

**REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET
POPULAIRE**



**MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA
RECHERCHE SCIENTIFIQUE**

**UNIVERSITE MOULOUD MAMMERI - TIZI OUZOU
FACULTE DES SCIENCES**

DEPARTEMENT DE MATHEMATIQUES

MEMOIRE DE FIN D'ETUDE

**En vue de l'obtention du diplôme de MASTER en Mathématiques Appliquées à la
Gestion**

Thème :

**Programmation mathématique floue et
applications**

Préparé par :

**HACOUR Sadia
KEBBAL Samia**

Encadré par:

**M^r CHEBBAH Mohammed
Maître Assistant A**

Devant le jury composé de :

M^r KASDI Kamel M.A.A President

M^r CHEBBAH Mohammed M.A.A Rapporteur

M^{me} LESLOUS Fadila M.A.A Examineur

Année universitaire 2017 - 2018

Remerciements

Nous remercions Dieu tout puissant qui nous a ouvert les portes du savoir et qui nous a éclairé le chemin et permis de mener à bien ce travail.

*Nous tenons à remercier **M^r CHEBBAH** de nous avoir fait l'honneur d'être le promoteur de ce mémoire.*

Nous remercions également les membres du jury pour avoir accepté d'évaluer notre travail.

Un grand merci à toutes les personnes qui nous ont soutenues de près ou de loin au cours de la réalisation de ce modeste travail.

Enfin, nous remercions chaleureusement nos chers enseignants pour leur soutien et leurs encouragements.

Dédicaces

Je dédie ce modeste travail

À mes chers parents

qui m'ont toujours comblé de leur amour, leur bonté et leur grande affection.

À mes chers frères

Mohammed, Juba, Rayan et Hakim

À mes chers grand parents

À mes chères tantes et mes chers oncles

À mes chères amies, spécialement

Hakima, Chahira, Lamia et Malika

À ma binome et sa famille

Sadia HACHOUR

Dédicaces

Je dédie ce modeste travail

À mes chers parents

qui m'ont toujours comblé de leur amour, leur bonté et leur grande affection.

À mes chers grand parents

À mes chers frères

Rafik, Fouad et Djugurtha

À mes chères soeurs

Siham et Sabrina

À mes très chères nièces

Ouarda et Yasmine

À mes chères tantes et mon cher oncle

À mon fiancé et sa famille

À mes chères amies spécialement

Nazli, Ouahiba, Dalida, Mélissa, Asma et Lydia

À ma binome et sa famille

Samia KEBBAL

Table des matières

Introduction générale	3
1 Théorie des ensembles flous	4
1.1 Introduction	4
1.2 Ensemble flou	4
1.2.1 Définition	4
1.3 Concepts usuels	5
1.4 Nombre flou	6
1.4.1 Définition d'un nombre flou	6
1.4.2 Opérations sur les ensembles flous	7
1.4.3 Nombre flou de type L-R	8
1.4.4 Intervalle flou	9
1.4.5 Nombre flou de type triangulaire	9
1.4.5.1 Opérations sur les nombres flous de type triangulaire	10
1.4.5.2 Comparaison de deux nombres flous triangulaire	11
1.4.6 Nombre flou de type trapézoïdal	11
1.4.6.1 Opérations sur les nombres flous de type trapézoïdal	11
1.4.6.2 Comparaison de deux nombres flous trapézoïdaux	12
1.5 Conclusion	12
2 Programmation Linéaire	13
2.1 Introduction	13
2.2 Définition et modélisation d'un programme linéaire	13
2.2.1 Définition	13
2.2.2 Modélisation	13
2.3 Forme générale d'un programme linéaire	14
2.3.1 Forme canonique et forme standard d'un programme linéaire :	15
2.4 Domaine de solutions réalisables et solution optimale	16
2.5 Propriétés géométriques	16
2.6 La méthode du simplexe	16
2.6.1 Base et solution de base	16
2.6.2 Algorithme du simplexe	17
2.6.3 Tableau du simplexe	18
2.7 Initialisation de l'algorithme du simplexe	19
2.7.1 Méthode des deux phases	20
2.7.2 M-méthode	22
2.8 Conclusion	24

3	Programmation linéaire floue	25
3.1	Introduction	25
3.2	Programmation flexible	25
3.2.1	Décision dans un environnement flou	25
3.2.2	Résolution d'un programme linéaire flexible	25
3.3	Programmation robuste	31
3.3.1	Programmation linéaire inexacte (Solster)	31
3.3.2	Résolution d'un Programme robuste	32
3.3.3	Cas des nombres flous	33
3.3.4	Cas des nombres flous de types L-R	33
3.4	Conclusion	36
4	Programmation mathématique	37
4.1	Introduction	37
4.2	Notion topologique	37
4.2.1	Ensembles convexes	37
4.2.2	Fonctions convexes	38
4.3	Définition d'un programme mathématique	39
4.3.1	Notion de solution	40
4.4	Classification d'un programme mathématique	41
4.5	Condition d'optimalité(sans contraintes)	41
4.6	Conditions d'optimalité (avec contraintes)	42
4.7	Qualification des contraintes	43
4.8	Programmation non linéaire floue flexible	44
4.9	Cas particulier de la programmation mathématique	46
4.9.1	Problème quadratique	46
4.9.2	Résolution d'un problème quadratique convexe.	46
4.9.2.1	Méthode quadratique de Wolfe (1959)	46
4.9.3	Algorithme	48
5	Application sur lingo	54
5.1	Introduction	54
5.2	Installation du logiciel	54
	Conclusion générale	62
	Bibliographie	62

Introduction générale

La programmation mathématique se propose pour l'objet d'étude théorique des problèmes d'optimisation ainsi que la conception et la mise en œuvre des algorithmes de résolution.

La présence du terme "programmation" dans le nom donné à la discipline peut s'expliquer historiquement par le fait que les premières recherches et les premières applications se sont développées dans le contexte de l'économie et de la recherche opérationnelle.

C'est ainsi que G.B.Dantzing propose en 1949 le terme de programmation linéaire pour l'étude des problèmes théoriques et algorithmes liés à l'optimisation des fonctions linéaires sous contraintes linéaires. Dans le même sens Kuhn et Tucker ont proposé la programmation non linéaire avec ou sans contraintes.

Dans le cas où les données sont mal connues ou imprécises de nature floue, on parlera alors de programmes linéaires flous qui sont considérés comme les meilleurs outils pour traiter des études de prise de décisions dans un environnement imprécis. Les connaissances imprécises n'ont été prises en considération qu'à partir de 1965, lorsque Zadeh, professeur à l'université de Californie de Berkeley, a introduit la notion de sous ensemble flou. A partir de l'idée d'appartenance partielle à une classe de catégorie aux limites mal définies, dans une généralisation de la théorie classique des ensembles admettant des situations intermédiaires entre le tout et le rien.

Dans tous les problèmes de décision, l'obtention des données précises du modèle traité est évidemment essentielle. Dans de nombreuses situations les données du problème ou certaines d'entre elles sont de nature imprécises et difficile à appréhender exactement. L'approche floue, introduite par Zadeh, est un outil adéquat pour modéliser cette imprécision.

- Le premier travail réalisé sur un problème de décision dans un environnement flou revient à Belleman et Zadeh. l'apport fondamental de cet article et de souligner que l'agrégation des critères peut être vu comme un problème de combinaison d'ensemble flous à l'aide d'opération ensembliste floues. En se basant sur ce principe, la *Programmation Mathématique Floue* a connu ces dix dernières années de nombreux développements et a donné lieu à de très nombreuses publications et ouvrages spécialisés .

Ce mémoire est organisée comme suit :

— *Dans le premier chapitre*, nous allons introduire la théorie des ensembles flous, donner les définitions et les concepts de base facilitant la compréhension de ce qui suivra .

— *Dans le deuxième chapitre*, nous parlerons de la programmation linéaire déterministe et la méthode de résolution qui est la méthode du simplexe classique de Dantzing.

— *Le troisième chapitre*, nous aborderons la programmation linéaire floue, nous donnerons la forme générale d'un programme linéaire flou, nous montrons la façon de procéder pour défuzzifier les contraintes et se ramener à un programme linéaire déterministe.

— *Le quatrième chapitre*, contient quelques notions topologiques, la convexité (ensemble convexe, fonction convexe), programme convexe, les conditions nécessaire de K-K-T, extension à des problèmes avec contraintes d'égalités et d'inégalités.

Chapitre 1

Théorie des ensembles flous

1.1 Introduction

La théorie des ensembles flous est apparue en 1965 à Berkeley[9], dans le laboratoire de Lotfi Zadeh. Cette théorie permet la formalisation des imprécisions dues à une connaissance globale d'un système très complexe et l'expression du comportement d'un système par des mots. Elle permet donc la standardisation de la description d'un système et du traitement de données aussi bien numérique qu'exprimées symboliquement par des qualifications linguistiques. Autrement dit, la théorie des ensembles flous est une modélisation mathématique du langage naturel. Son idée de base est la modification de la traditionnelle fonction caractéristique d'un ensemble de façon à introduire des degrés dans la relation traditionnelle d'appartenance.

1.2 Ensemble flou

Origine de la théorie des ensembles flous

Les catégories manipulées par le langage sont rarement tout ou rien. Leur représentation par des ensembles classique est souvent mal adaptée.

Idée de Zadeh

Remplacer l'appartenance à un ensemble classique par un degré d'appartenance pouvant prendre toutes les valeurs comprise entre 0 et 1.

Les ensembles flous peuvent être construits sur les ensembles ordinaires en modifiant la définition traditionnelle de la fonction caractéristique d'un ensemble.

Soit X un ensemble, appelé univers dont les éléments sont notés par x .

1.2.1 Définition

Un sous-ensemble flou $\tilde{A}$ dans un ensemble X est défini par la donnée d'une application :

$$\mu_{\tilde{A}} : X \rightarrow [0, 1]$$

$\mu_{\tilde{A}}$ est appelée fonction d'appartenance du sous-ensemble flou $\tilde{A}$.

$\forall x \in X$, $\mu_{\tilde{A}}(x)$ s'interprète comme :

1. Le degré d'appartenance de x à A
2. Le niveau de compatibilité avec un concept donné
3. Le niveau de similarité avec un prototype, etc

Remarque 1.1

On note $\tilde{A} = (X, \mu_{\tilde{A}}) = \{(x, \mu_{\tilde{A}}(x)); x \in X\}$

Exemple 1.1

On se propose de mesurer l'acuité visuelle (moyenne des deux yeux) des individus d'une certaine localité X .

Soit A l'ensemble des individus en cette localité ayant une bonne acuité visuelle. Cet ensemble a un contour mal défini.

En effet, il y a des individus dont l'acuité visuelle est égale à 1, 0.8, 0.6 ou toutes autre valeur comprise entre 0 et 1.

Dans ce cas on dit que A est un sous ensemble flou on le note par $\tilde{A}$.

1.3 Concepts usuels

• Sous ensembles de niveau α [1]

Un sous ensemble de niveau α noté $\tilde{A}^\alpha$ est l'ensemble :

$$\tilde{A}^\alpha = \{x \in X / \mu_{\tilde{A}}(x) \geq \alpha\}$$

Il contient tous les éléments de X qui sont compatibles avec $\tilde{A}$ à un niveau au moins égale à α .

Exemple 1.2

$$X = \{x_1, x_2, x_3, x_4, x_5\}$$

$$\tilde{A} = \{(x_1, 0.9), (x_2, 1), (x_3, 0.5), (x_4, 0.1), (x_5, 0.3)\}$$

$$\tilde{A}^{0.5} = \{x_1, x_2, x_3\}$$

• Sous ensembles de niveau α strict [1]

Un sous ensemble de niveau α strict noté $\tilde{A}^{\overline{\alpha}}$ est l'ensemble :

$$\tilde{A}^{\overline{\alpha}} = \{x \in X / \mu_{\tilde{A}}(x) > \alpha\}$$

pour l'exemple précédent :

$$\tilde{A}^{\overline{0.5}} = \{x_1, x_2\}$$

• Support de $\tilde{A}$ [1]

Le support de $\tilde{A}$ noté $\text{Supp } \tilde{A}$, c'est l'ensemble des éléments de X appartiennent tant soit peu à $\tilde{A}$ c'est-à-dire :

$$\text{Supp } \tilde{A} = \{x \in X / \mu_{\tilde{A}}(x) > 0\}$$

• Hauteur de $\tilde{A}$ [1]

La hauteur de $\tilde{A}$ notée $\text{Haut } \tilde{A}$ est définie comme suit :

$$\text{Haut } \tilde{A} = \text{Sup } \mu_{\tilde{A}}(x)$$

- **Sous-ensemble flou normalisé** [1]

$\tilde{A}$ est dite normalisé s'il existe $x \in X$ tel que :

$$\mu_{\tilde{A}}(x) = 1$$

Exemple 1.3

$$\tilde{A} = \{(x_1, 0.2), (x_2, 1), (x_3, 0), (x_4, 0.1), (x_5, 0.3)\}$$

$\tilde{A}$ est normalisé

- **Sous-ensemble flou convexe** [1]

$\tilde{A}$ est dite convexe si quelque soient x_1 et x_2 appartenant à X et $\lambda \in [0, 1]$ on a :

$$\mu_{\tilde{A}}(\lambda x_1 + (1 - \lambda) x_2) \geq \min [\mu_{\tilde{A}}(x_1) ; \mu_{\tilde{A}}(x_2)]$$

- **Noyau d'un sous-ensemble $\tilde{A}$** [4]

Le noyau d'un sous ensemble flou $\tilde{A}$ de X , noté $Noy(\tilde{A})$ est l'ensemble de tous les éléments qui lui appartiennent totalement (avec un degré 1) on a :

$$Noy(\tilde{A}) = \{x \in X / \mu_{\tilde{A}}(x) = 1\}$$

1.4 Nombre flou

1.4.1 Définition d'un nombre flou

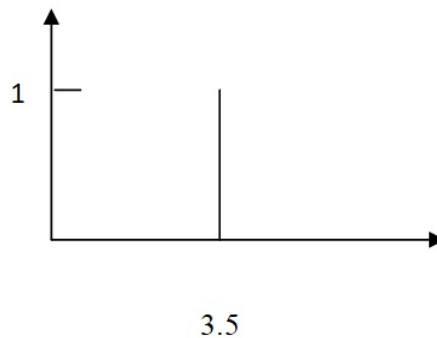
Un nombre flou est un ensemble flou $\tilde{A}$ convexe et normalisé, [13] de l'ensemble des nombres réels ($X = \mathbb{R}$) .

Remarque 1.2

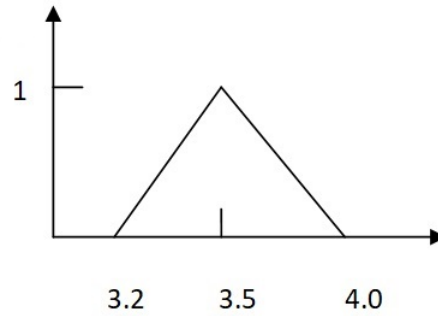
Si le $Noy(\tilde{A})$ est un intervalle de $\mathbb{R}$, on parle alors d'intervalle flou.

Exemple 1.4

- **Nombre réel**



- **Nombre flou**



:

Figure 1.1- Comparaison entre un nombre flou et un nombre réel

1.4.2 Opérations sur les ensembles flous

Les opérations sur les ensembles flous sont généralement des extensions des opérations connues sur les ensembles classiques (inclusion, égalité, réunion, intersection et complément). Elles s'appliquent d'ailleurs aux ensembles classiques lorsque les fonctions d'appartenance se réduisent à des valeurs de 0 ou 1 [4].

- **Inclusion**

Un ensemble flou $\tilde{A}$ est inclu dans un autre ensemble flou $\tilde{B}$, cela se traduit par :

$$\tilde{A} \subseteq \tilde{B} \iff \mu_{\tilde{A}}(x) \leq \mu_{\tilde{B}}(x), \forall x \in X$$

- **Égalité**

Soient deux ensembles flous $\tilde{A}$ et $\tilde{B}$ d'un univers X . On dit que $\tilde{A}$ et $\tilde{B}$ sont égaux si et seulement si leurs fonctions d'appartenance ont la même valeur en tout point x de X :

$$\mu_{\tilde{A}}(x) = \mu_{\tilde{B}}(x), \forall x \in X$$

- **Complément**

Le complément d'un ensemble flou $\tilde{A}$ d'un univers X , est l'ensemble flou $\overline{\tilde{A}}$ dont la fonction d'appartenance est :

$$\mu_{\overline{\tilde{A}}}(x) = 1 - \mu_{\tilde{A}}(x)$$

- **Réunion**

La réunion de deux ensembles flous $\tilde{A}$ et $\tilde{B}$ de même référentiel X est un ensemble flou noté $\tilde{A} \cup \tilde{B}$ dont la fonction d'appartenance est :

$$\mu_{\tilde{A} \cup \tilde{B}}(x) = \max \{ \mu_{\tilde{A}}(x), \mu_{\tilde{B}}(x) \}, \forall x \in X$$

- **Intersection**

L'intersection de deux ensembles flous $\tilde{A}$ et $\tilde{B}$ de même référentiel X est l'ensemble flou noté $\tilde{A} \cap \tilde{B}$ dont la fonction d'appartenance $\mu_{\tilde{A} \cap \tilde{B}}$ est définie par :

$$\mu_{\tilde{A} \cap \tilde{B}}(x) = \min \{ \mu_{\tilde{A}}(x), \mu_{\tilde{B}}(x) \}, \forall x \in X$$

Exemple 1.5

$$X = \{x_1, x_2, x_3, x_4, x_5\}$$

$$\tilde{A} = \{(x_1, 1), (x_2, 0.3), (x_3, 0.7), (x_4, 0.4), (x_5, 0)\}$$

$$\tilde{B} = \{(x_1, 0.9), (x_2, 0.2), (x_3, 0.1), (x_4, 0.6), (x_5, 0.5)\}$$

- $\tilde{A} \subsetneq \tilde{B}$
- $\tilde{A} \neq \tilde{B}$
- $\tilde{\tilde{A}} = \{(x_1, 0), (x_2, 0.7), (x_3, 0.3), (x_4, 0.6), (x_5, 1)\}$
- $\mu_{\tilde{A} \cup \tilde{B}} = \{(x_1, 1), (x_2, 0.3), (x_3, 0.7), (x_4, 0.6), (x_5, 0.5)\}$
- $\mu_{\tilde{A} \cap \tilde{B}} = \{(x_1, 0.9), (x_2, 0.2), (x_3, 0.1), (x_4, 0.4), (x_5, 0)\}$

1.4.3 Nombre flou de type L-R

Un nombre flou $\tilde{A}$ est de représentation L-R[15, 18], s'il existe deux fonctions L et R telle que sa fonction d'appartenance est définie par :

$$\alpha > 0, \beta > 0$$

$$\mu_{\tilde{A}}(x) = \begin{cases} L\left(\frac{m-x}{\alpha}\right) & \text{pour } x \leq m \\ R\left(\frac{x-m}{\beta}\right) & \text{pour } x \geq m \end{cases} \quad (1.1)$$

où :

L et R sont des fonctions dites de référence du nombre flou $\tilde{A}$ vérifiant les propriétés suivantes :

- L et R fonctions non croissantes sur $[0, +\infty[$;
- L et R fonctions symétriques : $L(x) = L(-x)$; $R(x) = R(-x)$, $\forall x$;
- $L(0) = R(0) = 1$;

On note $\tilde{A} = (m ; \alpha ; \beta)_{L-R}$.

