MINISTÈRE DE l'ENSEIGNEMENT SUPÉRIEUR
ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITÉ MOULOUD MAMMERI, TIZI-OUZOU

FACULTÉ DES SCIENCES, DÉPARTEMENT DE MATHÉMATIQUES

MÉMOIRE DE FIN D'ÉTUDES

Domaine : Mathématiques et Informatique

Spécialité : Mathématiques Appliquées

Option : Recherche Opérationnelle

Présenté par :

LADJICI Cylia et DACI Sara Ouarda

Thème

Optimisation mathématique théorie et pratique de l'analyse convexe à la programmation dynamique.

Devant le jury d'examen composé de :

Mme. OUBAKOUK Lynda	**Présidente**
Mme. LESLOUS Fadila	**Rapporteur**
Mme. SLIMI Farida	**Examinatrice**

Soutenue : 2024 - 2025

Remerciements

Nous remercions **DIEU**,
pour la force, la patience et la volonté qu'il nous a accordées tout au long de ce parcours.

Nous exprimons notre profonde gratitude à notre encadrante, **Madame LESLOUS**,
pour sa disponibilité et son accompagnement bienveillant,
qui ont largement contribué à l'élaboration de ce mémoire.

Nos remerciements vont également aux **Membres Du Jury**,
que nous remercions chaleureusement pour le temps consacré à évaluer notre travail.

Enfin, merci à toutes celles et ceux qui, de près ou de loin,
ont contribué à la réalisation de ce mémoire.
Votre soutien, aussi discret soit-il, a été essentiel.

Dédicace

À mes chers parents,
pour leur force tranquille et leur amour sincère, constant et sans détour.

À mes sœurs précieuses,
pour les discussions à rallonge, les silences qui disent tout,
et cette complicité qu'aucun mot ne peut vraiment décrire.

À toute ma famille et mes amis,
pour leur présence dans le chaos, leur humour qui allège tout, et leur patience face à mes "j'en peux plus".

Ce mémoire n'aurait pas le même goût sans vous.
Merci d'avoir été là — vraiment.

– *Cylia*

Dédicace

À mes chers parents,
lumière de ma vie,
ce que je suis aujourd'hui est le fruit de leurs prières, de leurs sacrifices et de leur soutien indéfectible, qui m'ont portée bien plus qu'ils ne l'imaginent.

À mon frère et à mes sœurs adorés,
pour leur amour sincère, leur énergie, et cette force qu'ils m'ont transmise sans même le savoir, dans les moments où j'en avais le plus besoin.

À toute ma famille et à mes amis,
pour chaque mot d'encouragement, chaque geste sincère, et leur bienveillance tout au long de ce parcours.

– *Sara*

Table des matières

Introduction générale

La recherche opérationnelle (RO) est une branche des mathématiques appliquées dédiée à la modélisation et à l'optimisation de systèmes complexes, dans le but de soutenir la prise de décision rationnelle [27]. Elle vise à allouer de façon optimale des ressources limitées à des activités concurrentes, en s'appuyant sur des outils mathématiques, algorithmiques et informatiques [17]. Née durant la Seconde Guerre mondiale pour améliorer l'efficacité des opérations militaires, la RO s'est progressivement étendue à des domaines variés tels que l'industrie, les transports, la finance, la logistique, la santé, l'éducation ou encore l'environnement. Son efficacité repose sur deux piliers fondamentaux : la modélisation mathématique des problèmes concrets et leur résolution algorithmique. De la programmation linéaire aux méthodes combinatoires, en passant par les simulations stochastiques et l'optimisation non linéaire, la RO offre un cadre rigoureux et puissant pour traiter des décisions complexes dans des environnements incertains et contraints [7].

Cette recherche de performance et d'efficacité propre à la RO, résonne aussi dans nos vies quotidiennes [2]. En effet, nous cherchons constamment à optimiser divers aspects de notre existence : notre emploi du temps, notre espace de rangement ou nos trajets. L'optimisation consiste à trouver la solution la plus efficace pour résoudre les problèmes et atteindre nos objectifs avec moins d'efforts [1], de temps ou de ressources. C'est une démarche qui nous aide à mieux organiser notre vie et à agir de manière plus stratégique.

Au-delà de ces cas concrets, l'optimisation soulève des questions complexes

sur le plan théorique, notamment lorsque les problèmes abordés ne permettent pas d'identifier facilement une solution optimale. Ce mémoire s'inscrit dans cette réflexion en analysant les défis liés à l'optimisation non convexe et les approches méthodologiques développées pour les surmonter. Il met en lumière les obstacles rencontrés et examine les solutions, tant théoriques que pratiques, mises en œuvre pour répondre de manière efficace à ces problématiques.

Le premier chapitre établit les **fondements théoriques** en introduisant de façon approfondie les concepts essentiels de l'optimisation convexe et non convexe [9, 23, 40]. Il précise les définitions clés, telles que celles des ensembles convexes, des fonctions convexes, des minimums locaux et globaux [10, 13, 16], ainsi que les conditions nécessaires et suffisantes d'optimalité (incluant les conditions de Karush-Kuhn-Tucker) [11, 31]. Le chapitre présente en détail les méthodologies classiques, tant analytiques que numériques [24, 25, 44], notamment les méthodes de Newton (et ses variantes adaptées aux cas non convexes) [3, 26] ainsi que la méthode du gradient de descente [30, 43], en expliquant leurs principes, leurs avantages et leurs limites. Une attention particulière est portée sur les difficultés spécifiques rencontrées lors de la résolution des problèmes non convexes, notamment la multiplicité des optimums locaux, la présence de points-selles et l'absence de garanties de convergence globale, ce qui souligne l'importance des outils adaptés pour traiter ces problématiques complexes.

Le deuxième chapitre s'intéresse à l'étude de **l'optimisation linéaire et non linéaire**, en analysant les techniques usuelles de résolution employées dans ces contextes [18, 28, 34]. Il présente notamment la méthode du simplexe [12], en décrivant son mécanisme itératif de pivotage, son traitement des cas dégénérés et son efficacité pratique malgré une complexité théorique défavorable. Le chapitre explore également la méthode Branch and Bound [21, 38, 45], largement utilisée en programmation linéaire en nombres entiers, en détaillant ses principes de séparation, d'évaluation et d'élagage. Une attention particulière est accordée aux particularités de l'optimisation non linéaire, où les contraintes et la fonction

objectif peuvent être non linéaires, rendant les problèmes plus complexes à résoudre. Le chapitre examine les approches classiques et les défis spécifiques de ce cadre, tels que les non-convexités, les interactions complexes entre variables, et la nécessité d'algorithmes robustes et performants.

Enfin, le troisième chapitre est consacré à l'exploration des **méthodes avancées** permettant de traiter efficacement les problèmes non convexes [14, 20, 30], qui échappent souvent aux techniques classiques. Il introduit la programmation dynamique [8, 15, 19], en expliquant la décomposition en sous-problèmes selon la récurrence de Bellman [6], ses applications typiques ainsi que ses limitations [29]. Le chapitre étudie également les techniques de relaxation convexe, qui consistent à approximer des problèmes non convexes par des formulations convexes plus tractables, permettant de calculer des bornes utiles ou d'obtenir des solutions approchées de haute qualité [37]. Enfin, il présente les stratégies de linéarisation, visant à transformer des expressions non linéaires en formulations linéaires exploitables par des solveurs puissants, comme ceux de la programmation en nombres entiers mixtes (MIP). Pour illustrer concrètement ces méthodes, nous avons choisi une application financière : sous le nom de Liquidation Optimale de Portefeuille (LOP)(ou d'Ecoulement de Larges Blocs dActifs (ELBA)) [33], modélisée comme un problème déterministe non convexe non linéaire. Ce problème a été choisi car il combine à la fois une structure mathématique complexe et un enjeu pratique fort, typique des situations réelles où les décisions doivent être prises sous contraintes. Il met en jeu des fonctions non linéaires, des variables mixtes et des effets d'échelle, ce qui en fait un terrain d'expérimentation pertinent pour les techniques de relaxation et de linéarisation abordées dans ce chapitre [5, 32, 41, 42].

Nous terminerons notre travail par une conclusion générale.

Chapitre 1

Optimisation convexe et optimisation non convexe

1.1 Introduction

L'optimisation convexe s'intéresse aux problèmes où la fonction objectif et les contraintes (si elles existent) présentent des propriétés de convexité. Ces propriétés assurent qu'un minimum local est aussi un minimum global, facilitant ainsi la résolution de tels problèmes. À l'inverse, l'optimisation non convexe concerne les problèmes où la fonction objectif ou les contraintes ne sont pas convexes. Dans ce cadre, la structure du problème devient plus complexe, et les minima locaux ne correspondent pas nécessairement à un minimum global. Cette absence de garanties globales sur l'optimalité des solutions rend la résolution de ces problèmes plus difficile.

1.2 Optimisation Convexe

1.2.1 Rappels et notions de base

Définition.1.1 (fonction différentiable) [9, 40]

Soit $K \subset \mathbb{R}^n$ un ouvert et $f : K \to \mathbb{R}^n$ une application. On dit que f est différentiable en un point $x \in K$ si toutes les dérivées partielles $\frac{\partial f}{\partial x_i}$ existent en x et s'il existe une application linéaire $L : \mathbb{R}^n \to \mathbb{R}^n$, appelée différentielle de f en x,

telle que

$$f(x+h) = f(x) + L(h) + \|h\| \cdot \epsilon(h),$$

où $\epsilon(h) \to 0$ quand $h \to 0$. En outre, f est dite de classe C^1 sur K si f est différentiable en tout point de K et si toutes ses dérivées partielles $\frac{\partial f}{\partial x_i}$ sont continues sur K.

Définition.1.2 (Différentiabilité à l'ordre 1) [9]

Si $f \in C^1(K, \mathbb{R}^n)$, alors on appelle **gradient** de f en x le vecteur de $\mathbb{R}^n$

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

Remarquons que $f(x)$ est un nombre alors que $\nabla f(x)$ est un vecteur.

