L'un d'eux est PuLP, que nous verrons en action dans la section suivante.

Utiliser PuLP

PuLP dispose d'une API de programmation linéaire plus pratique que SciPy. Nous n'avons pas besoin de modifier mathématiquement notre problème ni d'utiliser des vecteurs et des matrices. Tout est plus propre et moins sujet aux erreurs.

Comme d'habitude, nous commençons par importer ce dont nous avons besoin :

- from pulp import LpMaximize/LpMinimize, LpProblem, LpStatus, lpSum, LpVariable

Maintenant que PuLP est importé, nous pouvons résoudre notre problème.

Exemple.

On considère le problème de programmation linéaire suivant :

$$\begin{cases} Max \;\; Z = 3x_1 + 2x_2 \\ 2x_1 + x_2 \leq 18 \\ 2x_1 + 3x_2 \geq 42 \\ 3x_1 + x_2 \leq 24 \\ -x_1 + 5x_2 = 15 \\ x_1, x_2 \geq 0 \end{cases}$$

La première étape consiste à initialiser une instance de LpProblem pour représenter votre modèle :

- **installer la bibliothèque pulp.** !pip install pulp
- **Importer le package pulp.**

  from pulp import LpMaximize, LpProblem, LpVariable
- **Création du problème**

prob = LpProblem(name="*Programmation_Linaire*", sense=LpMaximize)

Nous utilisons le paramètre sense pour choisir d'effectuer une minimisation (LpMinimize ou 1, qui est la valeur par défaut) ou une maximisation (LpMaximize ou -1). Ce choix affectera le résultat de notre problème.

- **Initialiser les variables de décision.** Une fois que nous avons le modèle, nous pouvons définir les variables de décision comme instances de la classe LpVariable :
  - x_1 = LpVariable(name="x1", lowBound=0)
  - x_2 = LpVariable(name="x2", lowBound=0)

Nous devons fournir une limite inférieure avec lowBound=0 car la valeur par défaut est l'infini négatif. Le paramètre upBound définit la limite supérieure, mais nous pouvons l'omettre ici car sa valeur par défaut est l'infini positif.

- **Fonction économique.** prob += 3*x1 + 2*x2, "Z"

- **Définir les contraintes du problème**

  prob += 2 x1 + x2 <= 18, "contrainte1"

  prob += 2 x1+ 3 x2 <= 42, "contrainte2"

  prob += 3 x1 + x2 <= 24, "contrainte3"

  prob += -x1 + 5x2 = 15, "contrainte4"

- **Résolution du problème.** prob.solve()

- **Affichage des résultats.**
  - print("Status : la solution optimale est donnée par :", prob.status)
  - print("La valeur de la variable x1=", round(x1.value(), 2))
  - print("La valeur de la variable x2=", round(x2.value(), 2))
  - print("la valeur optimale z=", round(prob.objective.value(), 2))

Conclusion

Nous savons maintenant ce qu'est la PL et comment utiliser Python pour résoudre des problèmes de programmation linéaire. Nous avons également appris que les bibliothèques de programmation linéaire Python ne sont que des wrappers autour de solveurs natifs. Lorsque le solveur termine son travail, le wrapper renvoie l'état de la solution, les valeurs des variables de décision, les variables de marge, la fonction objectif, etc.

Dans cette séquence, nous avons appris à :

- Définir un modèle qui représente votre problème
- Créer un programme Python pour l'optimisation
- Exécuter le programme d'optimisation pour trouver la solution au problème.
- Récupérer le résultat de l'optimisation

Nous avons utilisé SciPy avec son propre solveur ainsi que PuLP avec CBC et GLPK, mais nous avons également appris qu'il existe de nombreux autres solveurs de programmation linéaire et wrappers Python. Nous sommes maintenant prêt à plonger dans le monde de la programmation linéaire!

A) Qu'est-ce qu'un vs code ?

- Qu'est-ce qu'un fichier .py ?*
  - *Extension .py* : Indique que le fichier contient du code Python.
  - Exécution : Le code est interprété par Python (ex : python *mon_script.py*).
- VS Code : Installer l'extension Python, cliquer sur "Run Python File".

B) Qu'est-ce qu'un Notebook Jupyter ?

- Outil interactif : Mélange code, texte, images, graphiques dans un même document.
- Utilisation : Analyse de données, prototypage, enseignement, rapport interactif.
- Lancé via : Serveur web local (ex : jupyter notebook).
- *Extension .ipynb*
- Data Science : Nettoyage, visualisation (Pandas, Matplotlib).
- Enseignement : Cours interactifs (maths, programmation).