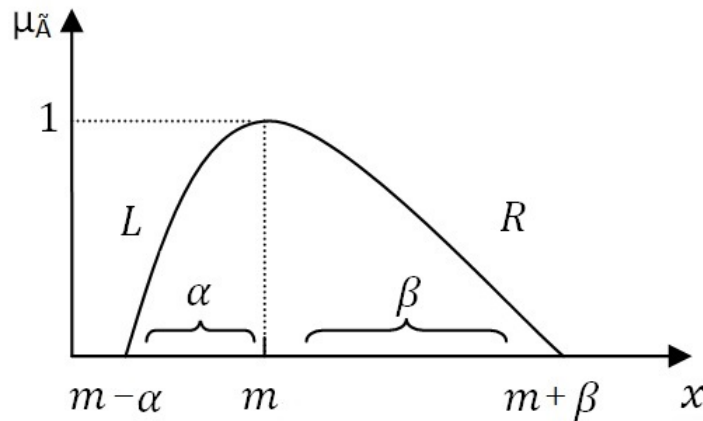


Figure 1.2- Représentation d'un nombre flou de type L-R

1.4.4 Intervalle flou

Un nombre flou plat de type L-R[18], ou intervalle flou, est tel qu'il existe $m, n \in \mathbb{R}$, avec $m < n$ de sorte que :

$\mu_{\tilde{A}}(x) = 1, \forall x \in [m, n]$ Sa fonction d'appartenance est définie par :

$\alpha > 0, \beta > 0$

$$\mu_{\tilde{A}}(x) = \begin{cases} L(\frac{m-x}{\alpha}) & \text{si } x \leq m \\ 1 & \text{si } m \leq x \leq n \\ R(\frac{x-n}{\beta}) & \text{si } x \geq n \end{cases} \quad (1.2)$$

Nous désignerons un tel nombre flou $\tilde{A}$ de type L-R par :

$$\tilde{A} = (m, n, \alpha, \beta)_{L-R}$$

où :

α et β sont les écarts à gauche et à droite de $\tilde{A}$ respectivement, m et n sont les valeurs modales inférieures et supérieures de $\tilde{A}$ respectivement ou bien la moyenne à gauche et à droite de $\tilde{A}$ respectivement, $[m, n]$ est le noyau de $\tilde{A}$.

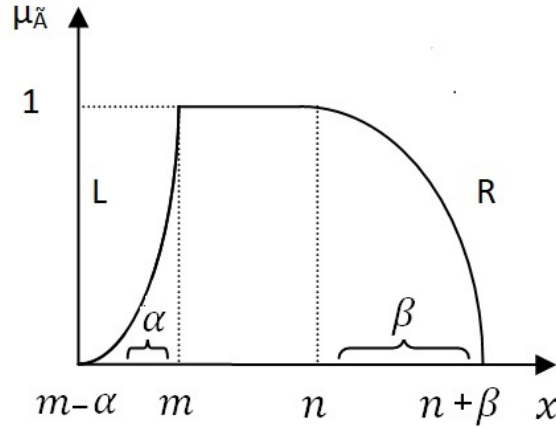


Figure 1.3 - Représentation d'un nombre flou plat de type L-R

Remarque 1.3

Il existe plusieurs types de nombres flous de type L-R. Lorsque les fonctions de références L et R sont linéaires, on parle alors de nombre flou de type triangulaire ou de type trapézoïdal.

1.4.5 Nombre flou de type triangulaire

Un nombre flou est dit de type triangulaire noté (a, α, β) si sa fonction d'appartenance est définie par [6] :

$\alpha > 0, \beta > 0$

$$\mu_{\tilde{A}}(x) \begin{cases} \frac{x - a + \alpha}{\alpha} & \text{si } a - \alpha \leq x \leq a \\ 1 & \text{si } x = a \\ \frac{a + \beta - x}{\alpha} & \text{si } a \leq x \leq a + \beta \end{cases} \quad (1.3)$$

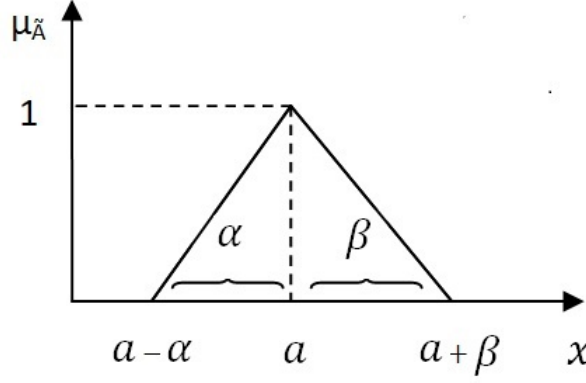


Figure 1.4 - Représentation d'un nombre flou triangulaire $(a; \alpha; \beta)$

1.4.5.1 Opérations sur les nombres flous de type triangulaire

Soient $\tilde{A}$ et $\tilde{B}$ deux nombres flous de type triangulaire .

- **Multiplications scalaires** [6]

Soit le nombre flou triangulaire $\tilde{A} = (a, \alpha_1, \beta_1)$

Il vient :

$$\begin{cases} \text{Si } \lambda > 0, \lambda \in \mathbb{R} : \lambda \otimes \tilde{A} = (\lambda a, \lambda \alpha_1, \lambda \beta_1) \\ \text{Si } \lambda < 0, \lambda \in \mathbb{R} : \lambda \otimes \tilde{A} = (\lambda a, -\lambda \beta_1, -\lambda \alpha_1) \end{cases}$$

- **Addition** [6]

Soient deux nombres flous triangulaire $\tilde{A} = (a, \alpha_1, \beta_1)$ et $\tilde{B} = (b, \alpha_2, \beta_2)$

Il vient :

$$\tilde{A} \oplus \tilde{B} = (\mathbf{a} + \mathbf{b}, \alpha_1 + \alpha_2, \beta_1 + \beta_2)$$

- **Soustraction** [6]

Étant donnée $\tilde{A} = (a, \alpha_1, \beta_1)$ et $\tilde{B} = (b, \alpha_2, \beta_2)$ deux nombres flous triangulaire .

Il vient :

$$-\tilde{B} = -(b, \alpha_2, \beta_2) = (-b, \beta_2, \alpha_2)$$

et

$$\tilde{A} \ominus \tilde{B} = (a - b, \alpha_1 + \beta_2, \beta_1 + \alpha_2).$$

1.4.5.2 Comparaison de deux nombres flous triangulaire

Soient deux nombres flous triangulaire $\tilde{A} = (a, \alpha_1, \beta_1)$ et $\tilde{B} = (b, \alpha_2, \beta_2)$

$$* \tilde{A} = \tilde{B} \Leftrightarrow a = b, \alpha_1 = \alpha_2, \beta_1 = \beta_2.$$

$$* \tilde{A} \leq \tilde{B} \Leftrightarrow a \leq b, a - \alpha_1 \leq b - \alpha_2, a + \beta_1 \leq b + \beta_2.$$

1.4.6 Nombre flou de type trapézoïdal

Un nombre flou $\tilde{A}$ est dit de type trapézoïdal noté $(a^L, a^U, \alpha, \beta)$ si sa fonction d'appartenance est donnée par[13] :

$$\mu_{\tilde{A}}(x) = \begin{cases} \frac{x - a^L + \alpha}{\alpha} & \text{Si } a^L - \alpha \leq x \leq a^L, \\ 1 & \text{Si } a^L \leq x \leq a^U, \\ \frac{a^U + \beta - x}{\beta} & \text{Si } a^U \leq x \leq a^U + \beta, \end{cases} \quad (1.4)$$

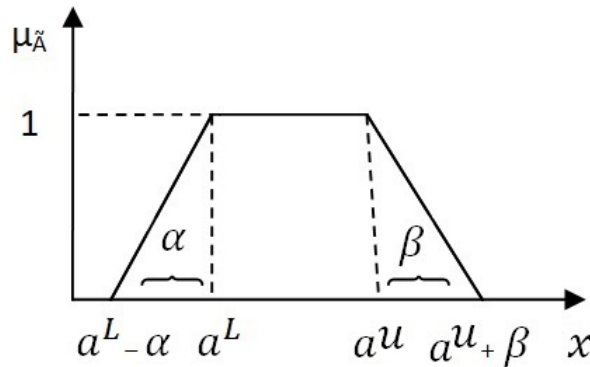


Figure 1.5 - Représentation d'un nombre flou trapézoïdal

1.4.6.1 Opérations sur les nombres flous de type trapézoïdal

Soient $\tilde{A}$ et $\tilde{B}$ deux nombres flous trapézoïdaux.

- **Multiplication scalaire** [14]

Soit le nombre flou $\tilde{A} = (a^L, a^U, \alpha_1, \beta_1)$ de type trapézoïdal .

Il vient :

$$* \text{ Si } \lambda > 0, \lambda \in \mathbb{R} : \lambda \otimes \tilde{A} = (\lambda a^L, \lambda a^U, \lambda \alpha_1, \lambda \beta_1) .$$

$$* \text{ Si } \lambda < 0, \lambda \in \mathbb{R} : \lambda \otimes \tilde{A} = (\lambda a^U, \lambda a^L, -\lambda \beta_1, -\lambda \alpha_1) .$$

- **Addition** [14]

Soient deux nombres flous $\tilde{A} = (a^L, a^U, \alpha_1, \beta_1)$ et $\tilde{B} = (b^L, b^U, \alpha_2, \beta_2)$ de type trapézoïdaux .

Il vient :

$$\tilde{A} \oplus \tilde{B} = (a^L + b^L, a^U + b^U, \alpha_1 + \alpha_2, \beta_1 + \beta_2)$$

- **Soustraction** [14]

Étant donnée deux nombres flous trapézoïdaux $\tilde{A} = (a^L, a^U, \alpha_1, \beta_1)$ et $\tilde{B} = (b^L, b^U, \alpha_2, \beta_2)$.

Il vient :

$$\tilde{B} = -(b^L, b^U, \alpha_2, \beta_2) = (-b^U, -b^L, \beta_2, \alpha_2) ,$$

$$\tilde{A} \ominus \tilde{B} = (a^L - b^U, a^U - b^L, \alpha_1 + \beta_2, \beta_1 + \alpha_2) .$$

1.4.6.2 Comparaison de deux nombres flous trapézoïdaux

Soient deux nombres flous trapézoïdaux $\tilde{A} = (a^L, a^U, \alpha_1, \beta_1)$ et $\tilde{B} = (b^L, b^U, \alpha_2, \beta_2)$.

$$* \tilde{A} = \tilde{B} \Leftrightarrow a^L = b^L, a^U = b^U, \alpha_1 = \alpha_2, \beta_1 = \beta_2.$$

$$* \tilde{A} \leq \tilde{B} \Leftrightarrow a^L \leq b^L, a^U \leq b^U, a^L - \alpha_1 \leq b^L - \alpha_2, a^U + \beta_1 \leq b^U + \beta_2.$$

Remarque 1.4

On note par $F(\mathbb{R})$ l'ensemble des nombres flous de type trapézoïdal .

1.5 Conclusion

Nous avons présenté dans ce chapitre, quelques notions de bases sur la logique floue en introduisant les concepts de base des ensembles et des nombres flous .

Chapitre 2

Programmation Linéaire

2.1 Introduction

À partir de la fin de la Seconde Guerre mondiale, des nouvelles méthodes permirent de résoudre des problèmes complexes là où les méthodes classiques échouaient. Ces méthodes furent connues sous le nom de programmation linéaire, développées principalement par George B. Dantzig, mathématicien américain et créateur de la méthode du Simplexe.

Dantzig, outre la programmation linéaire, étudié entre autres la programmation mathématique, la prise de décision et les modèles de planification à large échelle. L'impact de son œuvre fut considérable en gestion et en économie et ses méthodes restent totalement d'actualité.

Ce chapitre permet de se familiariser avec les notions relatives à la programmation linéaire[19].

2.2 Définition et modélisation d'un programme linéaire

2.2.1 Définition

La programmation Linéaire est un programme consistant à minimiser ou à maximiser une fonction objectif à plusieurs variables de décision soumises à un ensemble de contraintes exprimées sous forme d'équations linéaires[19].

2.2.2 Modélisation

Un modèle mathématique est une traduction d'une observation dans le but de lui appliquer les outils, les techniques et les théories mathématiques. Puis généralement, en sens inverse, la traduction des résultats mathématiques obtenus en prédiction ou opérations dans le monde réel.

En Recherche Opérationnelle, modéliser un problème consiste à identifier les variables intrinsèques, les différentes contraintes auxquelles sont soumises ces variables et l'objectif visé (optimisation)[13].

Exemple 2.1

L'intendant d'un lycée doit composer un menu qui doit contenir un minimum d'éléments nutritifs et qui doit être le moins coûteux possible. On se limite à une situation simple, deux denrées alimentaires principales D_1 , D_2 et trois éléments nutritifs, les vitamines V , les calories C et les protéines P . Le tableau suivant indique le nombre d'éléments nutritifs par unité d'aliment :

	V	C	P
D_1	1	1	3
D_2	5	2	2

► Une unité de D_1 contient 1 unité de V , 1 unité de C et 3 unités de P .

► Une unité de D_2 contient 5 unités de V , 2 unités de C et 2 unités de P .

Modèle mathématique

– Contraintes diététiques. Le menu doit comporter au minimum 5 unités de V , 4 unités de C , 6 unités de P . Les coûts unitaires sont 20 pour D_1 , 25 pour D_2 .

– Réalisation du menu. Un menu contenant x_1 unités de D_1 , x_2 unités de D_2 est réalisable si le couple (x_1, x_2) vérifie :

$$\begin{cases} x_1 + 5x_2 \geq 5 \\ x_1 + 2x_2 \geq 4 \\ 3x_1 + x_2 \geq 6 \\ x_1 \geq 0, x_2 \geq 0 \end{cases}$$

– Le programme linéaire :

le problème consiste à déterminer deux nombres x_1 et x_2 tels que :

$$\begin{cases} Z(x_1, x_2) = 20x_1 + 25x_2 \\ s.c \\ x_1 + 5x_2 \geq 5 \\ x_1 + 2x_2 \geq 4 \\ 3x_1 + x_2 \geq 6 \\ x_1 \geq 0, x_2 \geq 0 \end{cases} \quad (2.1)$$

ou Z est la fonction objectif à minimiser.

2.3 Forme générale d'un programme linéaire

Un programme linéaire a la forme générique suivante[18] :

$$\begin{cases} \max(\min) Z = \sum_{j=1}^n c_j x_j \\ s.c \\ \sum_{i=1}^n a_{ij} x_j \begin{cases} = \\ \leq \\ \geq \end{cases} b_i, i = \overline{1, m} \\ x_j \geq 0 \quad \forall j = \overline{1, n} \end{cases} \quad (2.2)$$

où :

c_j, a_{ij}, b_i : des nombres réels.

Il comporte n variables non négatives, m contraintes d'inégalités ou d'égalités et la fonction objectif à optimiser.

On peut aussi représenter ce programme par l'écriture matricielle suivante :

$$\left\{ \begin{array}{l} \max(\min) Z = c'x \\ s.c \\ Ax \left\{ \begin{array}{l} = \\ \leq \\ \geq \end{array} \right\} b \\ x \geq 0 \end{array} \right. \quad (2.3)$$

où :

$$A = (a_{ij}, 1 \leq i \leq m, 1 \leq j \leq n)$$

$$c = (c_j, 1 \leq j \leq n)$$

$$b = (b_i, 1 \leq i \leq m)$$

$$x = (x_j, 1 \leq j \leq n)$$

2.3.1 Forme canonique et forme standard d'un programme linéaire :

Un programme linéaire s'écrit sous forme canonique suivante[18] :

$$\left\{ \begin{array}{l} \max(\min) Z = \sum_{j=1}^n c_j x_j \\ s.c \\ \sum_{j=1}^n a_{ij} x_j \leq b_i, i = \overline{1, m} \\ x_j \geq 0; \quad \forall j = \overline{1, n} \end{array} \right. \quad (2.4)$$

Si par contre, en dehors des contraintes de non-négativité toutes les contraintes sont des égalités, on dit que le programme linéaire, est mis sous «forme standard». On peut toujours mettre un programme linéaire quelconque sous forme standard en ajoutant des variables supplémentaires appelées «variables d'écart» à partir de la forme canonique.

$$\left\{ \begin{array}{l} \max(\min) Z = \sum_{j=1}^n c_j x_j \\ s.c \\ \sum_{j=1}^n a_{ij} x_j = b_i, i = \overline{1, m} \\ x_j \geq 0 \quad \forall j = \overline{1, n} \end{array} \right. \quad (2.5)$$

2.4 Domaine de solutions réalisables et solution optimale

Définitions[3]

Solution réalisable : tout vecteur x qui satisfait les contraintes fonctionnelles et les contraintes de non négativité est appelé solution réalisable aux modèles de programmation linéaire (2.4) et (2.5).