Définition.1.3 (Différentiabilité à l'ordre 2) [9]

f est de classe C^2 sur K ssi $f \in C^1(K, \mathbb{R}^n)$ et $\nabla f \in C^1(K, \mathbb{R})$. On note alors H_f ou $\nabla^2 f$ la matrice jacobienne de ∇f ; elle est appelée **hessienne** de f. Cette matrice carrée de taille n est donnée par

$$[H_f(x)]_{ij} = \frac{\partial}{\partial x_j}\frac{\partial f}{\partial x_i}(x) = \frac{\partial^2 f}{\partial x_j \partial x_i}(x).$$

Proposition. [22]

a) Matrice semi-définie positive (SDP)

La matrice $H_f(x)$ est dite **semi-définie positive** si et seulement si :

$$\forall x \in \mathbb{R}^n, \quad x^T H_f(x) x \geq 0.$$

Cela équivaut à dire que **tous les vecteurs propres** de la matrice $H_f(x)$ sont **positifs ou nuls**.

Exemple

Considérons la matrice suivante :

$$A = \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}$$

Pour tout vecteur $x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \in \mathbb{R}^2$, on a :

$$x^T A x = (x_1 - x_2)^2 \geq 0$$

Cette expression est toujours positive ou nulle, donc la matrice est **semi-définie positive**.

En outre, calculons ses valeurs propres :

$$\det(A - \lambda I) = \begin{vmatrix} 1-\lambda & -1 \\ -1 & 1-\lambda \end{vmatrix} = (1-\lambda)^2 - 1 = \lambda^2 - 2\lambda$$

Les racines sont :

$$\lambda = 0 \quad \text{ou} \quad \lambda = 2$$

Les valeurs propres sont donc positives ou nulles, ce qui confirme que la matrice est **semi-définie positive**.

b) Matrice définie positive (DP)

La matrice $H_f(x)$ est dite **définie positive** si et seulement si :

$$\forall x \in \mathbb{R}^n \setminus \{0_{\mathbb{R}^n}\}, \quad x^T H_f(x) x > 0.$$

Cela équivaut à dire que **tous les vecteurs propres** de la matrice $H_f(x)$ sont **strictement positifs**.

Exemple

Considérons la matrice :

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

Pour tout vecteur non nul $x \in \mathbb{R}^2$, on a :

$$x^T A x = 2x_1^2 + 2x_1 x_2 + 2x_2^2 > 0$$

Cette expression est strictement positive pour tout $x \neq 0$, donc la matrice est **définie positive**.

Calculons ses valeurs propres :

$$\det(A - \lambda I) = \begin{vmatrix} 2-\lambda & 1 \\ 1 & 2-\lambda \end{vmatrix} = (2-\lambda)^2 - 1 = \lambda^2 - 4\lambda + 3$$

Les racines sont :

$$\lambda = 1 \quad \text{ou} \quad \lambda = 3$$

Les deux valeurs propres sont strictement positives, ce qui confirme que la matrice est **définie positive**.

c) Matrice définie négative (DN)

La matrice $H_f(x)$ est dite **définie négative** si et seulement si :

$$\forall x \in \mathbb{R}^n \setminus \{0_{\mathbb{R}^n}\}, \quad x^T H_f(x) x < 0.$$

Cela équivaut à dire que **tous les vecteurs propres** de la matrice $H_f(x)$ sont **strictement négatifs**.

Exemple

Considérons la matrice :

$$B = \begin{pmatrix} -3 & 0 \\ 0 & -1 \end{pmatrix}$$

Pour tout vecteur non nul $x \in \mathbb{R}^2$, on a :

$$x^T B x = -3x_1^2 - x_2^2 < 0$$

L'expression est strictement négative pour tout $x \neq 0$, donc la matrice est **définie négative**.

Les valeurs propres sont directement :

$$\lambda_1 = -3 \quad \text{et} \quad \lambda_2 = -1$$

Elles sont strictement négatives, ce qui confirme que la matrice est **définie négative**.

Définition.1.4 (Dérivée directionnelle) [2]

On appelle dérivée directionnelle de f dans la direction d au point x, notée $\delta f(x,d)$, la limite (éventuellement $\pm\infty$) du rapport :

$$\frac{f(x+hd)-f(x)}{h}$$

lorsque h tend vers 0. Autrement dit :

$$\delta f(x,d)=\lim_{h\to 0}\frac{f(x+hd)-f(x)}{h}=\nabla^{\top}f(x)\,d.$$

Définition.1.5 (Direction de descents) [9]

Soit $f:\mathbb{R}^n\to\mathbb{R}$ une fonction différentiable. Soient $x,d\in\mathbb{R}^n$. La direction d est une direction de descente en x si

$$d^{\top}\nabla f(x)<0.$$

1.2.2 Problème d'optimisation convexe

1.2.2.1 Définition [36]

Un problème d'optimisation convexe est un problème de minimisation ou maximisation de la forme :

$$\min_{x\in\mathbb{R}^n}(\max_{x\in\mathbb{R}^n})f(x)$$

où :

- $f:\mathbb{R}^n\to\mathbb{R}$ est une **fonction convexe**.
- Les contraintes (si elles existent) sont définies par **un ensemble convexe** $C\subseteq\mathbb{R}^n$, c'est-à-dire :

$$x\in C.$$

1.2.2.2 Ensemble convexe

1. **Définition [16]**
   Un ensemble $C\subset\mathbb{R}$ est dit **convexe** si, pour tout couple de points (x,y) appartenant à C et pour tout scalaire λ dans l'intervalle $[0,1]$, le point $\lambda x+(1-\lambda)y$ appartient également à C.

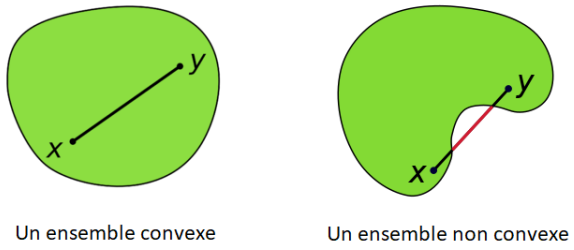


FIGURE 1.1 – La différence entre un ensemble convexe et non convexe

Exemple

Soit l'ensemble

$$C = \{(x, y) \in \mathbb{R}^2 \mid x \geq 0, y \geq 0\},$$

c'est-à-dire le premier quadrant du plan.

Pour tous $\lambda \in [0, 1]$ et pour tous points (x_1, y_1), $(x_2, y_2) \in C$, on a :

$$\lambda(x_1, y_1) + (1 - \lambda)(x_2, y_2) = (\lambda x_1 + (1 - \lambda)x_2,\ \lambda y_1 + (1 - \lambda)y_2).$$

Comme $\lambda x_1 + (1 - \lambda)x_2 \geq 0$ et $\lambda y_1 + (1 - \lambda)y_2 \geq 0$, alors

$$\lambda(x_1, y_1) + (1 - \lambda)(x_2, y_2) \in C.$$

Donc, C est convexe.

2. **Propriétés des ensembles convexes [13, 16]**

1. La définition d'ensemble convexe peut s'interpréter en disant que le segment reliant x et y doit être dans C.

2. Soit $x_1, x_2, \ldots, x_k \in \mathbb{R}^n$ et t_j telle que $t_j \geq 0$ et $\sum_{j=1}^{k} t_j = 1$. Toute expression de la forme

$$\sum_{j=1}^{k} t_j x_j.$$

S'appelle combinaison convexe des points x_j ou barycentre.

3. Si C_1 et C_2 sont deux ensembles convexes de $\mathbb{R}^n$, alors $K = C_1 \cap C_2$ est convexe et

$$K = \{x \mid x = x_1 + x_2,\ x_1 \in C_1 \text{ et } x_2 \in C_2\},$$

est convexe.

1.2.2.3 Fonction convexe

1. **Définition : [13]**

   Une fonction $f : C \to \mathbb{R}$, définie sur un ensemble convexe C, est dite **convexe** si, pour tout couple de points $(x, y) \in C^2$ et pour tout $\lambda \in [0, 1]$, elle vérifie l'inégalité suivante :

   $$f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda)f(y).$$

   On dit que f est **strictement convexe** si, pour tout $(x, y) \in C^2$ avec $x \neq y$ et pour tout $\lambda \in (0, 1)$, l'inégalité est stricte :

   $$f(\lambda x + (1 - \lambda)y) < \lambda f(x) + (1 - \lambda)f(y).$$

   Exemple

   Considérons la fonction $f : \mathbb{R} \to \mathbb{R}$ définie par :

   $$f(x) = x^2$$

   L'ensemble $\mathbb{R}$ est un ensemble convexe.

   Prenons deux réels quelconques $x, y \in \mathbb{R}$, et un scalaire $\lambda \in [0, 1]$. Vérifions l'inégalité de convexité :

   On a d'une part :

   $$f(\lambda x + (1 - \lambda)y) = (\lambda x + (1 - \lambda)y)^2$$

   $$= \lambda^2 x^2 + 2\lambda(1 - \lambda)xy + (1 - \lambda)^2 y^2 \quad (1)$$

   D'autre part :

   $$\lambda f(x) + (1 - \lambda)f(y) = \lambda x^2 + (1 - \lambda)y^2 \quad (2)$$

   Par comparaison de (1) et (2), on aura :

   $$f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda)f(y)$$

Et l'inégalité est **stricte** si $x \neq y$, car :

$$f(\lambda x + (1-\lambda)y) < \lambda f(x) + (1-\lambda)f(y) \quad \text{pour tout } x \neq y \text{ et } \lambda \in (0,1)$$

La fonction $f(x) = x^2$ est donc strictement convexe sur $\mathbb{R}$.