Domaine des solutions réalisable : c'est l'ensemble des solutions réalisables du programme linéaire c'est-à-dire l'ensemble des solutions qui satisfait simultanément les contraintes fonctionnelles et les contraintes de non négativité.

Solution optimale : toute solution réalisable qui optimise la fonction objectif.

2.5 Propriétés géométriques

On note :

$$D_R = \{x \in \mathbb{R}^n \mid Ax = b, x \geq 0\} \quad (2.6)$$

L'ensemble des solutions réalisables du problème (2.5) sous forme standard.

- Un polyèdre Q de $\mathbb{R}^n$ est défini par $Q = \{x \in \mathbb{R}^n \mid Ax \leq b\}$ ou A est une matrice $m \times n$.
- Un ensemble $E \subseteq \mathbb{R}^n$ est dit convexe si $\forall x, y \in E, \lambda x + (1 - \lambda)y \in E$ pour tout $0 \leq \lambda \leq 1$.
- Soit E un ensemble convexe non vide de $\mathbb{R}^n$, x est dit point extrême ou sommet de E si $x = \lambda x_1 + (1 - \lambda)x_2, \forall x_1, x_2 \in E$ et $\lambda \in]0; 1[$ alors $x = x_1 = x_2$.
- x est une solution de base réalisable si et seulement si x est un sommet de $D_{\mathbb{R}}$.
- L'optimum de la fonction objectif Z sur $D_{\mathbb{R}}$, s'il existe au moins un sommet de $D_{\mathbb{R}}$ [12].

2.6 La méthode du simplexe

Cette méthode est l'un des outils de résolution des problèmes de programmation linéaire. Elle consiste à suivre un certain nombre d'étapes avant d'obtenir la solution d'un problème donné. Il s'agit d'une méthode algébrique itérative qui permet de trouver la solution exacte d'un problème de programmation linéaire en un nombre fini d'étapes[19].

2.6.1 Base et solution de base

• La méthode du simplexe utilise la forme standard où le système d'égalité s'écrit sous la forme matricielle :

$$Ax = b \text{ où } A \text{ est une matrice } n \times m, \text{ de rang : } m \leq n$$

- On appelle base de A ou matrice de base toute sous matrice carrée inversible B , $m \times m$ de A ; pour une base B , on peut (en réarrangeant les colonnes si nécessaire) partitionner A en (B, H) et x en (x_B, x_H) .

- H est la matrice $m \times (n - m)$ des colonnes hors base ou matrice hors base. Le vecteur x_B à pour composantes les m variables de base (associées aux colonnes de B).

- x_H a pour composantes les $n - m$ variables hors base.

- J_B est l'ensemble des indices de base et J_H l'ensemble des indices hors base.

- Pour obtenir une solution de base on pose $x_H = 0$ alors, $Bx_B = b \Rightarrow x_B = B^{-1}b$. Cette solution est la solution de base associée à la base B , elle peut violer les contraintes de positivité.

- On appelle solution de base réalisable une solution de base dont toutes les variables sont positives ou nulles[2].

2.6.2 Algorithme du simplexe

Étape 0

Transformer le programme linéaire sous la forme standard par rapport à un ensemble d'indices de base J_B de façon à avoir une solution de base admissible $\{x_B\}$.

Étape 1

Calculer le m-vecteur des potentiels :

$$y' = c'_B A_B^{-1}$$

Calculer le vecteur des estimations :

$$\Delta_j = \begin{cases} y' a_j - c_j & , j \in J_H \\ 0 & , j \in J_B \end{cases}$$

où a_j est la $j^{\text{ème}}$ colonne de A

Critère d'optimalité

Si $\Delta_j \geq 0, j \in J_H$, stop la solution actuelle est optimale

sinon soit $j_0 \in J_H / \Delta_{j_0} = \min_{\Delta_j < 0, j \in J_H} \Delta_j$.

Étape 2

Si $A_B^{-1} a_{j_0} \leq 0$, Stop : la solution est infinie

Sinon soit : j_1 tel que $\theta^0 = \theta_{j_1} = \min \left\{ \frac{x_j}{x_{j_0 j}} / x_{j_0 j} > 0, j \in J_B \right\}$.

Étape 3

Calculer $\bar{x} = \bar{x}_J = (\bar{x}_{J_B}, \bar{x}_{J_H})$

$$\bar{x}_{J_B} = x_{j_B} - \theta A^{-1} a_{j_0}$$

$$\bar{x}_j = 0, \text{ Si } j \in J_H \setminus j_0, \bar{x}_{j_0} = \theta^0.$$

Poser $\bar{J}_B = (J_B \setminus j_1) \cup j_0, \bar{J}_H = (J_H \setminus j_0) \cup j_1, \bar{A}_B = (I, \bar{J}_B),$

d'où $\{\bar{x}, \bar{A}_B\}$ la nouvelle solution réalisable de base

Ceci consiste une itération [2] .

2.6.3 Tableau du simplexe

La forme du tableau de l'algorithme du simplexe facilite les calculs précédents et se prête bien à la programmation. Nous allons, à travers de l'exemple suivant résumer les différentes itérations du tableau du simplexe[2].

Exemple 2.2

Soit à résoudre le programme linéaire suivant :

$$\begin{cases} \max Z = 2x_1 + x_2 \\ s.c \\ x_1 + x_2 \leq 10 \\ -x_1 + 2x_2 \leq 10 \\ x_1 \geq 0, x_2 \geq 0 \end{cases} \quad (2.7)$$

Transformons le programme sous forme standard en ajoutant les variables d'écart x_3, x_4

$$\begin{cases} \max Z = 2x_1 + x_2 \\ s.c \\ x_1 + x_2 + x_3 = 10 \\ -x_1 + 2x_2 + x_4 = 10 \\ x_i \geq 0, i = \overline{1, 4} \end{cases} \quad (2.8)$$

La solution de base de départ est $x = (0, 0, 10, 10)$ avec $J_B = \{3, 4\}$ et $J_H = \{1, 2\}$

Le vecteur des potentiels est : $y' = c'_B A_B^{-1} = (0, 0) \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = (0, 0)$

Déterminons le vecteur des estimations Δ :

$$\Delta_j = y' a_j - c_j$$

$$\Delta_1 = y' a_1 - c_1 = (0, 0) \begin{pmatrix} 1 \\ -1 \end{pmatrix} - 2 = -2.$$

$$\Delta_2 = y' a_2 - c_2 = (0, 0) \begin{pmatrix} 1 \\ 2 \end{pmatrix} - 1 = -1.$$

$$\Delta_3 = \Delta_4 = 0 \text{ car on sait bien que : } \Delta_j = 0, j \in J_B$$

Le critère d'optimalité n'est pas vérifié, donc la solution de départ n'est pas optimale. Dressons le premier tableau du simplexe :

c			2	1	0	0	
c_B	Base	b	x_1	x_2	x_3	x_4	θ
0	x_3	10	1	1	1	0	$\frac{10}{1}$
0	x_4	10	-1	2	0	1	/
		Δ_j	-2	-1	0	0	

Donc on doit faire entrer x_1 dans la base et sortir x_3 , la nouvelle base sera

$\bar{x} = (x_1, x_4)$ pour avoir la nouvelle solution $\bar{x}$ dressons le deuxième tableau du simplexe :

c			2	1	0	0	
c_B	Base	b	x_1	x_2	x_3	x_4	θ
2	x_1	10	1	1	1	0	
0	x_4	20	0	3	1	1	/
		Δ_j	0	1	2	0	

Le critère d'optimalité est vérifié, donc la nouvelle solution $\bar{x} = (10, 0, 0, 20)$ est optimale pour le programme standard, d'où la solution optimale du programme de départ est $x^0 = (10, 0)$ avec $z^0 = 20$.

2.7 Initialisation de l'algorithme du simplexe

Il est impératif d'avoir une solution de base réalisable de départ pour pouvoir utiliser la méthode du simplexe. Dans plusieurs situations, celle-ci n'est pas toujours facile à obtenir. La méthode des deux phases et la M-méthode que nous allons exposer dans la suite de ce chapitre sont des variantes de la méthode du simplexe.

2.7.1 Méthode des deux phases

Considérons le problème de programmation linéaire suivant (sans solution de base réalisable de départ)[2] :

$$\begin{cases} \max Z = c'x \\ s.c \\ Ax = b \\ x \geq 0 \\ b \geq 0 \end{cases} \quad (2.9)$$

Première phase :

La première phase de résolution du problème (2.9) consiste à déterminer une solution de base réalisable de (2.9). Pour cela on construit le problème suivant :

$$\begin{cases} \max Z = -\sum_{i=1}^m x_{n+i} \\ s.c \\ [Ax]_i + x_{n+i} = b_i, \quad i = 1, \dots, m, \\ x_j \geq 0, \quad x_{n+i} \geq 0, \quad i = 1, \dots, m, \quad j = 1, \dots, n \end{cases} \quad (2.10)$$

Où les x_{n+i} sont appelées des variables artificielles. Le problème (2.10) possède $n + m$ variables $X = (x_1, \dots, x_n, x_{n+1}, \dots, x_{n+m})$ et m équations. Le vecteur $X = (0, \dots, 0, b_1, \dots, b_m)$ est réalisable pour (2.10).

D'un autre côté, la fonction objectif est bornée supérieurement :

- $\sum_{i=1}^m x_{n+i} \leq 0$, donc le problème (2.10) admet une solution optimale $X^0 = (x^0, x_a^0)$.

Soit $\{x^0, x_a^0\}$ une solution optimale de (2.10).

Si $x_a^0 \neq 0$ alors, les contraintes du problème de départ (2.9) sont contradictoires.

Deuxième phase :

Soit (x^0, x_a^0) une solution optimale de (2.10) avec des variables artificielles nulles, alors on utilise la solution x^0 avec sa matrice A_B^0 comme solution de base de départ du problème (2.9) et ceci constitue la deuxième phase.

Si $x_{n+i}^0 = 0, \forall i$ et $\exists$ un indice $i_0/a_{i_0} \in A_B^0$ alors, pour revenir à la deuxième phase, il faut exclure cette colonne de A_B^0 .

Exemple 2.3

$$\begin{cases} \max Z = x_1 - 2x_2 + x_3 \\ s.c \\ x_1 + x_2 + x_3 = 5 \\ -2x_1 + 2x_2 = 4 \\ x_1 \geq 0, x_2 \geq 0, x_3 \geq 0 \end{cases} \quad (2.11)$$

Forme standard

$$\begin{cases} \max Z = -x_4 - x_5 \\ s.c \\ x_1 + x_2 + x_3 = 5 \\ 2x_1 + x_2 + x_4 = 3 \\ -2x_1 + 2x_2 + x_5 = 4 \\ x_j \geq 0, j = \overline{1, 5} \end{cases} \quad (2.12)$$

La solution de base de départ est $x = (0, 0, 5, 3, 4)$

Première phase

c			0	0	0	-1	-1	
c_B	Base	b	x_1	x_2	x_3	x_4	x_5	θ_j
0	x_3	5	1	1	1	0	0	5
-1	x_4	3	2	1	0	1	0	3
-1	x_5	4	-2	2	0	0	1	2
		Δ_j	0	-3	0	0	0	

Le critère d'optimalité n'est pas vérifié. Donc on doit faire entrer x_2 dans la base et sortir x_5 , la nouvelle base sera $\bar{x} = (x_3, x_4, x_2)$.

c			0	0	0	-1	-1	
c_B	base	b	x_1	x_2	x_3	x_4	x_5	θ
0	x_3	3	2	0	1	0	$-\frac{1}{2}$	$\frac{3}{2}$
-1	x_4	1	3	0	0	1	$-\frac{1}{2}$	$\frac{1}{3}$
0	x_2	2	-1	1	0	0	$-\frac{1}{2}$	/
		Δ_j	-3	0	0	0	$\frac{3}{2}$	

Le critère d'optimalité n'est pas vérifié. Donc on doit faire entrer x_1 dans la base et sortir x_4 , la nouvelle base sera $\bar{x}_2 = (x_3, x_1, x_2)$ pour avoir la nouvelle solution x dressons le 3ème tableau du simplexe :

c			0	0	0	-1	-1
c_B	base	b	x_1	x_2	x_3	x_4	x_5
0	x_3	$\frac{7}{3}$	0	0	1	$-\frac{2}{3}$	$-\frac{1}{6}$
0	x_1	$\frac{1}{3}$	1	0	0	$\frac{1}{3}$	$-\frac{1}{6}$
0	x_2	$\frac{7}{3}$	0	1	0	$\frac{1}{3}$	$\frac{1}{3}$
		Δ_j	0	0	0	1	1

Le critère d'optimalité est vérifié, $\Delta_j \geq 0$ la solution est optimale on passe a la deuxième phase.

c			0	0	0
c_B	Base	b	x_1	x_2	x_3
1	x_3	$\frac{7}{3}$	0	0	1
1	x_1	$\frac{1}{3}$	1	0	0
-2	x_2	$\frac{7}{3}$	0	1	0
		Δ_j	0	0	0

$\Delta_j \geq 0$ alors le critère d'optimalité est vérifié, donc la nouvelle solution est

$\bar{x} = (x_1, x_2, x_3) = \left(\frac{1}{3}, \frac{7}{3}, \frac{7}{3}\right)$, d'où la solution optimale du programme de départ est

$x^0 = \left(\frac{1}{3}, \frac{7}{3}, \frac{7}{3}\right)$ avec $Z^0 = -2$.

2.7.2 M-méthode

Elle a rassemblé les deux phases en une phase appelée la M-méthode.[2]

Du problème (2.9), on construit un problème (2.13) de la manière suivante :

$$\begin{cases} \max Z = c'x - M \sum_{i=1}^m x_{n+i}, \\ s.c \\ [Ax]_i + x_{n+i} = b_i, i = \overline{1, m}, \\ x_j \geq 0 \quad j = \overline{1, n+m}; \end{cases} \quad (2.13)$$

Où $M \gg 0$ (un nombre positif très grand) et $x_{n+i}, i = \overline{n+m}$ des variables artificielles.

Le vecteur $X = (0, b) = (x = 0, x_{n+i} = b_i, i = \overline{1, m})$ est une solution de base réalisable de (2.13) avec $A_B = (a_{n+1}, \dots, a_{n+m})$.

Exemple 2.4

Considérons le problème de programmation linéaire suivant :

$$\begin{cases} \max Z = -x_1 - 2x_2 \\ s.c \\ 2x_1 + x_2 \geq 8 \\ x_1 + x_2 \leq 6 \\ x_1 + 5x_2 \geq 10 \\ x_j \geq 0, \quad j = \overline{1, 2} \end{cases} \quad (2.14)$$

On ajoute des variables d'écart x_3, x_4, x_5 :

$$\begin{cases} \max Z = -x_1 - 2x_2 \\ s.c \\ 2x_1 + x_2 - x_3 = 8 \\ x_1 + x_2 + x_4 = 6 \\ x_1 + 5x_2 - x_5 = 10 \\ x_j \geq 0, \quad j = \overline{1, 5} \end{cases} \quad (2.15)$$

Le M-problème correspondant est :

$$\begin{cases} \max Z = -x_1 - 2x_2 - M(x_6 + x_7) \\ s.c \\ 2x_1 + x_2 - x_3 + x_6 = 8 \\ x_1 + x_2 + x_4 = 6 \\ x_1 + 5x_2 - x_5 + x_7 = 10 \\ x_j \geq 0, \quad j = \overline{1, 7} \end{cases} \quad (2.16)$$

Pour ce problème, $x = (0, 0, 0, 6, 0, 8, 10)$ est une solution réalisable de base.

Dressons le premier tableau du simplexe :

c		-1	-2	0	0	0	-M	-M	
base	b	x_1	x_2	x_3	x_4	x_5	x_6	x_7	θ
x_6	8	2	1	-1	0	0	1	0	$\frac{8}{1}$
x_4	6	1	1	0	1	0	0	0	$\frac{8}{1}$
x_7	10	1	5	0	0	-1	0	1	$\frac{10}{5}$
		-3M+1	-6M+2	M	0	M	0	0	

Le critère d'optimalité n'est pas vérifié. Donc on doit faire entrer x_2 dans la base et sortir x_7 , la nouvelle base sera $\bar{x}_2 = (x_6, x_4, x_2)$ pour avoir la nouvelle solution x dressons le 2^{ème} tableau du simplexe :

	c	-1	-2	0	0	0	-M	-M	
base	b	x_1	x_2	x_3	x_4	x_5	x_6	x_7	θ
x_6	6	$\frac{9}{5}$	0	-1	0	$\frac{1}{5}$	1	$-\frac{1}{5}$	$\frac{10}{3}$
x_4	4	$\frac{4}{5}$	0	0	1	$\frac{1}{5}$	0	$-\frac{1}{5}$	5
x_2	2	$\frac{1}{5}$	1	0	0	$-\frac{1}{5}$	0	$\frac{1}{5}$	10
		$-\frac{9M}{5} + \frac{3}{5}$	0	M	0	$-\frac{M}{5} + \frac{2}{5}$	0	$\frac{6}{5}M - \frac{2}{5}$	

Le critère d'optimalité n'est pas vérifié. Donc on doit faire entrer x_1 dans la base et sortir x_6 , la nouvelle base sera $\bar{x}_2 = (x_1, x_4, x_2)$ pour avoir la nouvelle solution x dressons le 3^{ème} tableau du simplexe :

		-1	-2	0	0	0	-M	-M
base	b	x_1	x_2	x_3	x_4	x_5	x_6	x_7
x_1	$\frac{10}{3}$	1	0	$-\frac{5}{9}$	0	$\frac{1}{9}$	$\frac{5}{9}$	$-\frac{1}{9}$
x_4	$\frac{4}{3}$	0	0	$\frac{4}{9}$	1	$\frac{1}{9}$	$-\frac{4}{9}$	$-\frac{1}{9}$
x_2	$\frac{4}{3}$	0	1	$\frac{1}{9}$	0	$-\frac{2}{9}$	$-\frac{1}{9}$	$\frac{2}{9}$
		0	0	$\frac{1}{3}$	0	$\frac{1}{3}$	$M - \frac{1}{3}$	$M - \frac{1}{3}$

$\Delta_j \geq 0$, le critère d'optimalité est vérifié, d'où la solution du départ réalisable est

$$x^o = \left(\frac{10}{3}, \frac{4}{3} \right) \quad Z^0 = -6$$

2.8 Conclusion

Dans ce chapitre nous avons présenté une méthode de résolution des problèmes de programmation linéaire qui est la méthode du simplexe. Dans un premier temps nous avons parlé brièvement sur la programmation linéaire. Puis, nous avons montré l'application de cette méthode pour la résolution des problèmes de la programmation linéaire.

Chapitre 3

Programmation linéaire floue

3.1 Introduction

Un programme linéaire flou est un programme dont certaines ou toutes les données sont incertaines. L'incertitude est de nature floue on distingue deux cas entre autres :

1. le cas où les inégalités (ou les égalités) sont relaxées on parlera de programmation flexible.
2. le cas où les données imprécises sont représentées par des sous-ensembles flous, on parlera de programmation robuste[9].

3.2 Programmation flexible

3.2.1 Décision dans un environnement flou

On considère un problème dont l'objectif et les m contraintes sont vaguement définis, représentés par des ensembles flous d'un même référentiel X dont les fonctions d'appartenance respectives sont μ_0 et μ_i , $i = \overline{1, m}$. La décision qui doit satisfaire l'objectif et les m contraintes est donc représentée par un ensemble flou, intersection de ces derniers et dont la fonction d'appartenance est μ_D telle que[1] :

$$\mu_D(x) = \min(\mu_i(x) / i = 0, 1, 2, \dots, m) \quad .$$

La meilleure décision est déterminée par la résolution du problème suivant :

$$\max (\mu_D(x) / x \in X) = \max \{ \min (\mu_i(x) / i = 0, 1, 2, \dots, m) / x \in X \}.$$

3.2.2 Résolution d'un programme linéaire flexible

Considérons le programme linéaire flexible suivant[1, 5, 9] :

$$\begin{cases} \widetilde{\max} Z = c'x \\ s.c \\ A_i x \widetilde{\theta} b_i, i = 1, 2, \dots, m \\ x \geq 0 \end{cases} \quad (3.1)$$

où

$\tilde{\theta} \in \{\tilde{\leq}, \tilde{=}, \tilde{\geq}\}$ et $\tilde{\leq}, \tilde{=}, \tilde{\geq}$ sont les versions flexibles de $\leq, =, \geq$ respectivement .

La notation $\tilde{\cdot}$ désigne le fait que l'objectif et les contraintes ne sont pas des impératifs stricts.

Selon Zimermann, le programme (3.1) peut s'interpréter comme suit :

$x \in (\mathbb{R}^+)^n$ telle que :

$$\begin{cases} c'x \tilde{\leq} Z_0 \\ s.c \\ A_i x \tilde{\theta} b_i, i = \overline{1.m} \\ x \geq 0 \end{cases} \quad (3.2)$$

Z_0 : est une valeur fixée.

l'objectif flou et les contraintes floues peuvent être représentés par les ensembles flous respectifs U_0 et U_i , $i = \overline{1.m}$ dont les fonctions d'appartenance sont respectivement μ_0 et μ_i , $i = \overline{1.m}$, où μ_i , $i = \overline{1.m}$ est définie selon que $\tilde{\theta}$ est $\tilde{\leq}, \tilde{=}$ ou $\tilde{\geq}$ comme suit :

• $\tilde{\theta}$ est $\tilde{\leq}$

$$\mu_i(x) = \mu_i(A_i x) = \begin{cases} 1 & si \quad A_i x \leq b_i \\ 1 - \frac{A_i x - b_i}{d_i} & si \quad b_i \leq A_i x \leq b_i + d_i \\ 0 & si \quad A_i x > b_i + d_i \end{cases} \quad (3.3)$$

Représentation graphique :

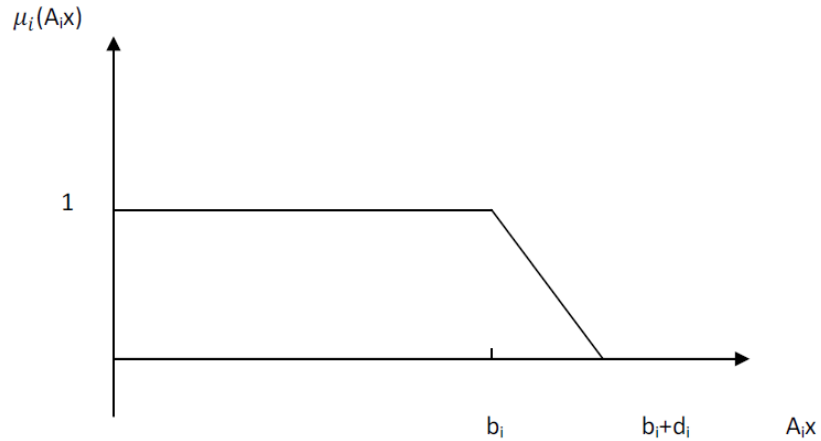


Figure 2.1 - Contraintes du type $A_i(x) \tilde{\leq} b_i$, $i = \overline{1.m}$.

• $\tilde{\theta}$ est $\tilde{=}$

$$\mu_i(x) = \mu_i(A_i x) = \begin{cases} 1 & \text{si } A_i x = b_i \\ 1 - \frac{A_i x - b_i}{d_i} & \text{si } b_i < A_i x \leq b_i + d_i \\ 1 - \frac{b_i - A_i x}{d_i} & \text{si } b_i - d_i \leq A_i x < b_i \\ 0 & \text{si } A_i x > b_i + d_i, A_i x < b_i - d_i \end{cases} \quad (3.4)$$

Représentation graphique :

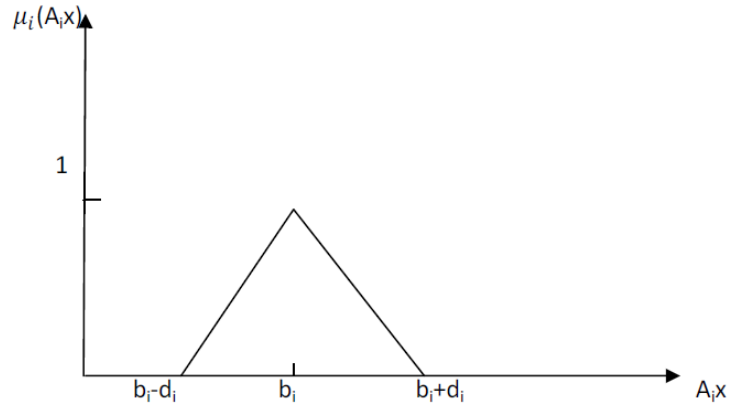


Figure 2.2 - Contraintes du types $A_i(x) \cong b_i, i = \overline{1.m}$.

• $\tilde{\theta}$ est $\gtrsim$

$$\mu_i(x) = \mu_i(A_i x) = \begin{cases} 1 & \text{si } A_i x \geq b_i \\ 1 - \frac{b_i - A_i x}{d_i} & \text{si } b_i - d_i \leq A_i x \leq b_i \\ 0 & \text{si } A_i x < b_i - d_i \end{cases} \quad (3.5)$$

Représentation graphique :

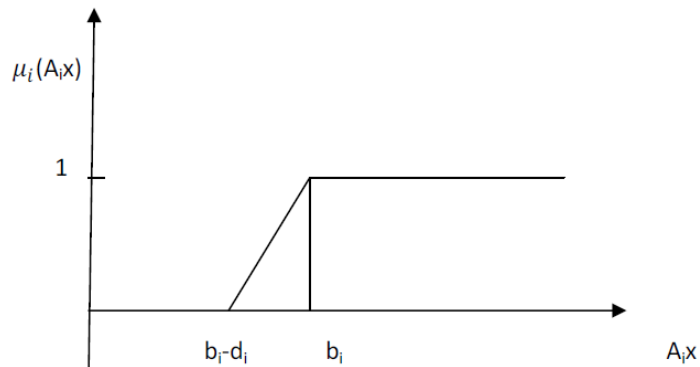


Figure 2.3 - contraintes du type $A_i(x) \gtrsim b_i, i = \overline{1.m}$.

Remarque 3.1

La méthode flexible joue un rôle identique à la fonction objectif et aux contraintes, on détermine la fonction d'appartenance de la fonction objectif de la même manière que les contraintes. Pour se faire, il faut fixer un objectif à atteindre et le degré de satisfaction.

Méthode d'agrégation des degrés de satisfaction

Il s'est basé sur le principe d'agrégation de Bellman et Zadeh pour déduire une fonction objectif finale à maximiser. cette fonction exprime le degré total de satisfaction.

$$\mu_D(x) = \min(\mu_0(x), \mu_1(x), \dots, \mu_m(x))$$

Chercher la solution qui réalise le meilleur degré de satisfaction. la solution optimale est alors déterminée par résolution du problème déterministe

$$\max_{x \geq 0} \mu_D(x)$$

qui est équivalent au problème suivant [18] :

$$\begin{cases} \max Z = \lambda \\ s.c \\ \lambda \leq \mu_i(x), & i = 0, 1, 2, \dots, m \\ 0 \leq \lambda \leq 1 \\ x \geq 0 \end{cases} \quad (3.6)$$

Exemple 3.1

On considère le programme flexible suivant :

$$\begin{cases} \widetilde{\max} Z = 5x_1 - x_2 \\ s.c \\ -x_1 + 2x_2 \widetilde{\leq} 5 \\ 3x_1 + x_2 \widetilde{\leq} 11 \\ x_1 \geq 0, x_2 \geq 0 \end{cases} \quad (3.7)$$

1) On détermine les fonctions d'appartenance de l'objectif et des contraintes sachant que l'objectif excède la valeur $Z_0 = 6$ et que les écarts de tolérance de l'objectif et des contraintes sont respectivement $d_0 = 2$, $d_1 = 5$, $d_2 = 1$:

Selon Zimermann, le programme (3.7) peut s'interpréter comme suit :

$$\begin{cases} 5x_1 - x_2 \widetilde{\leq} 6 \\ s.c \\ -x_1 + 2x_2 \widetilde{\leq} 5 \\ 3x_1 + x_2 \widetilde{\leq} 11 \\ x_1 \geq 0, x_2 \geq 0 \end{cases} \quad (3.8)$$

Transformation de l'objectif

$$\mu_0(x) = \begin{cases} 1 & \text{si } 5x_1 - x_2 \leq 6 \\ 1 - \frac{5x_1 - x_2 - 6}{2} & \text{si } 6 \leq 5x_1 - x_2 \leq 8 \\ 0 & \text{si } 5x_1 - x_2 > 8 \end{cases} \quad (3.9)$$

$$= \begin{cases} 1 & \text{si } 5x_1 - x_2 \leq 6 \\ -\frac{5x_1 + x_2 + 8}{2} & \text{si } 6 \leq 5x_1 - x_2 \leq 8 \\ 0 & \text{si } 5x_1 - x_2 > 8 \end{cases} \quad (3.10)$$

Transformation de la 1^{ère} contraintes :

$$\mu_1(x) = \begin{cases} 1 & \text{si } -x_1 + 2x_2 \leq 5 \\ 1 - \frac{(-x_1 + 2x_2) - 5}{5} & \text{si } 5 \leq -x_1 + 2x_2 \leq 10 \\ 0 & \text{si } -x_1 + 2x_2 > 10 \end{cases} \quad (3.11)$$

$$= \begin{cases} 1 & \text{si } -x_1 + 2x_2 \leq 5 \\ \frac{x_1 - 2x_2 + 10}{5} & \text{si } 5 \leq -x_1 + 2x_2 \leq 10 \\ 0 & \text{si } -x_1 + 2x_2 > 10 \end{cases} \quad (3.12)$$

Transformation de la 2^{ème} contraintes :

$$\mu_2(x) = \begin{cases} 1 & \text{si } 3x_1 + x_2 \leq 11 \\ 1 - \frac{3x_1 + x_2 - 11}{1} & \text{si } 11 \leq 3x_1 + x_2 \leq 12 \\ 0 & \text{si } 3x_1 + x_2 > 12 \end{cases} \quad (3.13)$$

$$= \begin{cases} 1 & \text{si } 3x_1 + x_2 \leq 11 \\ -3x_1 - x_2 + 12 & \text{si } 11 \leq 3x_1 + x_2 \leq 12 \\ 0 & \text{si } 3x_1 + x_2 > 12 \end{cases} \quad (3.14)$$

2) Le problème déterministe associé à (3.7) est :

$$\begin{cases} \max Z = \lambda \\ \text{s.c} \\ \lambda \leq \frac{-5x_1 + x_2 + 8}{2} \\ \lambda \leq \frac{x_1 - 2x_2 + 10}{5} \\ \lambda \leq -3x_1 - x_2 + 12 \\ x_1 \geq 0, x_2 \geq 0, 0 \leq \lambda \leq 1 \end{cases} \quad (3.15)$$

$$\left\{ \begin{array}{l} \max Z = \lambda \\ s.c \\ 2\lambda \leq -5x_1 + x_2 + 8 \\ 5\lambda \leq x_1 - 2x_2 + 10 \\ \lambda \leq -3x_1 - x_2 + 12 \\ x_1 \geq 0, x_2 \geq 0, 0 \leq \lambda \leq 1 \end{array} \right. \quad (3.16)$$

$$\left\{ \begin{array}{l} \max Z = \lambda \\ s.c \\ 2\lambda + 5x_1 - x_2 \leq 8 \\ 5\lambda - x_1 + 2x_2 \leq 10 \\ \lambda + 3x_1 + x_2 \leq 12 \\ x_1 \geq 0, x_2 \geq 0, 0 \leq \lambda \leq 1 \end{array} \right. \quad (3.17)$$

la forme canonique

$$\left\{ \begin{array}{l} \max Z = \lambda \\ s.c \\ 2\lambda + 5x_1 - x_2 + x_3 = 8 \\ 5\lambda - x_1 + 2x_2 + x_4 = 10 \\ \lambda + 3x_1 + x_2 + x_5 = 12 \\ \lambda + x_6 = 1 \\ x_j \geq 0, j = \overline{1,6} \end{array} \right. \quad (3.18)$$

Résolution du problème (3.7) par la méthode du simplexe :

c			1	0	0	0	0	0	1	θ_j
	Base	b	λ	x_1	x_2	x_3	x_4	x_5	x_6	
0	x_3	8	2	5	-1	1	0	0	0	4
0	x_4	10	5	-1	2	0	1	0	0	2
0	x_5	12	1	3	1	0	0	1	0	12
0	x_6	1	1	0	0	0	0	0	1	1
		Δ_j	-1	0	0	0	0	0	0	
0	x_3	6	0	5	-1	1	0	0	-2	
0	x_4	5	0	-1	2	0	1	0	-5	
0	x_5	11	0	3	1	0	0	1	-1	
1	λ	1	1	0	0	0	0	0	1	
		Δ_j	0	0	0	0	0	0	1	

$(x_1, x_2, x_3, x_4, x_5, x_6, \lambda) = (0, 0, 6, 5, 11, 0, 1)$ et une solution optimale de (P) avec $Z^* = 1$

3.3 Programmation robuste

Lorsque les coefficients des contraintes sont des données imprécises, l'inégalité est remplacées par $\subseteq$, on parle alors de la programmation robuste (généralisation d'un problème linéaire inexacte).

3.3.1 Programmation linéaire inexacte (Solster)

Un programme linéaire inexact est un programme de la forme[10] :

$$\begin{cases} \max Z = c'x \\ s.c \\ x_1 K_1 + x_2 K_2 + \dots + x_n K_n \subseteq K \\ x_j \geq 0, j = \overline{1, n} \end{cases} \quad (3.19)$$

où :

$(K_j); j = 1, \dots, n$ est K sont des ensembles convexes de $\mathbb{R}^n$, $\subseteq$ est l'inclusion entre ensembles et $+$ représente l'addition ensembliste, elle est définie comme suit : soient A et B deux ensembles vulgaires, alors $A + B = \{a + b, a \in A \text{ et } b \in B\}$.

3.3.2 Résolution d'un Programme robuste

Un programme robuste est un programme de la forme :[10]

$$\begin{cases} \max Z = c'x \\ s.c \\ x \odot \tilde{A}_1 \oplus x_2 \odot \tilde{A}_2 \oplus \dots \oplus x_n \odot \tilde{A}_n [\subseteq] \tilde{b} \\ x_j \geq 0, j = \overline{1.n} \end{cases} \quad (3.20)$$

où :

($\tilde{A}_j$), $j = \overline{1.n}$ sont des sous ensembles flous de $\mathbb{R}$ et " $\oplus$ " addition des ensembles flous et " $[\subseteq]$ " inclusion entre ensembles flous

On représente l'ensemble des contraintes de (3.19) par :

$$E = \left\{ x \in \mathbb{R}^n / x_1 \odot \tilde{A}_1 \oplus x_2 \odot \tilde{A}_2 \oplus \dots \oplus x_n \odot \tilde{A}_n [\subseteq] \tilde{b} / x \geq 0 \right\}$$

Théorème[11]

$x^0 \in E$ est optimale pour (3.19) si et seulement si $x^0 \in E$ est optimal pour le programme suivant :

$$\begin{cases} \max Z = c'x \\ s.c \\ x_1 \tilde{A}_1^\alpha + x_2 \tilde{A}_2^\alpha + \dots + x_n \tilde{A}_n^\alpha \subseteq \tilde{b}^\alpha \quad \forall \alpha \in]0 ; 1[\\ x_j \geq 0, j = \overline{1.n} \end{cases} \quad (3.21)$$

(3.20) est un programme semi-infini, c'est à dire un programme avec une infinité de contraintes.

En supposant que les images des fonctions d'appartenance des sous ensembles flous sont discrètes et finies, on obtient un programme linéaire avec un nombre fini de contraintes comme suit :

Proposition [10]

Si $Im \mu \tilde{A}_j = \{\alpha_1, \alpha_2, \dots, \alpha_p\}$ avec $0 = \alpha < \alpha_2 < \dots < \alpha_p < 1$

Alors :

$$x \in E = \left\{ x \in \mathbb{R}^n / x_1 \odot \tilde{A}_1 \oplus x_2 \odot \tilde{A}_2 \oplus \dots \oplus x_n \odot \tilde{A}_n [\subseteq] \tilde{b} / x \geq 0 \right\}$$

si et seulement si

$$x \in E' = \begin{cases} x_1 \tilde{A}_1^{\alpha_k} + x_2 \tilde{A}_2^{\alpha_k} + \dots + x_n \tilde{A}_n^{\alpha_k} \subseteq \tilde{b}^{\alpha_k} \quad k = \overline{1.p} \\ x_j \geq 0, \quad j = 1, 2, \dots, n \end{cases}$$

3.3.3 Cas des nombres flous

Si les composantes $\tilde{a}_{ij}$ de $\tilde{A}$ et $\tilde{b}_i$ de $\tilde{b}$ sont des nombres flous dont les ensembles de niveau respectifs sont des intervalles compacts.