2. **Propriétés des fonctions convexes : [10, 13]**

1. Soit une fonction réelle définie sur un ensemble convexe $C \subset \mathbb{R}^n$. Alors f est convexe si et seulement si son épigraphe

$$epi(f) = \{(x,r) : x \in C, r \geq f(x)\} \subset \mathbb{R}^n \times \mathbb{R}$$

est un ensemble convexe.

2. Soit f une fonction réelle définie sur un ensemble convexe $C \subset \mathbb{R}^n$. Alors f est convexe si et seulement si :

$$\frac{f\left(\sum_{k=1}^{n} \lambda_i x_i\right)}{n} \leq \sum_{k=1}^{n} \lambda_i f(x_i)$$

où $x_i \in C, i = 1, 2, \ldots, n, \lambda_i \geq 0, \sum_{i=1}^{n} \lambda_i = 1$.

3. Soit f une fonction réelle de classe C^1, définie sur un ensemble convexe $C \subset \mathbb{R}^n$. Alors f est convexe si et seulement si :

$$f(y) - f(x) \geq \nabla^t f(x)(y-x), \quad \forall x, y \in C.$$

4. Soit f une fonction réelle de classe C^2, définie sur un ensemble convexe $C \subset \mathbb{R}^n$. Alors f est convexe si et seulement si :

$$(y-x)^t H(x)(y-x) \geq 0, \quad \forall x, y \in C.$$

5. Soit $f : \mathbb{R}^n \to \mathbb{R}$ une fonction convexe. Si f est **coercive**, c'est-à-dire si elle vérifie :

$$\lim_{\|x\| \to +\infty} f(x) = +\infty,$$

alors f admet un **minimum global** sur $\mathbb{R}^n$.

1.2.2.4 Définition [27]

Soit $S \subset \mathbb{R}^n$ convexe. Une fonction $\overline{f} : S \to \mathbb{R}$ est une **surestimation concave** de f sur S si :

- $\overline{f}$ est concave.
- $\forall x \in S,\ f(x) \leq \overline{f}(x)$.

1.2.2.5 Définition [27]

Une fonction $\underline{f} : S \to \mathbb{R}$ est une **sous-estimation convexe** de f sur S si :

- $\underline{f}$ est convexe ;
- $\forall x \in S,\ f(x) \geq \underline{f}(x)$.

On note $\mathcal{U}(f, S)$ l'ensemble des surestimations concaves de f sur S, et $\mathcal{L}(f, S)$ l'ensemble des sous-estimations convexes. Leurs bornes respectives sont appelées *enveloppes* :

1.2.2.6 Définition [27]

Soit f une fonction définie sur un ensemble S.

- L'**enveloppe concave** de f sur S est la fonction :

$$E_f^S(x) = \inf_{\overline{f} \in \mathcal{U}(f,S)} \overline{f}(x)$$

où $\mathcal{U}(f, S)$ est l'ensemble des fonctions concaves majorant f sur S.

- L'**enveloppe convexe** de f sur S est la fonction :

$$e_f^S(x) = \sup_{\underline{f} \in \mathcal{L}(f,S)} \underline{f}(x)$$

où $\mathcal{L}(f, S)$ est l'ensemble des fonctions convexes minorant f sur S.

Ces enveloppes sont, en général, difficiles à caractériser explicitement. Toutefois, si f est elle-même convexe (resp. concave), alors elle coïncide avec sa enveloppe convexe (resp. concave).

1.3 Problème d'optimisation sans contraintes

1.3.1 Définitions

Considérons le problème d'optimisation sans contraintes (P) suivant :

$$\min_{x\in\mathbb{R}^n} f(x)$$

où $x \in \mathbb{R}^n$ est un vecteur de dimension n et $f(x)$ est la fonction objectif à minimiser.

Définition.1.1 [11, 31]

On dit que x^* est un **minimum local** de f s'il existe un voisinage V de x^* (c'est-à-dire $V \in \mathcal{V}(x^*)$) tel que :

$$\forall x \in V,\ f(x) \geq f(x^*).$$

où $\mathcal{V}(x^*)$ désigne l'ensemble des voisinages de x^*.

Définition.1.2 [11, 31]

On dit que x^* est un **minimum global** de f si :

$$\forall x \in \mathbb{R}^n,\ f(x) \geq f(x^*).$$

Définition.1.3 [11, 31]

Une suite $(x_n)_{n\in\mathbb{N}}$ de points de $\mathbb{R}^n$ est dite **suite minimisante** si :

$$\lim_{n\to+\infty} f(x_n) = \inf_{x\in\mathbb{R}^n} f(x).$$

Définition.1.4 [11, 31]

x_0 est un **point stationnaire** si et seulement si :

$$\nabla f(x_0) = 0.$$

Si x_0 n'est ni un **minimum** ni un **maximum**, alors est un **point singulier** (ou **point col**).

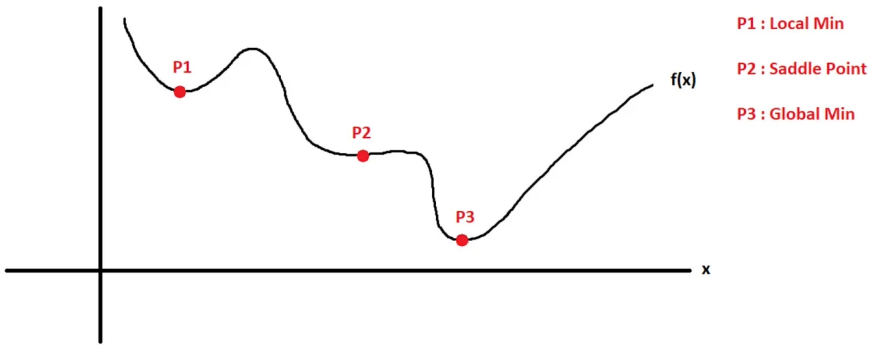


FIGURE 1.2 – Points stationnaires d'une fonction

Définition.1.5 [11, 31]

Une fonction f est dite **propre** si elle ne prend jamais la valeur $-\infty$ et n'est pas identiquement égale à $+\infty$.

1.3.2 Existence et unicité de la solution du problème

Théorème. ***(Existence) [11, 16]***
Soit $f : \mathbb{R}^n \to \mathbb{R}$ une fonction propre, continue et coercive. Alors, il existe un point $x^* \in \mathbb{R}^n$ qui réalise le minimum de f sur $\mathbb{R}^n$. Autrement dit, il existe $x^* \in \mathbb{R}^n$ tel que :

$$f(x^*) \leq f(x), \quad \forall x \in \mathbb{R}^n.$$

Théorème. ***(Unicité) [11, 16]***
Soit $f : \mathbb{R}^n \to \mathbb{R}$ une fonction strictement convexe. Alors, il existe au plus un point $x^* \in \mathbb{R}^n$ tel que :

$$f(x^*) = \min_{x \in \mathbb{R}^n} f(x).$$

Théorème. ***(Existence et unicité) [11, 16]***
Soit $f : \mathbb{R}^n \to \mathbb{R}$ une fonction vérifiant les propriétés suivantes :

1. *f est continue sur $\mathbb{R}^n$.*
2. *f est coercive.*
3. *f est strictement convexe.*

Alors, il existe un ***unique*** *point $x^* \in \mathbb{R}^n$ tel que :*

$$f(x^*) = \min_{x \in \mathbb{R}^n} f(x).$$

Théorème. ***[11, 16]***
Soit $f : \mathbb{R}^n \to \mathbb{R}$ une fonction de classe C^1. Supposons qu'il existe $\alpha > 0$ tel que, pour tout $x, y \in \mathbb{R}^n$, on ait :

$$\langle \nabla f(x) - \nabla f(y), x - y \rangle \geq \alpha \|x - y\|^2.$$

Alors, f est ***strictement convexe*** *et* ***coercive****. En particulier, le problème d'optimisation (P) :*

$$\min_{x \in \mathbb{R}^n} f(x)$$

admet une ***solution unique****.*

1.3.3 Conditions d'optimalité

1.3.3.1 Conditions nécessaires :

Théorème. ***[4, 7, 11]***
Soit x^* un minimum local de $f : \mathbb{R}^n \to \mathbb{R}$, où f est une fonction différentiable. Alors, on a les conditions nécessaires suivantes :

1. *Condition nécessaire du premier ordre :*

$$\nabla f(x^*) = 0.$$

2. *Condition nécessaire du second ordre :*

$$\nabla^2 f(x^*) \text{ est une matrice semi-définie positive.}$$

1.3.3.2 Conditions suffisantes :

Théorème. *[4, 7]*
Soit $f \in C^2(\mathbb{R}^n)$ (c'est-à-dire f est deux fois continûment différentiable), et soit $x^* \in \mathbb{R}^n$ tels que :

$$\nabla f(x^*) = 0$$

et

$$\nabla^2 f(x^*) \text{ est définie positive.}$$

*Alors, x^* est un **minimum local** de f.*

Preuve : [7, 11]
Soit d une direction arbitraire, on a :

$$f(x^* + td) = f(x^*) + t\, d^\top \nabla f(x^*) + \frac{1}{2} t^2\, d^\top \nabla^2 f(x^*) d + o(\|td\|^2).$$

Donc

$$f(x^* + td) - f(x^*) = t^2 \left(\frac{1}{2} d^\top \nabla^2 f(x^*) d + \frac{o(\|td\|^2)}{t^2} \right).$$

Pour t suffisamment petit, on aura :

$$t^2 \left(\frac{1}{2} d^\top \nabla^2 f(x^*) d + \frac{o(\|td\|^2)}{t^2} \right) > 0.$$

D'où

$$f(x^* + td) - f(x^*) > 0.$$

Ainsi, x^* est un minimum local de f.