Alors, le programme (3.20) s'écrit sous la forme suivante [1] :

$$\left\{ \begin{array}{l} \max Z = c'x \\ s.c \\ \begin{array}{llll} [\tilde{a}_{11}^{\alpha_1} \tilde{a}_{11}^{\alpha_1}] x_1 + & [\tilde{a}_{12}^{\alpha_1} \tilde{a}_{12}^{\alpha_1}] x_2 & \dots + & [\tilde{a}_{1n}^{\alpha_1} \tilde{a}_{1n}^{\alpha_1}] x_n \subseteq [\tilde{b}_1^{\alpha_1} \tilde{b}_1^{\alpha_1}] \\ [\tilde{a}_{21}^{\alpha_2} \tilde{a}_{21}^{\alpha_2}] x_1 + & [\tilde{a}_{22}^{\alpha_2} \tilde{a}_{22}^{\alpha_2}] x_2 & \dots + & [\tilde{a}_{2n}^{\alpha_2} \tilde{a}_{2n}^{\alpha_2}] x_n \subseteq [\tilde{b}_2^{\alpha_2} \tilde{b}_2^{\alpha_2}] \\ \vdots & \vdots & \vdots & \vdots \\ [\tilde{a}_{m1}^{\alpha_p} \tilde{a}_{m1}^{\alpha_p}] x_1 + & [\tilde{a}_{m2}^{\alpha_p} \tilde{a}_{m2}^{\alpha_p}] x_2 & \dots + & [\tilde{a}_{mn}^{\alpha_p} \tilde{a}_{mn}^{\alpha_p}] x_n \subseteq [\tilde{b}_m^{\alpha_p} \tilde{b}_m^{\alpha_p}] \end{array} \\ x_j \geq 0; \quad j = 1, \dots, n; \quad k = 1, \dots, p \end{array} \right\} \quad (3.22)$$

où :

$$\tilde{a}_{ij}^{\alpha_k} = \{ x \in R / \mu_{\tilde{a}_{ij}}(x) \geq \alpha_k \}$$

$$\underline{\tilde{a}}_{ij}^{\alpha_k} = \inf \{ x \in R / \mu_{\tilde{a}_{ij}}(x) \geq \alpha_k \}$$

$$\tilde{a}_{ij}^{\alpha_k} = \sup \{ x \in R / \mu_{\tilde{a}_{ij}}(x) \geq \alpha_k \}$$

$$\tilde{b}_i^{\alpha_k} = \{ x \in R / \mu_{\tilde{b}_i}(x) \geq \alpha_k \}$$

$$\underline{\tilde{b}}_i^{\alpha_k} = \inf \{ x \in R / \mu_{\tilde{b}_i}(x) \geq \alpha_k \}$$

$$\tilde{b}_i^{\alpha_k} = \sup \{ x \in R / \mu_{\tilde{b}_i}(x) \geq \alpha_k \}$$

Remarque 3.2[5]

puisque $\tilde{a}_{ij}$ et $\tilde{b}_i$ sont des nombres flous donc ce sont des ensembles flous convexes normalisés de $\mathbb{R}$.

les ensembles $\tilde{a}_{ij}^{\alpha_k}$ et $\tilde{b}_i^{\alpha_k}$ sont des convexe dans $\mathbb{R}$, donc des intervalle de $\mathbb{R}$.

et par hypothèse $\tilde{a}_{ij}^{\alpha_k}$ et $\tilde{b}_i^{\alpha_k}$ sont compact.

par conséquent :

$$\tilde{a}_{ij}^{\alpha_k} = [\underline{\tilde{a}}_{ij}^{\alpha_k}, \tilde{a}_{ij}^{\alpha_k}] \quad , \quad \tilde{b}_i^{\alpha_k} = [\underline{\tilde{b}}_i^{\alpha_k}, \tilde{b}_i^{\alpha_k}]$$

3.3.4 Cas des nombres flous de types L-R

si les composantes $\tilde{a}_{ij}$ de $\tilde{A}$ et $\tilde{b}_i$ de $\tilde{b}$ sont des nombres flous de type L-R[5].

$$\tilde{a}_{ij} = (m_{ij}, \alpha_{ij}, \beta_{ij}) \text{ et } \tilde{b}_i = (m_i, \alpha_i, \beta_i)$$

Où :

$$\begin{cases} \alpha_{ij} > 0 \\ \beta_{ij} > 0 \\ \alpha_i > 0 \\ \beta_i > 0 \end{cases}$$

$x = (x_1, x_2, \dots, x_n)$ est admissible pour le programme (3.19) si et seulement si x vérifie le système suivant :

$$\begin{cases} \sum_{j=1}^n m_{ij} x_j = m_i & i = 1, \dots, m \\ \sum_{j=1}^n \alpha_{ij} x_j \leq \alpha_i & i = 1, \dots, n \\ \sum_{j=1}^n \beta_{ij} x_j \leq \beta_i & i = 1, \dots, m \end{cases} \quad (3.23)$$

Exemple 3.2

On posant $\alpha = 0.5$ et on utilisant l'algorithme du simplexe, on trouve la solution optimale du problème (3.23).

$$\begin{cases} \max Z = x_1 + 2x_2 \\ s.c \\ \tilde{a}_1 x_1 + \tilde{a}_2 x_2 \subseteq \tilde{b} \\ x_2 \geq -1 \\ x_1 \geq 0, x_2 \geq 0 \end{cases} \quad (3.24)$$

Sachant que :

$$\tilde{a}_1 = \{(1/0.7), (2/0.5), (3/0.6), (5/0.3)\}$$

$$\tilde{a}_2 = \{(5/0.2), (3/0.9), (1/0.7), (2/0.6)\}$$

$$\tilde{b} = \{(6/0.7), (1/0.2), (-2/0.8), (5/0.4)\}$$

On aura :

$$\begin{cases} \max Z = x_1 + 2x_2 \\ s.c \\ [1, 2] x_1 + [2, 3] x_2 \subseteq [-2, 6] \\ x_2 \geq -1 \\ x_1 \geq 0, x_2 \geq 0 \end{cases} \quad (3.25)$$

$$\begin{cases} \max Z = x_1 + 2x_2 \\ \text{s.t.} \\ x_1 + 2x_2 \geq -2 \\ 2x_1 + 3x_2 \leq 6 \\ x_2 \geq -1 \\ x_1 \geq 0, x_2 \geq 0 \end{cases} \quad (3.26)$$

$$\begin{cases} \max Z = x_1 + 2x_2 \\ \text{s.t.} \\ x_1 + 2x_2 - x_3 = -2 \\ 2x_1 + 3x_2 + x_4 = 6 \\ x_2 - x_5 = -1 \\ x_1 \geq 0, x_2 \geq 0 \end{cases} \quad (3.27)$$

$$\begin{cases} \max Z = x_1 + 2x_2 \\ \text{s.t.} \\ -x_1 - 2x_2 + x_3 = 2 \\ 2x_1 + 3x_2 + x_4 = 6 \\ -x_2 + x_5 = 1 \\ x_1 \geq 0, x_2 \geq 0 \end{cases} \quad (3.28)$$

c			1	2	0	0	0	/
	<i>base</i>	<i>b</i>	x_1	x_2	x_3	x_4	x_5	θ_j
0	x_3	2	-1	-2	1	0	0	/
0	x_4	6	2	3	0	1	0	/
0	x_5	1	0	-1	0	0	1	/
		Δ_j	-1	-2	0	0	0	/
		1	2	0	0	0	0	
	<i>base</i>	<i>b</i>	x_1	x_2	x_3	x_4	x_5	/
0	x_3	6	$\frac{1}{3}$	0	1	$\frac{2}{3}$	0	/
2	x_2	2	$\frac{2}{3}$	1	0	$\frac{1}{3}$	0	/
0	x_5	3	$\frac{2}{3}$	0	0	$\frac{1}{3}$	1	/
		Δ_j	$\frac{1}{3}$	0	0	$\frac{2}{3}$	0	/

Le critère d'optimalité est vérifié, tous les $\Delta_j > 0$

$x^* = (0, 2)$ est la solution optimale du problème avec $Z^* = 4$.

3.4 Conclusion

Nous avons présenté, dans ce chapitre, la modélisation de l'imprécision dans les problèmes de décisions, en particulier, les problèmes linéaires flous avec la programmation flexible et la programmation robuste.

Chapitre 4

Programmation mathématique

4.1 Introduction

Ce chapitre est divisé en deux parties. Dans la première partie, nous allons présenter les notions mathématiques dont on aura besoin ; la deuxième partie sera consacrée à la programmation mathématique floue en étudiant le cas de programmation non linéaire floue flexible par exemple.

4.2 Notion topologique

Nous nous plaçons dans $\mathbb{R}^n$ qui, muni de la norme euclidienne[12]

$$\|x\| = (\sum_{i=1}^n x_i^2)^{1/2}$$

est un espace vectoriel normé.

Un élément x de $\mathbb{R}^n$ est un n vecteur tel que :

$$x = \begin{pmatrix} x_1 \\ \cdot \\ \cdot \\ \cdot \\ x_n \end{pmatrix}$$

On notera par x' le vecteur transposé de x , $x' = (x_1, \dots, x_n)$.

4.2.1 Ensembles convexes

Définition 4.1[12]

$y \in D$ est un point intérieur de D s'il existe $\varepsilon > 0$ tel que

$$\|x - y\| < \varepsilon \Rightarrow x \in D$$

(autrement dit, s'il existe une boule centrée en y et contenue dans D).

Définition 4.2[12]

Un sous ensemble est dit ouvert s'il coïncide avec son intérieur c'est-à-dire si :

$$D = \text{int}(D), (\text{int}(D) : \text{intérieur de } D)$$

Définition 4.3[12]

Un sous-ensemble $D \subseteq \mathbb{R}^n$ est dit fermé si son complémentaire est ouvert.

Définition 4.4[12]

On dit que $D \subseteq \mathbb{R}^n$ est borné si

$$\forall x \in D, \exists r > 0 \text{ tel que : } \|x\| < r .$$

Un ensemble compact de $\mathbb{R}^n$ est ensemble fermé borné sur $\mathbb{R}^n$.

Définition 4.5[12]

Un ensemble $D \subseteq \mathbb{R}^n$ est dit affine si et seulement si :

$$\forall x, y \in D, (1 - \lambda)x + \lambda y \in D, \forall \lambda \in \mathbb{R}.$$

Définition 4.6[12]

Étant donné $x \in \mathbb{R}^n$, $\nu(x)$ est un voisinage de x si c'est un sous-ensemble ouvert contenant x . De façon équivalente, $\nu(x)$ est un voisinage de x s'il contient une boule de centre x .

4.2.2 Fonctions convexes**Définition 4.7**[8]

On dit que une fonction $f : \mathbb{R}^n \rightarrow \mathbb{R}$, définie sur un ensemble convexe D , est convexe si elle vérifiée :

$$\forall (x, y) \in D^2, \forall \lambda \in [0, 1], f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda)f(y).$$

On dira que f est strictement convexe si

$$\forall (x, y) \in D^2, x \neq y, \forall \lambda \in]0, 1[f(\lambda x + (1 - \lambda)y) < \lambda f(x) + (1 - \lambda)f(y).$$

Définition 4.8[8]

La matrice Hessienne de $f(x_1, \dots, x_n)$ est de la forme :

$$H_f(x_1, \dots, x_n) = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{pmatrix}$$

Définition 4.9[8]

On définit la dérivée partielle première par rapport à x_i de la fonction $f(x_1, x_2, \dots, x_n)$:

$$\lim_{h \rightarrow 0} \frac{f(x_1, \dots, x_i + h, \dots, x_n) - f(x_1, \dots, x_i, \dots, x_n)}{h}$$

Elle est notée par $\frac{\partial f(x_1, \dots, x_n)}{\partial x_i}$.

On définit le gradient par :

$$\nabla f(x_1, \dots, x_n) = \begin{pmatrix} \frac{\partial f(x_1, \dots, x_n)}{\partial x_1} \\ \vdots \\ \frac{\partial f(x_1, \dots, x_n)}{\partial x_n} \end{pmatrix}$$

Théorème 4.1[8]

f est de classe C^2 , f est convexe sur $\mathbb{R}^n$ si et seulement si :

H_f est semi définie positive sur $\mathbb{R}^n$.

4.3 Définition d'un programme mathématique

De manière générale, la résolution de problèmes de programmation mathématique vise à déterminer l'allocation optimale de ressources limitées pour atteindre certains objectifs. Les allocations doivent minimiser ou maximiser une fonction dite objectif.

Un problème de programmation mathématique est un problème d'optimisation dans $\mathbb{R}^n$ qu'on peut présenter dans la forme suivante [7, 19] :

$$(P_1) \begin{cases} \min f(x) \text{ sur } X \\ s.c \\ g(x) \leq 0 \\ h(x) = 0 \end{cases} \quad (4.1)$$

$$D = \{x \in X / g(x) \leq 0 \text{ et } h(x) = 0\} \quad (4.2)$$

où $X \subset \mathbb{R}^n$, $x' = (x_1, x_2, \dots, x_n)' \in \mathbb{R}^n$ appelé variable de décision.

$f : \mathbb{R}^n \rightarrow \mathbb{R}$ est la fonction objectif

$g : \mathbb{R}^n \rightarrow \mathbb{R}^p$ et $h : \mathbb{R}^n \rightarrow \mathbb{R}^q$ sont des fonction vectorielles appelées fonctions contraintes, avec :

$$g(x) = (g_1(x), g_2(x), \dots, g_p(x)) \text{ et } h(x) = (h_1(x), h_2(x), \dots, h_q(x)).$$

signifiant que nous cherchons la solution x^0 qui minimise la fonction f sur l'ensemble X sous les contraintes $g(x) \leq 0$ et $h(x) = 0$. La valeur représentée par cette formulation est le nombre réel $f(x^0)$.

Remarque 4.1[7]

On dit généralement que le problème (4.1) est un problème de minimisation, la recherche du maximum d'une fonction f se ramène immédiatement au problème de minimisation de $F = -f$. Dans ce qui suit, nous n'envisageons que le cas de la minimisation.

4.3.1 Notion de solution

i. On appelle solution du problème (4.1) tout vecteur $x \in X$ vérifiant les contraintes, c'est-à-dire que : $g(x) \leq 0$ et $h(x) = 0$.

ii. On appelle solution optimale de (4.1) une solution x^0 qui minimise $f(x)$ sur l'ensemble de toutes les solutions, c'est-à-dire que : $f(x^0) \leq f(x)$ pour toute solution x admissible.

iii. On dit qu'un vecteur x^0 est un optimum local de (4.1) si il existe un voisinage $U_\delta(x^0)$, $\delta > 0$, de x^0 tel que x^0 soit optimum global du problème [7] :

$$\begin{cases} \min f(x) \\ s.c \\ g(x) \leq 0 \\ h(x) = 0 \\ x \in X \cap U_\delta(x^0) \end{cases} \quad (4.3)$$

Remarque 4.2

On notera qu'il est souvent possible de caractériser les optimums locaux d'un problème, c'est-à-dire de donner les conditions nécessaires et/ou suffisantes pour qu'une solution x soit un optimum local.

Illustrons les notions d'optimum local et d'optimum global par la figure (4.1) sur une fonction à une seule variable[7].

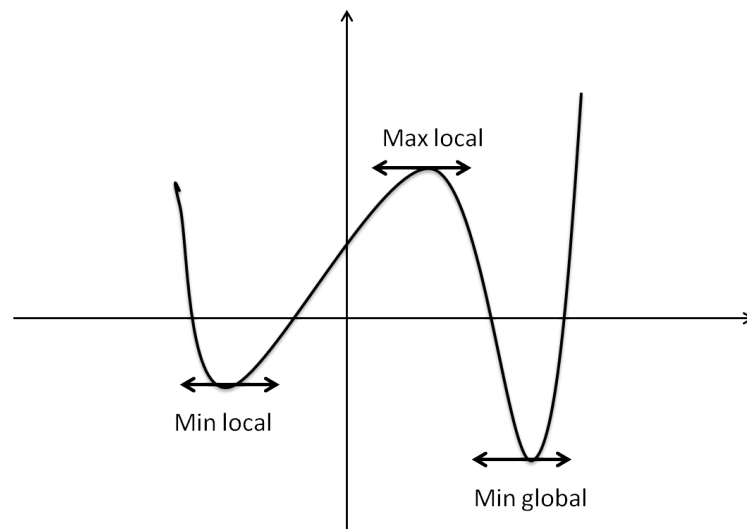


Figure 4.1- Optimum local et optimum global

Théorème 4.2(Weierstrass)[12]

Si f est une fonction réelle continue sur $D \subset \mathbb{R}^n$, compact (D fermé borné) alors le problème d'optimisation :

$$\begin{cases} \min f(x) \\ x \in D \end{cases} \quad (4.4)$$

a une solution optimale $x^* \in D$

4.4 Classification d'un programme mathématique

On classe le problème (4.1) selon des propriétés fondamentales, à savoir la convexité, la différentiabilité et la continuité des fonctions du problème (4.1).[12]

Définition 4.10

Un programme mathématique est dit convexe si : f , g et h sont linéaires sur un ensemble D des solutions réalisables convexe.

En général, f , g_i et h_j sont convexes donc le problème est convexe sur D convexe.

Si : f , g_i et h_j sont différentiables sur toutes les contraintes, le problème est dit différentiable. Signalons que, les programmes non convexes ou non différentiables sont les plus difficiles à traiter.

4.5 Condition d'optimalité(sans contraintes)

Soit le programme non linéaire et non contraints (PNC)[11] :

$$(P_3) \begin{cases} \min f(x) \\ x \in I \subseteq \mathbb{R}^n \end{cases} \quad (4.5)$$

Où la fonction f est au moins deux fois continument différentiables.