1.3.3.3 Conditions nécessaires et suffisantes :

Théorème. *[7]*
*Soit $f : \mathbb{R}^n \to \mathbb{R}$ une fonction convexe et différentiable. Alors, la condition nécessaire et suffisante pour que $x^* \in \mathbb{R}^n$ soit un **minimum global** de f sur $\mathbb{R}^n$ est que :*

$$\nabla f(x^*) = 0.$$

Preuve : [11]

Soit $f : \mathbb{R}^n \to \mathbb{R}$ une fonction **convexe**. Alors, on a l'inégalité suivante pour tout $x \in \mathbb{R}^n$:

$$f(x) \geq f(x^*) + (x - x^*)^\top \nabla f(x^*).$$

Si x^* est un **minimum global** de f sur $\mathbb{R}^n$, alors :

$$f(x) \geq f(x^*), \forall x \in \mathbb{R}^n.$$

Cela implique :

$$(x - x^*)^\top \nabla f(x^*) = 0, \forall x \in \mathbb{R}^n.$$

En particulier, on en déduit que :

$$\nabla f(x^*) = 0.$$

Réciproquement, si $\nabla f(x^*) = 0$, alors :

$$f(x) \geq f(x^*), \forall x \in \mathbb{R}^n,$$

ce qui montre que x^* est un **minimum global** de f.

1.3.4 Exemple

Recherche des extrema de $f(x, y) = x^3 + y^3 + 3xy$ [11, 31]

Calcul du gradient

Le gradient est donné par :

$$\nabla f(x, y) = \begin{pmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{pmatrix} = \begin{pmatrix} 3x^2 + 3y \\ 3y^2 + 3x \end{pmatrix}.$$

Recherche des points critiques

Les points critiques sont obtenus en résolvant :

$$\begin{cases} 3x^2 + 3y = 0. \\ 3y^2 + 3x = 0. \end{cases}$$

En simplifiant :

$$\begin{cases} x^2 + y = 0. \\ y^2 + x = 0. \end{cases}$$

En substituant $y = -x^2$ dans la seconde équation :

$$(-x^2)^2 + x = 0 \implies x^4 + x = 0.$$

Factorisation :

$$x(x^3 + 1) = 0.$$

Les solutions sont :

$$x = 0 \quad \text{ou} \quad x = -1.$$

En remplaçant dans $y = -x^2$, on obtient : Pour $x = 0$, $y = 0$, donc $(0, 0)$. et Pour $x = -1$, $y = -1$, donc $(-1, -1)$.

Analyse des points critiques

La matrice hessienne est :

$$\nabla^2 f(x, y) = \begin{pmatrix} 6x & 3 \\ 3 & 6y \end{pmatrix}.$$

Pour le point critique $(0, 0)$

$$\nabla^2 f(0, 0) = \begin{pmatrix} 0 & 3 \\ 3 & 0 \end{pmatrix}.$$

Les valeurs propres sont les solutions de :

$$\det \begin{pmatrix} -\lambda & 3 \\ 3 & -\lambda \end{pmatrix} = 0.$$

$$\lambda^2 - 9 = 0 \Rightarrow \lambda = 3 \quad \text{ou} \quad \lambda = -3.$$

Puisqu'elles sont de signes opposés, $(0, 0)$ est un **point selle**.

Pour le point critique $(-1,-1)$

$$\nabla^2 f(-1,-1) = \begin{pmatrix} -6 & 3 \\ 3 & -6 \end{pmatrix}.$$

Les valeurs propres sont les solutions de :

$$\det \begin{pmatrix} -6-\lambda & 3 \\ 3 & -6-\lambda \end{pmatrix} = 0.$$

$$\lambda^2 + 12\lambda + 27 = 0.$$

Les racines sont :

$$\lambda = -3 \quad \text{ou} \quad \lambda = -9.$$

Puisqu'elles sont toutes négatives, $(-1,-1)$ est un **maximum local**.

Valeur de la fonction aux points critiques

$$f(0,0) = 0, \quad f(-1,-1) = 1.$$

1.4 Problème d'optimisation avec contraintes

Le problème d'optimisation avec contraintes consiste à chercher une solution admissible x minimisant une fonction objective $f(x)$, soumise à des contraintes de type égalité, inégalité ou mixte [5, 31].

1.4.1 Optimisation avec contraintes d'égalités

Le problème d'optimisation avec contraintes de type égalité s'écrit sous la forme :

$$\min_{x \in \mathbb{R}^n} f(x)$$

soumise à : $h_i(x) = 0$ pour $i = 1, 2, \ldots, m$.

On suppose que les fonctions $h_i(x)$ sont continues et continûment dérivables.

1.4.1.1 Formulation du Lagrangien

Pour résoudre ce problème, on utilise la méthode des multiplicateurs de Lagrange. On forme la fonction de Lagrange (Lagrangien) :

$$\mathcal{L}(x, \lambda) = f(x) + \sum_{i=1}^{m} \lambda_i h_i(x)$$

où $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_m)$ sont les multiplicateurs de Lagrange.

1.4.1.2 Recherche des points critiques

On cherche les points critiques du Lagrangien en annulant son gradient :

$$\nabla_{x,\lambda}\mathcal{L}(x, \lambda) = \begin{pmatrix} \nabla_x \mathcal{L}(x, \lambda) \\ \nabla_\lambda \mathcal{L}(x, \lambda) \end{pmatrix} = 0$$

avec :

$$\nabla_x \mathcal{L}(x, \lambda) = \begin{pmatrix} \frac{\partial \mathcal{L}(x,\lambda)}{\partial x_1} \\ \vdots \\ \frac{\partial \mathcal{L}(x,\lambda)}{\partial x_n} \end{pmatrix} = 0$$

et

$$\nabla_\lambda \mathcal{L}(x, \lambda) = \begin{pmatrix} \frac{\partial \mathcal{L}(x,\lambda)}{\partial \lambda_1} \\ \vdots \\ \frac{\partial \mathcal{L}(x,\lambda)}{\partial \lambda_m} \end{pmatrix} = 0.$$

1.4.1.3 Condition de deuxième ordre

Pour étudier la nature des points critiques, on utilise la condition de deuxième ordre. Cela revient à analyser la matrice $Q(w)$ définie par :

$$Q(w) = \begin{pmatrix} \nabla_x^2 \mathcal{L}(x, \lambda) & \nabla_x h(x)^T \\ \nabla_x h(x) & 0 \end{pmatrix}$$

où :

- $\nabla_x^2 \mathcal{L}(x, \lambda)$ est la matrice hessienne du Lagrangien par rapport à x.
- $\nabla_x h(x)$ est la matrice jacobienne des contraintes :

$$\nabla_x h(x) = \begin{pmatrix} \frac{\partial h_1(x)}{\partial x_1} & \cdots & \frac{\partial h_1(x)}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial h_m(x)}{\partial x_1} & \cdots & \frac{\partial h_m(x)}{\partial x_n} \end{pmatrix}$$

1.4.1.4 Détermination du minimum (ou maximum) global

Si la fonction admet plusieurs minimums (ou maximums) locaux, il est nécessaire d'évaluer la fonction $f(x)$ pour chaque point critique afin de déterminer le minimum (ou maximum) global.

1.4.2 Exemple

Minimiser la fonction $f(x) = -x_1x_2$ soumise à la contrainte
$g(x) = 2x_1 + 2x_2 - 8 = 0$ [35].

Calcul du Lagrangien

Le Lagrangien est donné par :

$$\mathcal{L}(x, \lambda) = -x_1x_2 + \lambda(2x_1 + 2x_2 - 8).$$

Calcul des points critiques

Les points critiques sont obtenus en résolvant les équations suivantes :

$$\frac{\partial \mathcal{L}}{\partial x_1} = -x_2 + 2\lambda = 0. \quad (1)$$

$$\frac{\partial \mathcal{L}}{\partial x_2} = -x_1 + 2\lambda = 0. \quad (2)$$

$$\frac{\partial \mathcal{L}}{\partial \lambda} = 2x_1 + 2x_2 - 8 = 0. \quad (3)$$

Analyse des équations L'équation (1) implique que :

$$x_2 = 2\lambda. \quad (4)$$

L'équation (2) implique que :

$$x_1 = 2\lambda. \quad (5)$$

En remplaçant les équations (4) et (5) dans l'équation (3), on obtient :

$$4\lambda + 4\lambda - 8 = 0 \implies 8\lambda - 8 = 0 \implies \lambda = 1.$$

Ainsi, il y a un seul point critique :

$$(x_1, x_2, \lambda) = (2, 2, 1).$$

Étude de la nature des points critiques

Pour déterminer la nature du point critique, on étudie la matrice hessienne du Lagrangien. La matrice hessienne est donnée par :

$$\nabla_x^2 \mathcal{L}(x, \lambda) = \begin{pmatrix} 0 & -1 \\ -1 & 0 \end{pmatrix}.$$

La matrice jacobienne de la contrainte est :

$$\nabla g(x) = \begin{pmatrix} 2 \\ 2 \end{pmatrix}.$$

On forme la matrice étendue associée aux conditions de second ordre :

$$Q = \begin{pmatrix} \nabla_x^2 \mathcal{L}(x, \lambda) & \nabla g(x) \\ (\nabla g(x))^\top & 0 \end{pmatrix} = \begin{pmatrix} 0 & -1 & 2 \\ -1 & 0 & 2 \\ 2 & 2 & 0 \end{pmatrix}.$$

Le déterminant de cette matrice est :

$$|Q| = \begin{vmatrix} 0 & -1 & 2 \\ -1 & 0 & 2 \\ 2 & 2 & 0 \end{vmatrix} = 8.$$

Puisque le déterminant est strictement positif et la matrice est symétrique, cela indique que le point critique est un minimum local. De plus, comme il est unique, il s'agit ici d'un minimum global.

1.4.3 Optimisation avec contraintes d'inégalités

Le problème d'optimisation avec contraintes de type inégalité s'écrit sous la forme :

$$\min_{x \in \mathbb{R}^n} f(x)$$

soumise à :

$$h_i(x) \leq 0 \quad \text{pour } i = 1, 2, \ldots, m.$$

Les contraintes d'inégalité sont de type inférieur ou égal ($\leq$). Si des contraintes de type supérieur ou égal ($\geq$) sont présentes, il suffit de les exprimer par leur opposé pour se ramener au cas étudié.