Définition 4.11

Soit $x^* \in I$, on dit que $y \in \mathbb{R}^n$ une direction admissible au point x^* s'il existe $\alpha > 0$ tel que :

$$x^* + ty \in I, \forall t \in [0, \alpha].$$

Soit x^* un minimum local de $f(x)$, on a alors nécessairement pour tout $t > 0$ assez petit.

$f(x^* + ty) - f(x^*) \geq 0$, $\forall y$ admissible ceci implique :

$$\lim_{t \rightarrow 0} \frac{f(x^* + ty) - f(x^*)}{t} = \nabla f(x^*) y \geq 0.$$

En particulier, la direction $y = -\nabla f(x^*)$ est admissible et on déduit :

$$-\|\nabla f(x^*)\|^2 \geq 0 \implies \|\nabla f(x^*)\|^2 \leq 0 \implies \nabla f(x^*) = 0.$$

Théorème 4.3 : (condition nécessaire du 1^{er} ordre) (cas sans contraintes)[11]

Soit x^* minimum local pour (4.5), on a alors $\nabla f(x^*) = 0$.

Un point x satisfaisant cette condition est appelé un point stationnaire.

Au deuxième ordre, on obtient :

$$f(x^*) \leq f(x^* + ty) = f(x^*) + t \nabla f(x^*)' y + \frac{1}{2} t^2 y' H_f(x^*) y + O(t^2) \text{ quand } t \text{ tend vers } 0.$$

Comme x^* est un minimum local on a :

$$\lim_{t \rightarrow 0} \frac{f(x^* + ty) - f(x^*)}{t^2} = \frac{1}{2} y' H_f(x^*) y + \lim_{t \rightarrow 0} \frac{O(t^2)}{t^2} = \frac{1}{2} y' H_f(x^*) y \geq 0$$

Théorème 4.4 : (condition nécessaire du 2^{ème} ordre)[11]

Soit x^* un minimum local pour (4.4) on a alors :

$$\begin{cases} \nabla f(x^*) = 0 \\ y' H_f(x^*) y \geq 0, \quad \forall y \in I \end{cases} \quad (4.6)$$

La matrice hessienne au point x^* est donc semi-définie positive. Ces conditions ne sont pas suffisantes.

Théorème 4.6.3 : (conditions suffisantes du 2^{ème} ordre)[11]

Soit x^* un point pour le problème (4.4) tel que :

$$\begin{cases} \nabla f(x^*) = 0 \\ y' H_f(x^*) y > 0, \quad \forall y \in I \end{cases} \quad (4.7)$$

Alors x^* est un minimum local (strict).

4.6 Conditions d'optimalité (avec contraintes)

Soit le programme mathématique (PM)[8] :

$$\begin{cases} \min f(x) \\ s.c \\ g_i(x) \leq 0, \quad i = \overline{1.m} \\ h_j(x) = 0, \quad j = \overline{1.p} \\ x \in \mathbb{R}^n \end{cases} \quad (4.8)$$

Où f, g_i et h_j sont définies $\mathbb{R}^n \rightarrow \mathbb{R}$ sont au moins deux fois continument différentiables.

Définition 4.12[7]

Le lagrangien du programme mathématique (4.7) est défini par :

$$L(x, \lambda, \mu) = f(x) + \sum_{i=1}^m \lambda_i g_i(x) + \sum_{j=1}^p \mu_j h_j(x).$$

$$\lambda_i \geq 0, \mu_j \in \mathbb{R}$$

4.7 Qualification des contraintes

Pour que la qualification des contraintes (QC) soit vérifiée en tout point x dans $D = \{x \in \mathbb{R}^n / g_i(x) \leq 0 \text{ et } h_j(x) = 0 / i = 1, \dots, m, j = 1, \dots, p\}$, il suffit que l'une des conditions (1) ou (2) soient réalisée[12] :

(1) - Toutes les contraintes de (4.7) sont linéaires, affines où convexe.

(2) - Toutes les contraintes d'inégalités $g_i(x)$ ($i = \overline{1..m}$) sont convexes et il existe $\bar{x} \in \mathbb{R}^n$ vérifiant $g_i(\bar{x}) < 0, \forall i = 1, \dots, m$ et $h_j(x)$ ($j = \overline{1..p}$) affines.

Pour que (QC) soit vérifiée en un point $x^0 \in D$ il suffit que l'on ait :

- Les gradients $\nabla g_i(x^0)$ ($i = 1 \dots m$), $\nabla h_i(x^0)$ ($i = \overline{1..p}$)
des contraintes saturées en x^0 sont linéairement indépendantes.

Remarque 4.3[12]

La résolution complète traite dans l'ordre les points suivants :

- L'existence d'une solution optimale.
- La caractérisation de la solution.
- L'élaboration d'algorithme pour calculer cette solution.

Théorème 4.5 : (Karush-Kuhn-Tucker)[8](avec contraintes, condition du 1^{er} ordre)

Soit x^* un minimum local régulier de (PM).

Alors, il existe les multiplicateurs $\lambda^* \in \mathbb{R}_+^m$ et $\mu^* \in \mathbb{R}^p$ tel que :

$$\left\{ \begin{array}{l} \nabla f(x^*) + \sum_{i=1}^m \lambda_i^* \nabla g_i(x^*) + \sum_{j=1}^p \mu_j^* \nabla h_j(x^*) = 0. \\ \text{avec} \\ (\nabla_x L(x^*, \lambda^*, \mu^*) = 0); \text{ (condition d'optimalité).} \\ \lambda_i^* g_i(x^*) = 0, \quad i = \overline{1..m}; \text{ (condition de complémentarité).} \\ \lambda_i^* \geq 0, \quad i = \overline{1..m}; \quad \quad \quad (K.K.T) \\ g_i(x^*) \leq 0, \quad i = \overline{1..m} \\ h_j(x^*) = 0, \quad j = \overline{1..p} \end{array} \right. \quad (4.9)$$

Remarque 4.4

1- Si les contraintes ne sont pas qualifiées en x^* , les conditions de K.K.T ne s'appliquent pas (x^* peut être optimal sans vérifier ces conditions).

2- Si (4.7) est convexe, les conditions de K.K.T sont à la fois nécessaires et suffisantes pour que x^* soit un minimum global.

Soit $I(x^*) = \{i / i = \overline{1, m} ; g_i(x^*) = 0\}$, est l'ensemble des indices des contraintes d'inégalités actives au point x^* .

Définition 4.13[12]

On définit un espace tangent $T(x^*)$ au point x^* d'un problème (4.7) par :

$$T(x^*) = \left\{ y \in \mathbb{R}^n : \nabla g_i(x^*)' y \leq 0, i \in I(x^*) \text{ et } \nabla h_j(x^*)' y = 0, j = 1, \dots, p \right\}.$$

Théorème 4.6 : (Conditions nécessaires du 2^{ème} ordre)[8]

Soit x^* un minimum local régulier de (4.7). Alors, il existe des multiplicateurs $\lambda^* \in \mathbb{R}_+^m$ et $\mu^* \in \mathbb{R}^p$ tel que :

- Les conditions de K.K.T sont satisfaites.
- $y' H_L(x^*, \lambda^*, \mu^*) y \geq 0, \forall y \in T(x^*)$, (Hessienne semi définie positive).

Théorème 4.7 : (conditions suffisantes du 2^{ème} ordre) [11]

S'il existe des multiplicateurs $\lambda^* \in \mathbb{R}_+^m$ et $\mu^* \in \mathbb{R}^p$ tels que :

les conditions de K.K.T sont satisfaites.

$$y' H_L(x^*, \lambda^*, \mu^*) y > 0, \quad \forall y \in T(x^*)$$

$T(x^*)$ est toujours un espace tangent du problème (4.7) au point x^* . Alors, le point x^* est un minimum local strict de (4.7) .

4.8 Programmation non linéaire floue flexible

Considérons le problème de programmation non linéaire, (noté PNL), suivant [9] :

$$\begin{cases} \max f(x) \\ \text{s.c} \\ g_i(x) \leq 0, i = \overline{1, m} \end{cases} \quad (4.10)$$

où $x' = (x_1, \dots, x_n)$ un vecteur à n composantes ; $f(x) = f(x_1, \dots, x_n)$ et $g_i(x) = g_i(x_1, \dots, x_n)$, $i = \overline{1, m}$, sont des fonctions réelles à n variables $x_1, \dots, x_n$.

En s'inspirant du modèle de PL flexible donné par Sakawa a porté les changements suivants au problème PNL : il est possible de relaxer l'exigence qui consiste en une maximisation flexible de la fonction objective sous des contraintes flexibles. Pour une telle situation, le problème de PNL est transformé en un problème de programmation non linéaire floue flexible (noté PNLF) donné sous la forme suivante :

$$\begin{cases} \widetilde{max} f(x) \\ s.c \\ g_i(x) \widetilde{\leq} 0, i = \overline{1, m} \end{cases} \quad (4.11)$$

où les symboles $\widetilde{max}$ et $\widetilde{\leq}$ représentent les relations flexibles ou bien floues de maximiser et $\widetilde{\leq}$ respectivement du cas usuel. Cela veut dire que “la fonction objective doit être maximisée aussi que possible” et “les contraintes doivent être satisfaites aussi bien que”, respectivement.

Toute exigence par rapport à la fonction objective $f(x)$ et aux contraintes $g_i(x)$, $i = \overline{1, m}$, peut être représentée par la fonction d'appartenance $\mu_i(g_i(x))$, $i = \overline{0, m}$, par convention, on note $\mu_0(f_i(x))$ par $\mu_0(g_0(x))$.

Pour exprimer le degré de satisfaction, il doit déterminer la fonction d'appartenance subjective $\mu(g_i(x))$, qui est une fonction monotone décroissante par rapport à g_i sous la forme suivante :

$$\mu_i(g_i(x)) = \begin{cases} 1 & si \ g_i(x) \leq g_i^1 \\ d_i(x) & si \ g_i^1 \leq g_i(x) \leq g_i^0, \ i = \overline{0, m} \\ 0 & si \ g_i^0 \leq g_i(x) \end{cases} \quad (4.12)$$

où

g_i^1 représente la valeur de g_i pour laquelle la fonction d'appartenance est égale à 1 ;

g_i^0 représente la valeur de g_i pour laquelle la fonction d'appartenance est égale à 0 ;

pour les valeurs intermédiaires, la valeur de g_i est exprimée par $d_i(x)$ qui est une fonction monotone décroissante par rapport à g_i .

En utilisant les fonctions d'appartenance et en se basant sur le principe de Bellman et Zadeh, Sakawa a défini “le degré total de satisfaction ” du décideur par la fonction d'appartenance suivante :

$$\mu_D(x) = \min_{i=\overline{0, m}} \{\mu_i(g_i(x))\}$$

La solution est alors, celle qui réalise le meilleur degré de satisfaction, c'est à dire :

chercher la solution x^* telle que $\max_x \min_{i=\overline{0, m}} \{\mu_i(g_i(x))\}$.

En introduisant une variable auxiliaire λ , le problème est transformé au problème de maximisation suivant :

$$\begin{cases} max \lambda \\ \lambda \leq \mu_i(g_i(x)), \ i = \overline{0, m} \\ 0 \leq \lambda \leq 1 \end{cases} \quad (4.13)$$

Dans ce cas, on peut utiliser n'importe quel algorithme de programmation non linéaire, pour résoudre ce problème.

Remarque 4.5

Les différentes approches proposées dans la littérature, pour résoudre les problèmes d'optimisation floue (PLF et PNLf), dépendent :

1. de la façon dont le flou intervient dans le modèle (inégalités floues, contraintes floues, maximisation floue, fonction floue).
2. de la nature de l'aspect flou dans le modèle (nombre flou selon Dubois, nombre flou du type L-R, nombre flou triangulaire symétrique ou non symétrique).

4.9 Cas particulier de la programmation mathématique

4.9.1 Problème quadratique

Définition 4.14[16]

Un problème d'optimisation avec une fonction quadratique et des contraintes linéaires est appelé un programme quadratique. Le problème quadratique général peut s'écrire comme suit :

$$\left\{ \begin{array}{l} \min f(x) = \frac{1}{2}x'Dx + c'x + g \\ s.c \\ Ax = b \\ Qx \leq h \\ x \in \mathbb{R}^n \end{array} \right. \quad (4.14)$$

Où D est une matrice symétrique, c, x sont des n -vecteurs, A est une (p, n) matrice avec $\text{rang}(A) = p$, Q est une $(m * n)$ matrice, $\text{rang}(Q) = m$, b est un p -vecteurs, et h est un m -vecteurs, $g \in \mathbb{R}$.

Remarque 4.6

Si la matrice D est semi définie positive, on dit que le problème est un problème quadratique convexe (le problème est dit strictement convexe si D est définie positive).

4.9.2 Résolution d'un problème quadratique convexe.

4.9.2.1 Méthode quadratique de Wolfe (1959)

Introduction[16]

Beaucoup d'algorithmes ont été développés pour la résolution du problème de programmation quadratique convexe, mais il serait intéressant de connaître la méthode la plus classique de Wolfe, qui n'est autre que la méthode du simplexe légèrement modifiée.

Le principe de cette méthode est la résolution du système de Kuhn-tucker et consiste à trouver une solution réalisable pour un système linéaire avec une condition supplémentaire du type $x_j \delta_j = 0$.

Où x et δ sont des vecteurs de même dimension. En d'autre terme c'est trouver une solution réalisable basique en résolvant un problème de programmation linéaire, assujetti à la nouvelle condition.

Cas d'un problème quadratique standard[16]

$$\begin{cases} \min f(x) = \frac{1}{2}x'Dx + c'x \\ s.c \\ Ax = b \\ x \geq 0 \end{cases} \quad (4.15)$$

Où $D' = D \geq 0$, $c \in \mathbb{R}^n$, $\text{rang} A = m < n$,

Ce problème peut encore s'écrire sous la forme équivalente suivant :

$$\begin{cases} \min f(x) = \frac{1}{2}x'Dx + c'x \\ s.c \\ Ax - b \leq 0 \\ -Ax + b \leq 0 \\ -x \leq 0 \end{cases} \quad (4.16)$$

Le point minimum x^* est alors caractérisé par les équations et les inégalités suivantes :
Il existe deux m -vecteurs $\lambda_1^* \geq 0$, $\lambda_2^* \geq 0$ ainsi qu'un vecteur $\delta \geq 0$ tels que

$$\begin{cases} \frac{\partial L}{\partial x}(x^*, \lambda_1^*, \lambda_2^*, \delta^*) = 0 \\ \lambda_1^* (Ax^* - b) = 0 \\ \lambda_2^* (-Ax^* + b) = 0 \\ \delta^{*'} x^* = 0 \\ Ax^* - b = 0 \\ x^* \geq 0 \\ \delta^* \geq 0 \end{cases} \quad (4.17)$$

où

$$L(x, \lambda_1, \lambda_2, \delta) = \frac{1}{2}x'Dx + c'x + \lambda_1'(Ax - b) + \lambda_2'(-Ax + b) - \delta'x,$$

$$\frac{\partial L}{\partial x}(x, \lambda_1, \lambda_2, \delta) = Dx + c + A'\lambda_1 - A'\lambda_2 - \delta,$$

Posant $\lambda = \lambda_1 - \lambda_2$, $\lambda \in \mathbb{R}^m$, le point x^* est défini par le système suivant :

$$\begin{cases} Ax^* = b & (L_2) \\ \delta^{*'} x^* = 0 & (L_3) \\ \lambda^{*'} (Ax^* - b) = 0 & (L_4) \\ x^* \geq 0, \lambda^* \in \mathbb{R}^m, \delta^* \geq 0 & (L_5) \end{cases} \quad (4.18)$$

Comme l'équation (L_4) est tout le temps vérifiée, le système se réduit ainsi :

$$\begin{cases} Dx^* + A'\lambda^* - \delta^* = -c & (L_1) \\ Ax^* = b & (L_2) \\ \delta^{*'} x^* = 0 & (L_3) \\ x^* \geq 0, \lambda^* \in \mathbb{R}^m, \delta^* \geq 0 \end{cases} \quad (4.19)$$

Un tel système n'est pas linéaire par rapport au multi-vecteur $(x^*, \lambda^*, \delta^*)$ à cause de l'équation (L_3) . On obtient donc un système linéaire de $(n + m)$ équations à $(n + m + n)$ inconnues, constitué des équations (L_1) et (L_2) , avec en plus n équations non linéaires $\delta x_j = 0, j = \overline{1.n}$. Pour trouver une solution telle que $\delta x_j = 0, j = \overline{1.n}$, il suffit d'obtenir une solution réalisable basique du système linéaire, avec x_j basique et δ_j non basique ou vice-versa. Pour cela, on appliquera la première phase du simplexe et on choisira l'indice j_0 du vecteur qui entre dans la base de telle sorte que les vecteurs-colonnes correspondant à x_{j_0} et δ_{j_0} ne se retrouvent pas en même temps dans. Pour appliquer la méthode du simplexe, il faut alors écrire le système $(L_1), (L_2)$ sous forme standard, à savoir que le second membre doit être positif ou nul, ainsi que le vecteur λ^* qui doit être réécrit sous la forme :

$$\lambda_i^* = \alpha_i^* - \alpha_{m+i}^*, \quad \alpha_{m+i}^* \geq 0, \quad \alpha_i^* \geq 0, \quad i = \overline{1.m}$$

4.9.3 Algorithme

Début

1. Introduire les données, D, A, b, c ;

Appliquer les conditions de K.K.T au problème ;

Déterminer les équations de K.K.T ;

Détermination des paramètres du programme linéaire ;

2. Détermination des paramètres du programme linéaire ;

Introduire les variables artificielle ν_i ;

Construire la matrice des contraintes A

Construire le vecteur du second membre b

Construire le vecteur des coûts c ;

3. Initialiser le vecteur solution $(x, \lambda, \delta, \nu)$;

Déterminer l'ensemble des indices J_B et J_H ;

Extraire les éléments de base x_B, c_B, A_B ;

4. Calculer le vecteur des potentiels $y' = c_B' A_B^{-1}$;

Calculer le vecteur des estimations $\Delta_H = y' A_H - c_H'$;

Si $\Delta_H \geq 0$ Alors la solution actuelle est optimale ;

Fin si

Sinon aller à (5) ;

5. Déterminer la variable qui entre en base tout en vérifiant la condition

$$\delta_j x_j = 0, \quad j = \overline{1, n};$$

Déterminer la variable qui sort de la base ;

Mettre à jour A_B, x_B, c_B, J_B, J_H et aller en 4 ;

Fin[16].

Exemple 4.1 Soit à résoudre le problème non linéaire flou suivant :

$$\left\{ \begin{array}{l} \min f(x) = x_1^2 + x_1 x_2 + 6x_2^2 - 2x_1 + 8x_2 \\ s.c \\ \tilde{a}_1 x_1 + \tilde{a}_2 x_2 \leq \tilde{b} \\ x_1 \geq 0, x_2 \geq 0 \end{array} \right. \quad (4.20)$$

avec

$$\begin{aligned} \tilde{a}_1 &= (1, 2, 3); \\ \tilde{a}_2 &= (4, 5, 6); \\ \tilde{b} &= (5, 7, 9). \end{aligned}$$

$$\left\{ \begin{array}{l} \min f(x) = x_1^2 + x_1 x_2 + 6x_2^2 - 2x_1 + 8x_2 \\ s.c \\ x_1 + 4x_2 \leq 5 \\ 2x_1 + 5x_2 \leq 7 \\ 3x_1 + 6x_2 \leq 9 \\ x_1 \geq 0, x_2 \geq 0 \end{array} \right. \quad (4.21)$$

$$\left\{ \begin{array}{l} \min f(x) = x_1^2 + x_1 x_2 + 6x_2^2 - 2x_1 + 8x_2 \\ s.c \\ x_1 + 4x_2 + x_3 = 5 \\ 2x_1 + 5x_2 + x_4 = 7 \\ 3x_1 + 6x_2 + x_5 = 9 \\ x_j \geq 0, j = \overline{1, 5} \end{array} \right. \quad (4.22)$$

Si $(x_1, x_2, x_3, x_4, x_5)$ est un point minimum de la fonction f , alors $\exists \lambda \in \mathbb{R}^3, \delta_1, \delta_2, \delta_3, \delta_4, \delta_5 \geq 0$ tel que :

$$L(x.\lambda.\partial) = x_1^2 + x_1x_2 + 6x_2^2 - 2x_1 + 8x_2 + \lambda_1(x_1 + 4x_2 + x_3 - 5) + \lambda_2(2x_1 + 5x_2 + x_4 - 7) + \lambda_3(3x_1 + 6x_2 + x_5 - 9) - x_1\delta_1 - x_2\delta_2 - x_3\delta_3 - x_4\delta_4 - x_5\delta_5$$

Alors on obtient le système suivant :

$$\frac{\partial L}{\partial x_1} = 2x_1 + x_2 - 2 + \lambda_1 + 2\lambda_2 + 3\lambda_3 - \delta_1.$$

$$\frac{\partial L}{\partial x_2} = x_1 + 12x_2 + 8 + 4\lambda_1 + 5\lambda_2 + 6\lambda_3 - \delta_2.$$

$$\frac{\partial L}{\partial x_3} = \lambda_1 - \delta_3 = 0 \iff \lambda_1 = \delta_3.$$

$$\frac{\partial L}{\partial x_4} = \lambda_2 - \delta_4 = 0 \iff \lambda_2 = \delta_4.$$

$$\frac{\partial L}{\partial x_5} = \lambda_3 - \delta_5 = 0 \iff \lambda_3 = \delta_5.$$

et

$$\lambda_1(x_1 + 4x_2 + x_3 - 5) = 0$$

$$\lambda_2(2x_1 + 5x_2 + x_4 - 7) = 0$$

$$\lambda_3(3x_1 + 6x_2 + x_5 - 9) = 0$$

$$\left\{ \begin{array}{ll} 2x_1 + x_2 - \delta_1 + \delta_3 + 2\delta_4 + 3\delta_5 = 2 & L_1 \\ x_1 + 12x_2 - \delta_2 + 4\delta_3 + 5\delta_4 + 6\delta_5 = -8 & L_2 \\ x_1 + 4x_2 + x_3 = 5 & L_3 \\ 2x_1 + 5x_2 + x_4 = 7 & L_4 \\ 3x_1 + 6x_2 + x_5 = 9 & L_5 \\ \delta_j x_j = 0 & \\ x_j \geq 0, \delta_j \geq 0, j = \overline{1,5} & \end{array} \right. \quad (4.23)$$

En multipliant (L_2) par (-1) on aura :

$$\left\{ \begin{array}{ll} 2x_1 + x_2 - \delta_1 + \delta_3 + 2\delta_4 + 3\delta_5 = 2 & \\ -x_1 - 12x_2 + \delta_2 - 4\delta_3 - 5\delta_4 - 6\delta_5 = 8 & \\ x_1 + 4x_2 + x_3 = 5 & \\ 2x_1 + 5x_2 + x_4 = 7 & \\ 3x_1 + 6x_2 + x_5 = 9 & \\ x_j \geq 0, \delta_j \geq 0, j = \overline{1,5} & \end{array} \right. \quad (4.24)$$

Nous avons obtenu un système linéaire de cinq équations et dix inconnus.

Pour trouver une solution telle que $\delta_j x_j = 0, j = \overline{1,5}$, il suffit d'obtenir une solution réalisable basique (x, δ) du système linéaire avec x_j basique et δ_j hors base ou vice versa. Pour cela il faut choisir l'indice j_0 entrant dans la base de telle sorte que les vecteurs-colonnes correspondant à x_{j_0} et δ_{j_0} ne se trouvant pas en même temps dans la base.

Appliquons la première phase du simplexe :

On considère le problème de programmation linéaire suivant :

$$\left\{ \begin{array}{l} \max Z = -\nu_1 \\ s.c \\ 2x_1 + x_2 - \delta_1 + \delta_3 + 2\delta_4 + 3\delta_5 + \nu_1 = 2 \\ -x_1 - 12x_2 + \delta_2 - 4\delta_3 - 5\delta_4 - 6\delta_5 = 8 \\ x_1 + 4x_2 + x_3 = 5 \\ 2x_1 + 5x_2 + x_4 = 7 \\ 3x_1 + 6x_2 + x_5 = 9 \\ x_j \geq 0, \delta_j \geq 0, j = \overline{1,5}, \nu_1 \geq 0 \end{array} \right. \quad (4.25)$$

Le vecteur :

$$\bar{x}(x, \delta, \nu_1) = (x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}) = (0, 0, 5, 7, 9, 0, 8, 0, 0, 0, 2)$$

est une solution réalisable initiale basique du problème.

Dressons alors les tableaux simplexe suivant :

c			0	0	0	0	0	0	0	0	0	0	-1	
c_B	base	b	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	θ
-1	x_1	2	2	1	0	0	0	-1	0	1	2	3	1	$\frac{2}{2}$
0	x_7	8	-1	-12	0	0	0	0	1	-4	-5	-6	0	/
0	x_3	5	1	4	1	0	0	0	0	0	0	0	0	5
0	x_4	7	2	5	0	1	0	0	0	0	0	0	0	3.5
0	x_5	9	3	6	0	0	1	0	0	0	0	0	0	3
	Δ_j	-2	-1	0	0	0	0	1	0	-1	-2	-3	0	/

Remarquons que $j_0 = 1$ est l'indice qui entre dans la base tandis que l'indice $j_0 = 11$ sort, car l'indice $j_0 = 8$, $j_0 = 9$ et $j_0 = 10$ ne peuvent pas être choisis à cause de la présence de a_3, a_4, a_5 dans la base, afin que la condition $x_3\delta_3 = 0$, $x_4\delta_4 = 0$ et $x_5\delta_5 = 0$ sont assurées.

c			0	0	0	0	0	0	0	0	0	0	0	0
c_B	base	b	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	
0	x_1	1	1	$\frac{1}{2}$	0	0	0	$-\frac{1}{2}$	0	$\frac{1}{2}$	1	$\frac{3}{2}$	$\frac{1}{2}$	
0	x_7	9	0	$-\frac{23}{2}$	0	0	0	-1	1	-3	-3	-3	1	
0	x_3	4	0	$\frac{7}{2}$	1	0	0	$\frac{1}{2}$	0	$-\frac{1}{2}$	-1	$-\frac{3}{2}$	$-\frac{1}{2}$	
0	x_4	5	0	4	0	1	0	1	0	-1	-2	-3	-1	
0	x_5	6	0	$\frac{9}{2}$	0	0	1	$\frac{3}{2}$	0	$-\frac{3}{2}$	-3	$-\frac{9}{3}$	$-\frac{3}{2}$	
	Δ_j		0	0	0	0	0	0	0	0	0	0	1	

Le critère d'optimalité étant vérifié, il s'ensuit que

le vecteur $\bar{x}(x, \delta, \nu_1) = (1, 0, 4, 5, 6, 0, 9, 0, 0, 0)$ est une solution optimale du problème linéaire, par conséquent, la solution réalisable basique du système d'optimalité du problème quadratique et le vecteur $(\hat{x}, \hat{\delta})$ telle que :

$$\hat{x} = (\hat{x}_1, \hat{x}_2, \hat{x}_3, \hat{x}_4, \hat{x}_5) = (1, 0, 4, 5, 6)$$

$$\hat{\delta} = (\hat{\delta}_1, \hat{\delta}_2, \hat{\delta}_3, \hat{\delta}_4, \hat{\delta}_5) = (0, 9, 0, 0, 0)$$

donc le point minimum du problème est le vecteur $x^* = (1, 0)$, avec $f(x^*) = -1$.

Exemple 4.2

On considère le programme non linéaire flexible :

$$\left\{ \begin{array}{l} \widetilde{min} f(x) = x_1^2 + x_1x_2 + 6x_2^2 - 2x_1 + 8x_2 \\ s.c \\ -x_1 + 2x_2 \leq 5 \\ 3x_1 + x_2 \leq 11 \\ x_1 \geq 0, x_2 \geq 0 \end{array} \right. \quad (4.26)$$

1) On détermine les fonctions d'appartenance de l'objectif et des contraintes sachant que l'objectif excède la valeur $Z_0 = 1$ et que les écarts de tolérance de l'objectif et des contraintes sont respectivement $d_0=2, d_1=5, d_2 =1$:

Transformation de l'objectif

$$\mu_0(x) \left\{ \begin{array}{ll} 1 & si \quad x_1^2 + x_1x_2 + 6x_2^2 - 2x_1 + 8x_2 \geq 1 \\ 1 - \frac{1 - (x_1^2 + x_1x_2 + 6x_2^2 - 2x_1 + 8x_2)}{2} & si \quad -1 \leq x_1^2 + x_1x_2 + 6x_2^2 - 2x_1 + 8x_2 \leq 2 \\ 0 & si \quad x_1^2 + x_1x_2 + 6x_2^2 - 2x_1 + 8x_2 < -1 \end{array} \right. \quad (4.27)$$

$$\mu_0(x) \left\{ \begin{array}{ll} 1 & si \quad x_1^2 + x_1x_2 + 6x_2^2 - 2x_1 + 8x_2 \geq 1 \\ \frac{x_1^2 + x_1x_2 + 6x_2^2 - 2x_1 + 8x_2 + 1}{2} & si \quad -1 \leq x_1^2 + x_1x_2 + 6x_2^2 - 2x_1 + 8x_2 \leq 2 \\ 0 & si \quad x_1^2 + x_1x_2 + 6x_2^2 - 2x_1 + 8x_2 < -1 \end{array} \right. \quad (4.28)$$

La transformation de la 1^{ère} contraintes et la 2^{ème} contraintes est faite dans l'exemple 3.1

$$\left\{ \begin{array}{l} max \lambda \\ s.c \\ \lambda \leq \frac{x_1^2 + x_1x_2 + 6x_2^2 - 2x_1 + 8x_2 + 1}{2} \\ \lambda \leq \frac{x_1 - 2x_2 + 10}{5} \\ \lambda \leq -3x_1 - x_2 + 12 \\ x_1 \geq 0, x_2 \geq 0, 0 \leq \lambda \leq 1 \end{array} \right. \quad (4.29)$$

$$\left\{ \begin{array}{l} \max \lambda \\ s.c \\ 2\lambda - x_1^2 - x_1x_2 - 6x_2^2 + 2x_1 - 8x_2 \leq 1 \\ 5\lambda - x_1 + 2x_2 \leq 10 \\ \lambda + 3x_1 + x_2 \leq 12 \\ x_1 \geq 0, x_2 \geq 0, 0 \leq \lambda \leq 1 \end{array} \right. \quad (4.30)$$

le point minimum du problème est le vecteur $x^* = (1.23, 1.23)$, et $\lambda = 1$ avec $f(x^*) = 1$ cette solution est trouvé avec lingo.

Conclusion

Nous avons présenté, dans ce chapitre des notions de la programmation mathématique, en particulier, nous avons résolu un programme non linéaire floue avec la méthode du simplexe quadratique de Wolfe, qui n'est autre que la méthode du simplexe légèrement modifiée dont son principe réside dans la résolution du système de Karush-Kuhn-Tucher .

Chapitre 5

Application sur lingo

Lingo : Logiciel pour la résolution des programmes linéaires et non linéaires.

5.1 Introduction

Lingo est un logiciel utilisé pour résoudre les modèles d'optimisation linéaire, entier et quadratique, il est aussi utilisé pour résoudre les modèles d'optimisation globale non linéaire. Une des caractéristique de lingo c'est qu'il offre des outils qui peuvent aider à l'analyse des modèles en utilisant la méthode du simplexe.

5.2 Installation du logiciel

Pour utiliser cette version de lingo il est conseillé d'avoir au moins un processeur 486 et 8Mo de mémoire RAM. Il faut aussi prévoir un espace disque dur de 2 Mo pour pouvoir l'installer. Les étapes de l'installation sont :

1. Démarrer windows
2. Insérer CD-ROM
3. Cliquer sur l'icône setup (Install) dans votre explorateur de Windows
4. Suivre les instructions de l'écran. Pour plus d'information sur ce logiciel visiter l'adresse web www.lingo.com.

Application 1 : Exemple 2.2

Programmation du problème linéaire.

The screenshot shows the Lingo 13.0 interface with a 'Solution Report - Lingo1' window open. The main window displays the Lingo model code, and the report window shows the results of the solver.

Model Code:

```
max=2*x1+x2;  
x1+x2<=10;  
-x1+2*x2<=10;  
end
```

Solution Report - Lingo1

Global optimal solution found.
Objective value: 20.00000
Infeasibilities: 0.000000
Total solver iterations: 1
Model Class: LP

Total variables: 2
Nonlinear variables: 0
Integer variables: 0

Total constraints: 3
Nonlinear constraints: 0

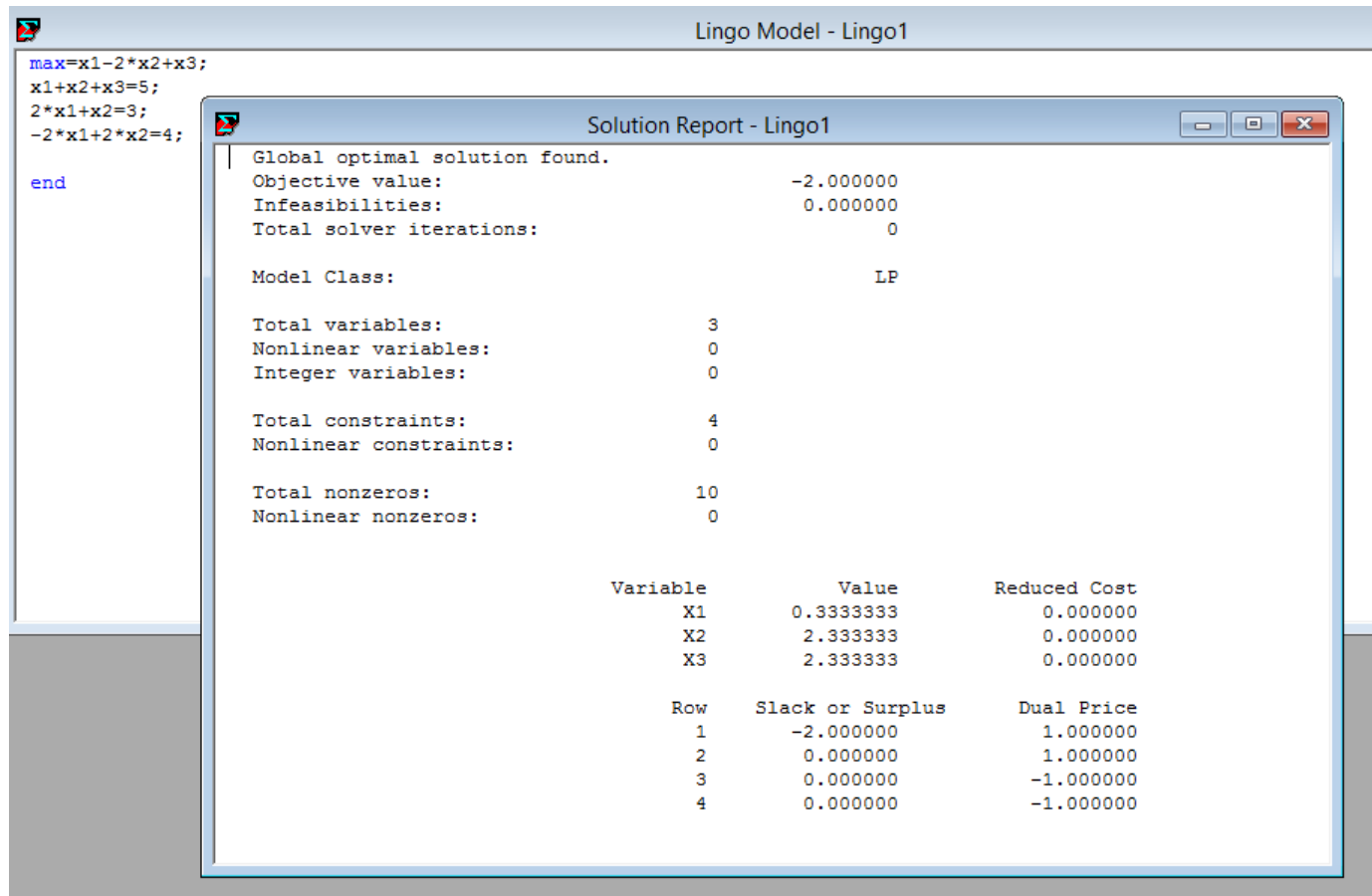
Total nonzeros: 6
Nonlinear nonzeros: 0

Variable	Value	Reduced Cost
X1	10.00000	0.000000
X2	0.000000	1.000000

Row	Slack or Surplus	Dual Price
1	20.00000	1.000000
2	0.000000	2.000000
3	20.00000	0.000000

Application 2 : Exemple 2.3

Programmation du problème linéaire.