Pour résoudre ce problème, on utilise la méthode de **Karush-Kuhn-Tucker (KKT)**. Cette méthode utilise des paramètres appelés **multiplicateurs de Karush-Kuhn-Tucker** (notés μ_i) pour construire une fonction Lagrangienne.

Le Lagrangien s'écrit :

$$\mathcal{L}(x, \mu) = f(x) + \sum_{i=1}^{m} \mu_i h_i(x)$$

où $\mu = (\mu_1, \mu_2, \ldots, \mu_m)$ sont les multiplicateurs de Karush-Kuhn-Tucker.

D'après Karush-Kuhn-Tucker, si en un point x, alors $f(x)$ admet un minimum local dans un domaine admissible D défini par des relations de contraintes inégalités $h_i(x) \leq 0$, il existe alors des paramètres $\mu_i \geq 0$ tels que la fonction de Lagrange vérifie les conditions suivantes :

$$\begin{cases} \nabla_x \mathcal{L}(x, \mu) = 0 \\ \nabla_\mu \mathcal{L}(x, \mu) \leq 0 \\ \mu_i h_i(x) = 0 \\ h_i(x) \leq 0 \\ \mu_i \geq 0 \end{cases}$$

Ces conditions sont appelées **conditions d'optimalité globale**.

Lorsque le sens des inégalités des contraintes change, le signe des μ_i change également.

1.4.4 Exemple

Minimiser la fonction $f(x) = 4x_1^2 + 5x_2^2$ soumise à la contrainte $h(x) = x_1 - 1 \leq 0$ [35].

Calcul du Lagrangien

Le Lagrangien s'écrit :

$$\mathcal{L}(x, \mu) = 4x_1^2 + 5x_2^2 + \mu(x_1 - 1).$$

Conditions de Karush-Kuhn-Tucker (KKT)

Les conditions de KKT sont :

$$\frac{\partial \mathcal{L}}{\partial x_1} = 8x_1 + \mu = 0. \quad (1)$$

$$\frac{\partial \mathcal{L}}{\partial x_2} = 10x_2 = 0. \quad (2)$$

$$\frac{\partial \mathcal{L}}{\partial \mu} = x_1 - 1 \leq 0. \quad (3)$$

$$\mu\, h(x) = \mu(x_1 - 1) = 0. \quad (4)$$

$$\mu \geq 0. \quad (5)$$

Analyse des conditions KKT

À partir de l'équation (4), on a deux cas à considérer :

$$\mu = 0 \quad \text{ou} \quad x_1 - 1 = 0.$$

Cas 1 : Si $\mu = 0$ alors :
L'équation (1) donne : $8x_1 = 0 \implies x_1 = 0$.
L'équation (2) donne : $10x_2 = 0 \implies x_2 = 0$.
Donc le point $(0, 0)$ vérifie toutes les conditions de KKT.

Ainsi, $(0, 0)$ est un minimum local.

Cas 2 : Si $x_1 = 1$ alors :
L'équation (1) donne : $8(1) + \mu = 0 \implies \mu = -8$.
L'équation (2) donne : $10x_2 = 0 \implies x_2 = 0$.
Cette solution est rejetée car la condition (5) ($\mu \geq 0$) n'est pas vérifiée.
Donc le point $(0, 0)$ est la seule solution du problème. Par conséquent, $(0, 0)$ est un minimum global.

1.4.5 Optimisation avec contraintes mixtes

Le problème d'optimisation avec contraintes mixtes s'écrit sous la forme :

$$\min_{x \in \mathbb{R}^n} f(x)$$

soumise à :

$$\begin{cases} g_j(x) = 0 & \text{pour } j = 1, 2, \ldots, p \\ h_i(x) \leq 0 & \text{pour } i = 1, 2, \ldots, m \end{cases}$$

Dans le cas où il y a simultanément des contraintes de type égalité et de type inégalité, on introduit pour chaque contrainte un paramètre qui lui correspond. Généralement, on appelle :

- **Multiplicateurs de Lagrange**, les paramètres λ_j relatifs aux contraintes d'égalité.
- **Paramètres de Karush-Kuhn-Tucker**, les paramètres μ_i relatifs aux contraintes d'inégalité.

Pour résoudre ce problème, on déclare premièrement la fonction de Lagrange suivante :

$$\mathcal{L}(x, \mu, \lambda) = f(x) + \sum_{j=1}^{p} \lambda_j g_j(x) + \sum_{i=1}^{m} \mu_i h_i(x)$$

ou, sous forme vectorielle :

$$\mathcal{L}(x, \mu, \lambda) = f(x) + \lambda^T g(x) + \mu^T h(x)$$

En faisant le changement de variable suivant $z = (x, \lambda)$, l'équation peut être réécrite sous la forme :

$$\mathcal{L}(z, \mu) = F(z) + \mu^T h(x)$$

où :

$$F(z) = f(x) + \lambda^T g(x)$$

Finalement, le problème revient à rechercher l'optimum de la fonction $F(z)$ soumise à des contraintes de type inégalité $h_i(x) \leq 0$.

Les conditions de Karush-Kuhn-Tucker s'écrivent :

$$\begin{cases} \nabla_z \mathcal{L}(z, \mu) = 0 \\ \nabla_x \mathcal{L}(z, \mu) = 0 \\ \nabla_\lambda \mathcal{L}(z, \mu) = 0 \\ \nabla_\mu \mathcal{L}(z, \mu) \leq 0 \\ \mu_i h_i(x) = 0 \\ \mu_i \geq 0 \end{cases}$$

1.4.6 Exemple

Minimiser la fonction $f(x) = x_2 + x_3$ [35] soumise aux contraintes :

$$\begin{cases} x_1 + x_2 + x_3 = 1. \\ x_1^2 + x_2^2 + x_3^2 \leq 1. \end{cases}$$

Calcul du Lagrangien

Le Lagrangien s'écrit :

$$\mathcal{L}(x, \mu, \lambda) = x_2 + x_3 + \lambda(x_1 + x_2 + x_3 - 1) + \mu(x_1^2 + x_2^2 + x_3^2 - 1).$$

Conditions de Karush-Kuhn-Tucker (KKT)

Les conditions de KKT sont :

$$\frac{\partial \mathcal{L}}{\partial x_1} = \lambda + 2\mu x_1 = 0. \quad (1)$$

$$\frac{\partial \mathcal{L}}{\partial x_2} = 1 + \lambda + 2\mu x_2 = 0. \quad (2)$$

$$\frac{\partial \mathcal{L}}{\partial x_3} = 1 + \lambda + 2\mu x_3 = 0. \quad (3)$$

$$\frac{\partial \mathcal{L}}{\partial \lambda} = x_1 + x_2 + x_3 - 1 = 0. \quad (4)$$

$$\frac{\partial \mathcal{L}}{\partial \mu} = x_1^2 + x_2^2 + x_3^2 - 1 \leq 0. \quad (5)$$

$$\mu\, h(x) = \mu(x_1^2 + x_2^2 + x_3^2 - 1) = 0. \quad (6)$$

$$\mu \geq 0. \quad (7)$$

Analyse des conditions KKT

L'équation (6) donne deux cas à considérer :

$$\mu = 0 \quad \text{ou} \quad x_1^2 + x_2^2 + x_3^2 - 1 = 0.$$

Cas 1 : Si $\mu = 0$ alors :

L'équation (1) donne : $\lambda = 0$.
L'équation (2) donne : $1 + \lambda = 0 \implies \lambda = -1$.
L'équation (3) donne : $1 + \lambda = 0 \implies \lambda = -1$.
Il y a une contradiction ($\lambda = 0$ et $\lambda = -1$), donc cette solution est rejetée.

Cas 2 : Si $x_1^2 + x_2^2 + x_3^2 - 1 = 0$ alors :
- Les équations (1), (2) et (3) donnent :

$$\begin{cases} x_1 = -\frac{\lambda}{2\mu}. \\ x_2 = -\frac{\lambda+1}{2\mu}. \\ x_3 = -\frac{\lambda+1}{2\mu}. \end{cases}$$

- En remplaçant dans l'équation (4), on obtient :

$$-\frac{\lambda}{2\mu} - \frac{\lambda+1}{2\mu} - \frac{\lambda+1}{2\mu} = 1 \implies -\frac{3\lambda+2}{2\mu} = 1.$$

- En résolvant, on trouve :

$$\lambda = -\frac{2\mu+2}{3}.$$

- En remplaçant λ dans les expressions de x_1, x_2 et x_3, on obtient :

$$\begin{cases} x_1 = \frac{2\mu+2}{6\mu}. \\ x_2 = \frac{2\mu-1}{6\mu}. \\ x_3 = \frac{2\mu-1}{6\mu}. \end{cases}$$

- En utilisant la contrainte $x_1^2 + x_2^2 + x_3^2 = 1$, on obtient :

$$\left(\frac{2\mu+2}{6\mu}\right)^2 + 2\left(\frac{2\mu-1}{6\mu}\right)^2 = 1.$$

- En simplifiant, on trouve :

$$\mu^2 = \frac{4}{9} \implies \mu = \pm\frac{2}{3}.$$

- La solution $\mu = -\frac{2}{3}$ est rejetée car elle ne vérifie pas la condition (7) ($\mu \geq 0$).

Solution admissible

Si $\mu = \frac{2}{3}$, alors :

$$\begin{cases} x_1 = 1. \\ x_2 = 0. \\ x_3 = 0. \\ \lambda = -1. \end{cases}$$

Cette solution est admissible car elle vérifie toutes les conditions de KKT.
Donc le point $(1, 0, 0)$ est un minimum global avec $f(1, 0, 0) = 0$.

1.5 Optimisation non convexe

1.5.1 Introduction

L'optimisation non convexe, longtemps considérée comme insoluble, a récemment gagné en attention grâce à des travaux en informatique, traitement du signal et statistiques. Ces recherches ont révélé que, bien que les problèmes non convexes soient généralement NP-difficiles, ils peuvent désormais être résolus de manière efficace en exploitant des structures supplémentaires propres à ces problèmes.

1.5.2 Problème d'optimisation non convexe

Comme on a vu pécédemment un problème d'optimisation se formule généralement comme :

$$\min_{x \in \mathbb{R}^n} f(x) \quad \text{s.c.} \quad x \in C,$$

où f est la fonction objectif et C l'ensemble des contraintes.

Le problème est dit **convexe** si f est une fonction convexe et C un ensemble convexe. En revanche, un problème d'optimisation qui viole l'une de ces conditions, c'est-à-dire qui possède une fonction objectif non convexe, un ensemble de contraintes non convexe, ou les deux, est appelé un **problème d'optimisation non convexe** [20].

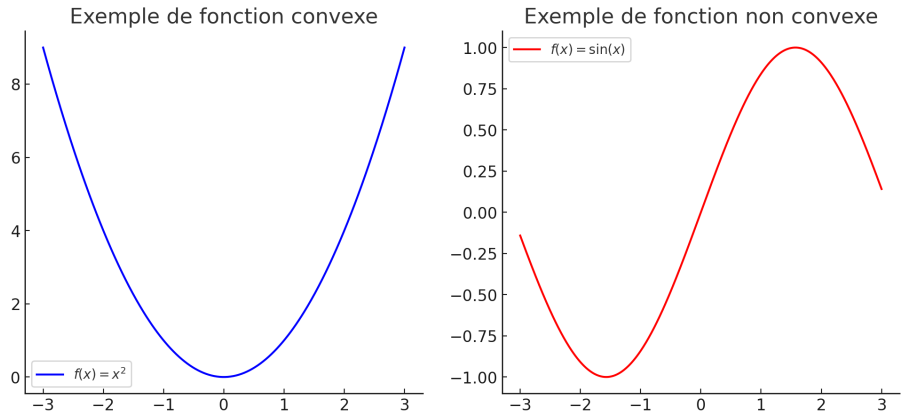


FIGURE 1.3 – Comparaison entre fonction convexe et fonction non convexe

Différence entre problème convexe et non convexe

Aspect	**Problème convexe**	**Problème non convexe**
Fonction objectif	Convexe	Non convexe
Contraintes	Convexes (inégalités), affines (égalités)	Non convexes ou non linéaires
Minima locaux	Aucun (un seul minimum global)	Multiples minima locaux
Points-selles	Absents	Présents
Complexité	Souvent résoluble en temps polynomial	Généralement NP-difficile
Garanties d'optimalité	Optimalité globale garantie	Pas de garantie d'optm globale

TABLE 1.1 – Comparaison entre les problèmes convexes et non convexes.

1.5.3 Fonction non convexe

1.5.3.1 Définitions

Définition.1 (fonction non convexe) [23]

Une fonction $f : \mathbb{R}^n \to \mathbb{R}$ est dite **non convexe**s'il existe au moins deux points $x \in \mathbb{R}^n$, $y \in \mathbb{R}^n$ et un scalaire $\lambda \in [0, 1]$ tels que :

$$f(\lambda x + (1 - \lambda)y) > \lambda f(x) + (1 - \lambda)f(y).$$

Définition.2 (Lipschitzianité) [23]

– On définit la constante de Lipschitz L comme :

$$L = \sup_{x \in \mathbb{R}^n} \|\nabla f(x)\|_2.$$

– La Lipschitzianité implique que :

$$|f(x) - f(y)| \leq L\|x - y\|_2 \quad \text{pour tout } x, y \in \mathbb{R}^n.$$

Définition.3 (Points critiques) [23]

– Un point x est un **point critique** si :

$$\nabla f(x) = 0.$$

– Les points critiques peuvent être classés en trois catégories :
  – **Minimums locaux** : $\nabla f(x) = 0$ et $\nabla^2 f(x)$ est **semi-définie positive**.
  – **Maximums locaux** : $\nabla f(x) = 0$ et $\nabla^2 f(x)$ est **semi-définie négative**.
  – **Points selle** : $\nabla f(x) = 0$, mais $\nabla^2 f(x)$ a à la fois des **valeurs propres positives et négatives**.

1.5.3.2 Propriétés de fonction non convexe [10, 13, 31]

1. **Matrice Hessienne qui change de signe** : Il existe des valeurs propres négatives, ce qui indique des zones concaves et donc la non-convexité.
2. **Continuité** : Une fonction non convexe peut être continue. Exemple :

   $$f(x) = x^3 - x$$

   Cependant, en optimisation discrète, certaines fonctions de coût sont discontinues, comme dans les problèmes d'optimisation combinatoire (ex : problème du sac à dos).
3. **Différentiabilité** : Certaines fonctions non convexes sont différentiables partout.
   Exemple :

   $$f(x) = x^3 - 3x$$

   D'autres ont des points de non-différentiabilité.
   Exemple :

   $$f(x) = -|x|$$

est également *non convexe*, et elle *n'est pas différentiable en* $x = 0$. En effet, on a :

$$f(x) = \begin{cases} -x & \text{si } x \geq 0 \\ x & \text{si } x < 0 \end{cases} \quad \Rightarrow \quad f'(x) = \begin{cases} -1 & \text{si } x > 0 \\ 1 & \text{si } x < 0 \end{cases}$$

La dérivée à droite en $x = 0$ vaut -1, tandis que celle à gauche vaut 1. La dérivée n'existe donc pas en ce point.

4. **Non-validité des conditions de KKT (Karush-Kuhn-Tucker) globalement** : Dans un problème non convexe, les conditions KKT ne garantissent pas forcément un optimum global.

5. **Minimums locaux multiples**
   Une fonction non convexe peut posséder plusieurs minima locaux, c'est-à-dire des points x^* tels que :

$$\exists \epsilon > 0, \forall z \in B(x^*, \epsilon),\ f(z) \geq f(x^*).$$

6. **Absence de garanties globales**
   Pour une fonction non convexe, les algorithmes d'optimisation ne garantissent pas la convergence vers un minimum global. Par exemple, la descente de gradient peut converger vers un minimum local ou un point de selle :

$$x_{k+1} = x_k - \eta \nabla f(x_k).$$

1.5.4 Méthodes de résolution

1.5.4.1 Méthodes analytiques : [1]

Un problème d'optimisation non convexe est souvent difficile à résoudre analytiquement, car il peut admettre plusieurs minima locaux et ne garantit pas une solution unique. Cependant, on peut parfois utiliser des méthodes analytiques selon les cas suivants :

1. Cas d'une fonction non convexe sans contraintes

1. **Trouver les points stationnaires**

- Les points stationnaires sont les candidats aux extremums. On les trouve en annulant la **dérivée première**.
- Calculer la dérivée première de la fonction $f(x)$:

$$\nabla f(x) = \frac{d}{dx} f(x) \tag{1.1}$$

- Résoudre l'équation $\nabla f(x) = 0$ pour trouver les valeurs critiques de x.

2. **Déterminer la nature des points stationnaires**
   - On utilise la **dérivée seconde** pour classifier les points stationnaires.
   - Calculer la dérivée seconde :

$$H(x) = \nabla^2 f(x) = \frac{d^2}{dx^2} f(x) \tag{1.2}$$

   - Évaluer $H(x)$ aux points critiques :
     - Si $H(x) > 0$, alors x est un **minimum local**.
     - Si $H(x) < 0$, alors x est un **maximum local**.
     - Si $H(x) = 0$, le test est **insuffisant**, il faut examiner la dérivée d'ordre supérieur ou analyser le comportement local.
3. **Vérifier la nature globale du minimum**
   - Pour déterminer si un minimum local est aussi un **minimum global**, on analyse le comportement asymptotique de $f(x)$.
   - Évaluer :

$$\lim_{x \to \pm\infty} f(x) \tag{1.3}$$

     - Si $f(x) \to -\infty$, il **n'existe pas de minimum global**.
     - Si $f(x)$ est **bornée inférieurement**, on compare les valeurs de $f(x)$ aux points critiques pour trouver le minimum global.

Exemple [1]

Considérons la fonction non convexe suivante : $f(x) = x^3 - 3x$

1. **Trouver les points stationnaires**
   - Calculons la dérivée première de $f(x)$:
   $$\nabla f(x) = \frac{d}{dx}(x^3 - 3x) = 3x^2 - 3$$
   - Résolvons $\nabla f(x) = 0$:
   $$3x^2 - 3 = 0$$
   $$3x^2 = 3$$
   $$x^2 = 1 \Rightarrow x = \pm 1$$
   - Les points critiques sont donc $x = -1$ ou $x = 1$.
2. **Déterminer la nature des points stationnaires**
   - Calculons la dérivée seconde de $f(x)$:
   $$H(x) = \nabla^2 f(x) = \frac{d^2}{dx^2}(x^3 - 3x) = 6x$$
   - Évaluons $H(x)$ aux points critiques :
     - Pour $x = 1$:
     $$H(1) = 6(1) = 6 > 0 \Rightarrow x = 1 \text{ est un minimum local.}$$
     - Pour $x = -1$:
     $$H(-1) = 6(-1) = -6 < 0 \Rightarrow x = -1 \text{ est un maximum local.}$$
3. **Vérifier la nature globale du minimum**
   - Analysons la limite aux extrémités :
   $$\lim_{x \to +\infty} f(x) = \lim_{x \to +\infty} (x^3 - 3x) = +\infty$$
   $$\lim_{x \to -\infty} f(x) = \lim_{x \to -\infty} (x^3 - 3x) = -\infty$$
   - Comme $f(x) \to -\infty$ lorsque $x \to -\infty$, il n'existe pas de minimum global.
   - Cependant, si l'on impose une contrainte sur le domaine, comme $x \in [-2, 2]$, le minimum global devient $x = 1$.