The screenshot shows the Lingo Model window and the Solution Report window. The Lingo Model window contains the following code:

```
max=x1-2*x2+x3;  
x1+x2+x3=5;  
2*x1+x2=3;  
-2*x1+2*x2=4;  
  
end
```

The Solution Report window displays the following information:

Global optimal solution found.
Objective value: -2.000000
Infeasibilities: 0.000000
Total solver iterations: 0

Model Class: LP

Total variables: 3
Nonlinear variables: 0
Integer variables: 0

Total constraints: 4
Nonlinear constraints: 0

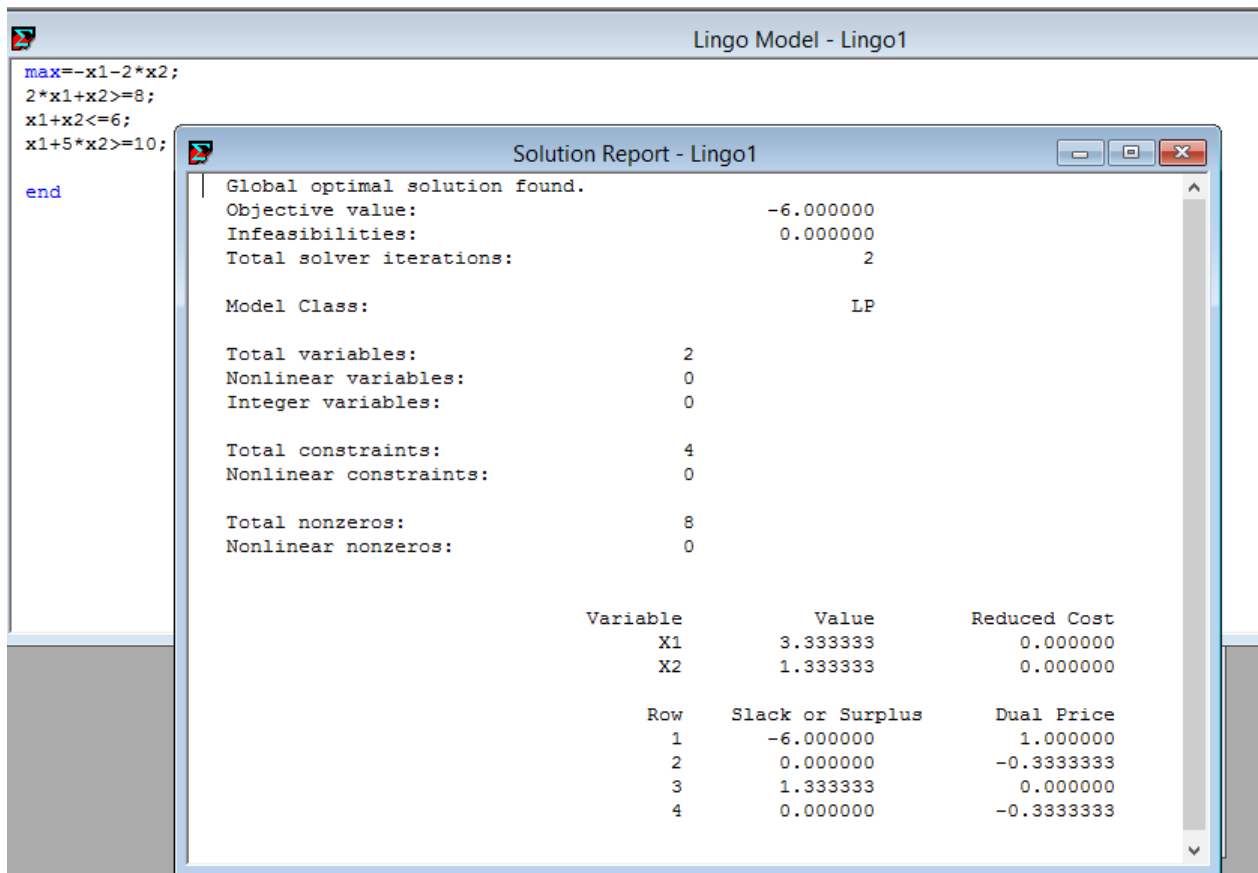
Total nonzeros: 10
Nonlinear nonzeros: 0

Variable	Value	Reduced Cost
X1	0.3333333	0.000000
X2	2.333333	0.000000
X3	2.333333	0.000000

Row	Slack or Surplus	Dual Price
1	-2.000000	1.000000
2	0.000000	1.000000
3	0.000000	-1.000000
4	0.000000	-1.000000

Application 3 : Exemple 2.4

Programmation du problème linéaire.



The screenshot shows the Lingo Model window and the Solution Report window. The Lingo Model window contains the following code:

```
max=-x1-2*x2;  
2*x1+x2>=8;  
x1+x2<=6;  
x1+5*x2>=10;  
  
end
```

The Solution Report window displays the following information:

Global optimal solution found.
Objective value: -6.000000
Infeasibilities: 0.000000
Total solver iterations: 2
Model Class: LP
Total variables: 2
Nonlinear variables: 0
Integer variables: 0
Total constraints: 4
Nonlinear constraints: 0
Total nonzeros: 8
Nonlinear nonzeros: 0

Variable	Value	Reduced Cost
X1	3.333333	0.000000
X2	1.333333	0.000000

Row	Slack or Surplus	Dual Price
1	-6.000000	1.000000
2	0.000000	-0.3333333
3	1.333333	0.000000
4	0.000000	-0.3333333

Application 4 : Exemple 3.1

Programmation du problème linéaire flexible.

The screenshot displays the LINGO software interface. The main window, titled "Lingo Model - Lingo1", contains the following model code:

```
Model:
max= y;
2*y+5*x1-x2<=8;
5*y-x1+2*x2<=10;
y+3*x1+x2<=12;
y<=1;
y>=0;

end
```

Overlaid on the main window is the "Solution Report - Lingo1" dialog box. It reports the following information:

Global optimal solution found.

Objective value:	1.000000
Infeasibilities:	0.000000
Total solver iterations:	0

Model Class: LP

Total variables:	3
Nonlinear variables:	0
Integer variables:	0

Total constraints:	6
Nonlinear constraints:	0

Total nonzeros:	12
Nonlinear nonzeros:	0

Variable	Value	Reduced Cost
Y	1.000000	0.000000
X1	0.000000	0.000000
X2	0.000000	0.000000

Row	Slack or Surplus	Dual Price
1	1.000000	1.000000
2	6.000000	0.000000
3	5.000000	0.000000
4	11.00000	0.000000
5	0.000000	1.000000
6	1.000000	0.000000

Application 5 : Exemple 3.2

Programmation du problème linéaire.

The image shows two windows from the Lingo software. The top window, titled "Lingo Model - Lingo1", contains the following model code:

```
max=-x1+2*x2;  
x1+2*x2>=-2;  
2*x1+3*x2<=6;  
x2>=-1;  
  
end
```

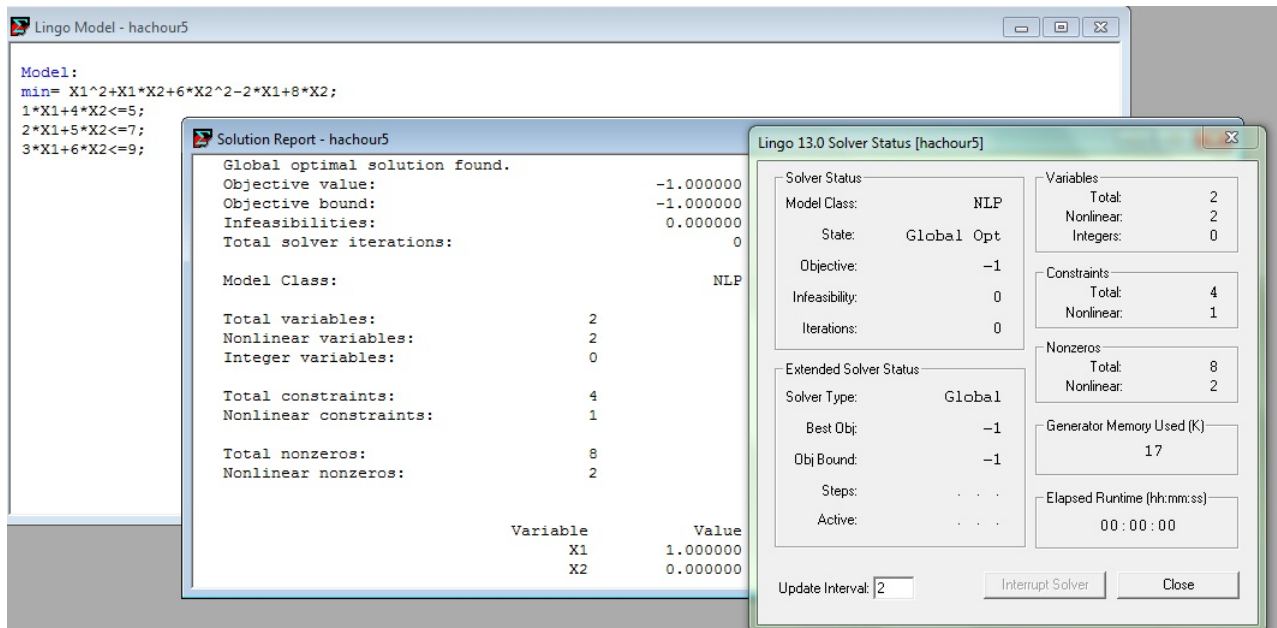
The bottom window, titled "Solution Report - Lingo1", displays the results of the optimization. It reports a global optimal solution found with an objective value of 4.000000 and zero infeasibilities. The model is classified as LP. The solution report includes the following data:

Variable	Value	Reduced Cost
X1	0.000000	2.333333
X2	2.000000	0.000000

Row	Slack or Surplus	Dual Price
1	4.000000	1.000000
2	6.000000	0.000000
3	0.000000	0.666667
4	3.000000	0.000000

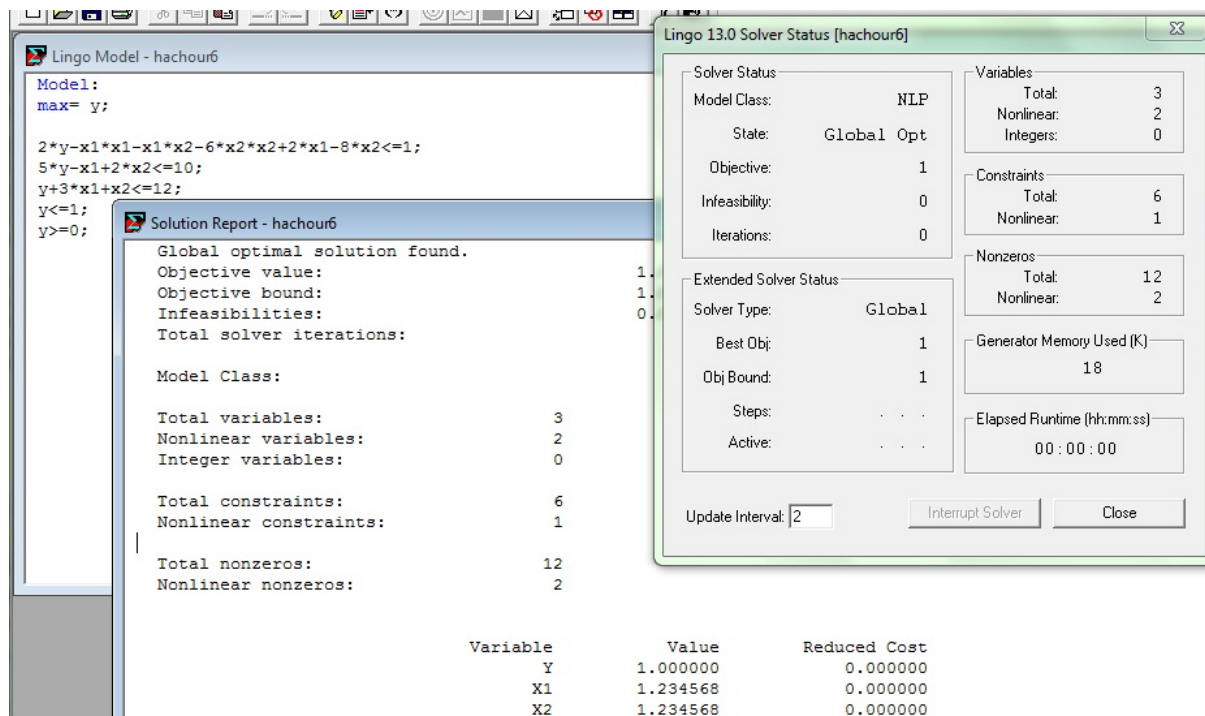
Application 6 : Exemple 4.1

Programmation du problème non linéaire.



Application 7 : Exemple 4.2

Programmation du problème non linéaire.



Conclusion générale

Depuis plusieurs années, on considère que les deux sources d'incertitude principales sont : le manque d'information et la variabilité des phénomènes. On modélise, alors, les informations soit par distribution de probabilité (information aléatoire) soit par des ensembles flous (informations incomplètes). La théorie des ensembles flous apparaît comme un outil bien adapté pour modéliser un concept vague.

Dans notre travail, nous avons abordé des programmes linéaires dont les données sont supposées être connues avec précision qui sont appelés des problèmes linéaires d'optimisation déterministe dont la résolution s'est faite par la méthode du simplexe.

Ensuite, nous avons traité des programme linéaire, dont les données sont approximatives ou vague, qui sont appelés des problèmes linéaires d'optimisation flous. Nous avons proposés pour résoudre un programme linéaire flou, deux méthodes : *Flexible* (fonction caractéristique) et *Robuste*, et cela en transformant ce programme en un programme linéaire déterministe qui sera résolu par la méthode du simplexe.

L'originalité de notre travail est de proposé deux nouvelles techniques de résolution du programme mathématique flous appliqué à deux problèmes différents tel que :

- 1- Problème du programmation quadratique de type triangulaire.
- 2- Programmation mathématique flexible.

Cette thèse nous a permis de prendre connaissance des techniques de résolution de programme mathématique dans les ensembles flous.

Ce qui est déjà pour nous une grande découverte et un champ nouveau. Pour perspective, nous pouvons envisager une suite pour ce travail dans les sens suivant :

- Résolution d'un problème mathématique flous quelconque avec quelques paramètres flous .
- Résolution d'un problème mathématique quelconque ou tous les paramètres du problème sont flous .

Bibliographie

- [1] Aiche F., sur la programmation linéaire multi-objectif floue et stochastique, Thèse de Doctorat, Université de Tizi Ouzou (2013).
- [2] Aiden M., Oukacha B., Programmation linéaire, Editions Pages Bleues (2005).
- [3] Amor F., Programmation linéaire, Office des Publications Universitaires.
- [4] Antoine Massoud., Mémoire Présenté En Vue De l'Obtention Du Diplôme De Maitrise Et Sciences Appliquées (Mathématiques Appliquées), Juin 1999 @ Antoine.
- [5] Dilem K., Hammad L., Programmation Multi-objectif et application aux ensembles flous ,Thèse de Master ,Université de Tizi Ouzou (2015).
- [6] Dubois D., Prade H., Fuzzy Sets and Systems : Theory and Applications. Academic Press, New York,(1980).
- [7] Hamid K.,Programmation Mathématique bi-niveaux et Contrôle Optimal bi-niveaux, Mémoire De Magister,Université de Tizi Ouzou (2010).
- [8] Gilbert J. Ch. , Élément d'Optimisation Différentiable,Théorie et Algorithme Version du 19 mars (2007).
- [9] kacher F., Concept d'équilibre pour un jeu non coopératif sous forme normale avec paramètre indéterminés flous,Thèse de doctorat ,Université de Tizi Ouzou (2006).
- [10] Luhandjula M.K., Fuzzy optimization : An appraisal, Fuzzy Sets and Systems 146 (2004) 187-203.Michel B., Introduction à l'optimisation différentiable, presse polytechnique et universitaire romandes (2006).
- [11] Michel B., Introduction à l'optimisation différentiable, presse polytechnique et universitaire romandes(2006).
- [12] Michel M., Programmation mathématique,Théorie et Algorithme 2eme édition,TEC et DOC (2009).
- [13] Nasser S. H., and Ardil E., Simplex Method for Fuzzy Variable Linear Programming Problems. International Scholarly and Scientific Research & Innovation, 3(10)36-40,(2009).
- [14] Nasser S. H., Ardil E., Yazdani A., and Zaefarian R., Simplex Method for Solving Linear Programming Problems With Fuzzy Numbers. International Scholarly and Scientific Research & Innovation, 1(10) 500-504, (2007).
- [15] Noora A. A., Karami P., Ranking Function and its Application to fuzzy DEA , Ranking Functions and its Application to Fuzzy DEA - Hikari. Publier en ligne : [www.m-hikari.com /imf... / 29 -32.../ nooraIMF 29-32-2008.](http://www.m-hikari.com/imf.../29-32.../nooraIMF29-32-2008.) ,(2008).
- [16] P.wolfe., the simplex method for quadratic programming. Econometrica (1959).
- [17] Scheid J-F., Graphe et recherche opérationnelle, Graphe et Recherche Opérationnelle. ESIAL. publier en ligne : www.isig.ac.cd/isiggoma/data/chier/5568820f13cf6.pdf (2011).
- [18] Teghem J., Programmation Linéaire. Édition ellipses SMA éditions de l'université de Bruxelles(1996).
- [19] yadolah D., optimisation appliquée, Sylvie GomanWeber et jean pierre renbfer édition spinger France (2005).