2. Cas d'une fonction non convexe avec contraintes convexes

1. **Trouver les points critiques**
   - Calculer le gradient $\nabla f(x)$ et résoudre $\nabla f(x) = 0$ pour trouver les points stationnaires.
   - Vérifier si ces points appartiennent à la région admissible définie par $g(x) \geq 0$.
2. **Vérifier la nature des points critiques**
   - Calculer la Hessienne $\nabla^2 f(x)$:
     - Si $\nabla^2 f(x)$ est définie positive, alors il s'agit d'un minimum local.
     - Si $\nabla^2 f(x)$ est définie négative, alors il s'agit d'un maximum local.
     - Si $\nabla^2 f(x)$ est indéfinie, une analyse plus approfondie est nécessaire.

   Si un point critique ne satisfait pas la contrainte, il est exclu de la recherche de l'optimum global.
3. **Étudier les points aux frontières de la contrainte**
   - Si aucun point critique ne respecte $g(x) \geq 0$:
     - Résoudre $g(x) = 0$ pour obtenir les bornes admissibles.
     - Vérifier si $f(x)$ atteint un minimum à ces bornes.
     - Si $f(x)$ est décroissante ou croissante sur la région admissible, identifier le minimum à la borne la plus appropriée.
4. **Comparer les valeurs de $f(x)$**
   - Comparer $f(x)$ aux points critiques valides et aux points frontières.
   - Identifier le minimum global admissible sous la contrainte $g(x) \geq 0$.

Exemple [1]

On considère la fonction non convexe : $f(x) = x^3 - 3x$
et la contrainte convexe : $g(x) = x + 2 \geq 0 \quad$ (soit $x \geq -2$)

1. **Vérification des points critiques**
   Nous avons déjà trouvé les points critiques $x = \pm 1$. On vérifie s'ils appartiennent à la région admissible $x \geq -2$:
   - $x = -1$: **admissible**
   - $x = 1$: **admissible**

   Tous les points critiques sont donc valides sous la contrainte.
2. **Nature des points critiques**
   Nous avons déjà déterminé la nature des points :
   - $x = -1$ est un **maximum local**.
   - $x = 1$ est un **minimum local**.
3. **Vérification aux frontières**
   La contrainte impose $x \geq -2$. On évalue donc $f(x)$ à la borne $x = -2$:
   $$f(-2) = (-2)^3 - 3(-2) = -8 + 6 = -2$$
4. **Comparaison des valeurs**
   On compare les valeurs de $f(x)$ aux points admissibles :
   - $f(-1) = 2$
   - $f(1) = -2$
   - $f(-2) = -2$

   Le minimum global sous contrainte est donc atteint en $x = 1$ et $x = -2$ avec $f(x) = -2$.

3. Cas d'une fonction non convexe avec contraintes non convexes

Dans la plupart des cas réels, lorsque la fonction et les contraintes sont non convexes, il devient complexe, voire impossible, d'obtenir une solution exacte par des méthodes analytiques. Par conséquent, on privilégie souvent des approches numériques pour approcher la solution optimale, telles que la méthode de Newton ou le gradient de descente, que nous aborderons par la suite.

Remarque : Les méthodes analytiques présentées dans les cas étudiés sont adaptées aux fonctions non convexes de complexité modérée, où il est possible de dériver explicitement les expressions des gradients et de la hessienne. Cependant, pour des fonctions plus complexes ou de grande dimension, ces approches deviennent rapidement impraticables et nécessitent l'utilisation de méthodes numériques.

Transformation des fonctions non convexes par changement de variables

Pour les fonctions non convexes, le changement de variables ne garantit pas que la non-convexité disparaisse après la transformation [24, 25, 44]. Cependant, dans certains cas, une bonne substitution peut :

– **Simplifier** la structure du problème.
– **Faciliter** la dérivation analytique.
– **Réduire** les contraintes complexes en les rendant plus maniables.

Exemple [1]

Considérons la fonction suivante : $f(x, y) = e^x + y^2 - e^{-x}$.
Cette fonction est **non convexe** car $-e^x$ est une fonction non convexe. Nous allons appliquer un **changement de variables** pour simplifier sa résolution.

Changement de variables

Posons :

$$u = e^x.$$

Ainsi, comme $e^{-x} = \frac{1}{e^x}$, nous obtenons une nouvelle fonction :

$$g(u, y) = u + y^2 - \frac{1}{u}, \text{ avec } u > 0.$$

Résolution analytique

Après avoir effectué la résolution analytique vue précédemment, nous trouvons que cette fonction possède un **minimum global** en :

$$(x^*, y^*) = (0, 0).$$

1.5.4.2 Méthodes numériques :

Méthode de Newton

Définition

La méthode de **Newton** [3, 25] est une méthode numérique itérative utilisée pour résoudre des problèmes d'optimisation, y compris les problèmes non convexes. Elle repose sur l'approximation locale de la fonction objectif par une fonction quadratique, dont le minimum est calculé à chaque itération. Cependant, dans le cas non convexe, des adaptations sont nécessaires pour garantir la convergence et la stabilité de la méthode.

Principe de la méthode de Newton

La méthode de Newton consiste à approximer la fonction objectif f au voisinage du point courant x_k par une fonction quadratique :

$$q(x) = f(x_k) + \nabla f(x_k)^T(x - x_k) + \frac{1}{2}(x - x_k)^T\nabla^2 f(x_k)(x - x_k).$$

Le point x_{k+1} est choisi comme le minimum de $q(x)$, ce qui conduit à la formule itérative :

$$x_{k+1} = x_k - [\nabla^2 f(x_k)]^{-1}\nabla f(x_k).$$

Cette méthode converge rapidement pour une fonction quadratique strictement convexe, mais peut rencontrer des difficultés dans le cas général.

Algorithme de Newton

Entrée : $x_0 \in [\alpha, \beta]$ l'approximation initiale
N le nombre maximal d'itérations à effectuer

Sortie : Une approximation de x^* ou un message d'échec.

---

1. Initialiser $k := 0$.

2. Résoudre $D^2g(x_n)s_n + Dg(x_n) = 0$ pour $s_n \in [\alpha, \beta]$.
3. $x_{n+1} := x_n + s_n$.
4. $n := n + 1$.
5. Retourner à 3.

Adaptations pour les problèmes non convexes

Pour surmonter ces difficultés, plusieurs adaptations sont proposées :

1. **Contrôle du pas de déplacement (λ_k)** :
   - On introduit un paramètre de pas λ_k dans la formule itérative :

$$x_{k+1} = x_k - \lambda_k[\nabla^2 f(x_k)]^{-1}\nabla f(x_k).$$

   - Le pas λ_k est choisi pour garantir une diminution de la fonction f. Par exemple, on utilise une recherche linéaire pour trouver λ_k tel que :

$$f(x_k + \lambda_k d_k) < f(x_k).$$

2. **Modification du hessien** :
   - Si $\nabla^2 f(x_k)$ n'est pas définie positive, on la modifie pour obtenir une matrice M_k définie positive. Une approche courante consiste à ajouter une matrice diagonale $\mu_k I$:

$$M_k = \mu_k I + \nabla^2 f(x_k),$$

     où $\mu_k > 0$ est choisi de sorte que toutes les valeurs propres de M_k soient supérieures ou égales à une constante $\gamma > 0$.
   - Cette modification garantit que M_k est définie positive et que la direction $d_k = -[M_k]^{-1}\nabla f(x_k)$ est une direction de descente :

$$\nabla f(x_k)^T d_k = -\nabla f(x_k)^T [M_k]^{-1}\nabla f(x_k) < 0.$$

3. **Convergence globale** :

– La modification du hessien et le contrôle du pas de déplacement permettent de garantir la convergence globale de la méthode. En effet, la direction d_k est toujours une direction de descente, et la recherche linéaire assure une diminution suffisante de f à chaque itération.
– Sous des conditions raisonnables (par exemple, f est bornée inférieurement et possède des dérivées continues), la méthode modifiée converge vers un point critique (un minimum local ou un point de selle).

Algorithme de Newton modifié [9]

Voici les étapes typiques pour appliquer la méthode de Newton modifiée :

1. **Initialisation** : Choisir un point de départ x_0 et une constante $\gamma > 0$.
2. **Itération** :
   - Calculer le gradient $\nabla f(x_k)$ et la hessienne $\nabla^2 f(x_k)$.
   - Si $\nabla^2 f(x_k)$ n'est pas définie positive, la modifier en

$$M_k = \mu_k I + \nabla^2 f(x_k)$$

     où μ_k est choisi pour garantir que M_k est définie positive.
   - Calculer la direction de descente $d_k = -[M_k]^{-1}\nabla f(x_k)$.
   - Utiliser une recherche linéaire pour déterminer λ_k tel que : $f(x_k + \lambda_k d_k) < f(x_k)$.
   - Mettre à jour $x_{k+1} = x_k + \lambda_k d_k$.
3. **Critère d'arrêt** : Répéter jusqu'à ce que $\|\nabla f(x_k)\|$ soit suffisamment petit ou que le nombre maximal d'itérations soit atteint.

En résumé, la méthode de Newton, adaptée pour les problèmes non convexes, est un outil puissant pour l'optimisation numérique. Elle nécessite toutefois des modifications pour garantir la convergence globale et la stabilité, en particulier dans le cas où la hessienne n'est pas définie positive.

Exemple : Méthode de Newton modifiée

Considérons la fonction suivante :

$$f(x) = \sin(x).$$

Cette fonction est non convexe et possède une infinité de minima et maxima locaux.

Gradient et Hessien

Le gradient et la hessienne de $f(x)$ sont donnés par :

$$\nabla f(x) = \cos(x),$$

$$\nabla^2 f(x) = -\sin(x).$$

Itération de la méthode de Newton modifiée

La méthode de Newton modifiée utilise la formule itérative suivante :

$$x_{k+1} = x_k - \lambda_k [M_k]^{-1} \nabla f(x_k),$$

où $M_k = \mu_k I + \nabla^2 f(x_k)$ est une modification du hessien pour garantir qu'il est défini positif.

Application numérique

Supposons que nous commençons avec $x_0 = 0.5$ et $\mu_0 = 1$.

1. **Itération 1** :
   - $x_0 = 0.5$
   - $\nabla f(x_0) = \cos(0.5) \approx 0.8776$
   - $\nabla^2 f(x_0) = -\sin(0.5) \approx -0.4794$
   - Le hessien modifié est $M_0 = \mu_0 I + \nabla^2 f(x_0) = 1 - 0.4794 = 0.5206$
   - $d_0 = -[M_0]^{-1} \nabla f(x_0) \approx -\frac{0.8776}{0.5206} \approx -1.6859$

- Recherche linéaire : $\lambda_0 = 1$ donne $f(x_0 + \lambda_0 d_0) \approx \sin(-1.1859) \approx -0.9285 < \sin(0.5) \approx 0.4794$
- $x_1 = x_0 + \lambda_0 d_0 \approx 0.5 - 1.6859 \approx -1.1859$

2. **Itération 2** :
   - $x_1 \approx -1.1859$
   - $\nabla f(x_1) = \cos(-1.1859) \approx 0.3750$
   - $\nabla^2 f(x_1) = -\sin(-1.1859) \approx 0.9270$
   - $M_1 = \mu_1 I + \nabla^2 f(x_1) = 1 + 0.9270 = 1.9270$
   - $d_1 = -[M_1]^{-1} \nabla f(x_1) \approx -\frac{0.3750}{1.9270} \approx -0.1946$
   - Recherche linéaire : $\lambda_1 = 1$ donne $f(x_1 + \lambda_1 d_1) \approx \sin(-1.3805) \approx -0.9819 < \sin(-1.1859) \approx -0.9285$
   - $x_2 = x_1 + \lambda_1 d_1 \approx -1.1859 - 0.1946 \approx -1.3805$
3. **Itération 3** :
   - $x_2 \approx -1.3805$
   - $\nabla f(x_2) = \cos(-1.3805) \approx 0.1898$
   - $\nabla^2 f(x_2) = -\sin(-1.3805) \approx 0.9819$
   - $M_2 = \mu_2 I + \nabla^2 f(x_2) = 1 + 0.9819 = 1.9819$
   - $d_2 = -[M_2]^{-1} \nabla f(x_2) \approx -\frac{0.1898}{1.9819} \approx -0.0958$
   - Recherche linéaire : $\lambda_2 = 1$ donne $f(x_2 + \lambda_2 d_2) \approx \sin(-1.4763) \approx -0.9965 < \sin(-1.3805) \approx -0.9819$
   - $x_3 = x_2 + \lambda_2 d_2 \approx -1.3805 - 0.0958 \approx -1.4763$

Après quelques itérations supplémentaires (7 itérations), la méthode converge vers $x^* \approx -1.5708$ (soit $-\pi/2$), qui est un minimum local de la fonction.

Méthode de Gradient de descente

Définition

La méthode de **gradient de descente** [43] est un algorithme d'optimisation du premier ordre qui vise à trouver un **minimum local** d'une fonction ob-

jectif $f : \mathbb{R}^n \to \mathbb{R}$. À chaque itération, elle "descend" progressivement vers les valeurs des paramètres qui minimisent la fonction.

Algorithme [30]

Soit $x \in \mathbb{R}^n$ un vecteur de paramètres à optimiser, et $f(x)$ la fonction objectif (non convexe). à l'itération t on a :

$$x_{t+1} = x_t - \eta \nabla f(x_t),$$

où :

- x_t : valeur des paramètres à l'itération t.
- η : le **taux d'apprentissage** (learning rate), un hyperparamètre qui contrôle la taille du pas de mise à jour.
- $\nabla f(x_t)$: le gradient de la fonction objectif par rapport à x à l'itération t.

Convergence de la descente de gradient

La descente de gradient converge vers un **point critique** (c'est-à-dire un point où le gradient s'annule ou devient arbitrairement petit) sous des conditions appropriées. Ces conditions incluent :

– **Lipschitz-continuité du gradient**
La fonction objectif f doit avoir un gradient **Lipschitz-continu**. Cela signifie qu'il existe une constante $L > 0$ telle que :

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\| \quad \text{pour tout } x, y \in \mathbb{R}^n.$$

Cette condition garantit que le gradient ne varie pas trop brutalement, ce qui permet de contrôler les pas de mise à jour.

– **Taux d'apprentissage adaptatif**
Le taux d'apprentissage η doit être choisi suffisamment petit pour assurer la convergence. Typiquement, on impose :

$$0 < \eta < \frac{1}{L},$$

où L est la constante de Lipschitz du gradient. Si η est trop grand, l'algorithme peut diverger ou osciller.

– **Condition de diminution suffisante**
À chaque itération, la fonction objectif doit décroître suffisamment. Formellement, on exige :

$$f(x_{t+1}) \leq f(x_t) - \frac{\eta}{2}\|\nabla f(x_t)\|^2.$$

Cela garantit que l'algorithme progresse vers un point critique.

– **Convergence vers un point critique**
Sous les conditions ci-dessus, la descente de gradient converge vers un **point critique**, c'est-à-dire un point x^* où :

$$\nabla f(x^*) = 0 \quad \text{ou} \quad \|\nabla f(x^*)\| \leq \epsilon,$$

où ϵ est une tolérance prédéfinie. Ce point critique peut être un **minimum local**, un **point selle**, ou un **maximum local**.

Exemple d'application [30]

Nous appliquons la méthode du gradient de descente pour minimiser la fonction précédente :

$$f(x) = \sin(x).$$

1. **Calcul du gradient**
   Le gradient de la fonction $f(x)$ est donné par sa dérivée première :

$$\nabla f(x) = \cos(x).$$

2. **Algorithme du gradient de descente**
   On a :

$$x_{t+1} = x_t - \eta \nabla f(x_t),$$

   avec :
   – **Point initial** : $x_0 = 0{,}5$
   – **Taux d'apprentissage** : $\eta = 0{,}1$

– **Précision** : $\epsilon = 10^{-5}$

3. **Calcul des itérations**

– **Itération 1** :

$$\nabla f(x_0) = \cos(0{,}5) \approx 0{,}877583$$
$$x_1 = 0{,}5 - 0{,}1 \times 0{,}877583 \approx 0{,}412242$$

– **Itération 2** :

$$\nabla f(x_1) = \cos(0{,}412242) \approx 0{,}916202$$
$$x_2 = 0{,}412242 - 0{,}1 \times 0{,}916202 \approx 0{,}320622$$

– **Itération 3** :

$$\nabla f(x_2) = \cos(0{,}320622) \approx 0{,}949015$$
$$x_3 = 0{,}320622 - 0{,}1 \times 0{,}949015 \approx 0{,}225720$$

4. **Condition d'arrêt**
   L'algorithme s'arrête lorsque :

$$|\nabla f(x_k)| < \epsilon$$

Après un certain nombre d'itérations (45 itérations), la méthode du gradient de descente converge vers une solution optimale locale :

$$x^* \approx 1{,}570796.$$

Remarque

Il est important de noter que ni la **méthode de Newton** ni la **descente de gradient** ne garantissent un **minimum global**. Elles convergent vers des *minimums locaux* ou des *points critiques*, mais dépendent du point de départ et ne explorent pas tout l'espace de recherche.

Conclusion

La **méthode de Newton** converge plus rapidement que la descente de gradient grâce à sa **convergence quadratique**, qui prend en compte la courbure locale de la fonction via la matrice hessienne.

Chapitre 2

Optimisation linéaire et optimisation non linéaire

2.1 Introduction

Dans ce chapitre nous présentons l'optimisation linéaire, également appelée **programmation linéaire**, s'applique aux problèmes où la fonction objectif et les contraintes sont exprimées de manière linéaire. Cette approche est couramment utilisée dans des domaines tels que la gestion, l'économie, les transports et l'industrie, avec des méthodes de résolution comme l'algorithme du **simplexe**.

En revanche, l'optimisation **non linéaire** aborde des problèmes où la fonction objectif ou les contraintes présentent des non-linéarités. Ces problèmes sont généralement plus complexes et requièrent des techniques de résolution avancées, telles que les algorithmes de gradient, les méthodes de Newton ou des approches comme la méthode **Branch and Bound**, adaptée notamment aux problèmes combinatoires ou mixtes.

2.2 Optimisation linéaire

2.2.1 Définitions

Un programme linéaire (PL) est un modèle mathématique qui peut s'écrire sous la forme suivante : Maximiser ou minimiser la fonction objectif

$$\min \text{ ou } \max z = \sum_{i=1}^{n} c_i x_i$$

Sous les contraintes

$$g = a_{i1}x_1 + \cdots + a_{in}x_n \begin{cases} \leq b_i & \text{pour } i = 1, \ldots, m \\ \geq b_i & \text{pour certaines contraintes} \end{cases}$$

Et

$$x_j \geq 0, \text{ pour } j = 1, \ldots, n$$

où z et les contraintes $g_1, \ldots, g_m$ sont tous linéaire c.à.d., elles vérifient

$$z(\alpha x + \beta y) = \alpha z(x) + \beta z(y),$$

$$g_i(\alpha x + \beta y) = \alpha g_i(x) + \beta g_i(y), \quad i = 1, \ldots, m,$$

et a_{ij}, b_i et c_i sont des constantes connues.

Les contraintes

$$x_j \geq 0, \text{ pour } j = 1, \ldots, n$$

sont appelées **contraintes de non-négativité** [12].

Définition 1 [12] : On appelle **solution** d'un problème de programmation linéaire tout vecteur x qui satisfait les contraintes.

Une solution est appelée **solution réalisable** si elle vérifie les contraintes de non-négativité. Dans le cas contraire, on dit que la solution est **non réalisable**.

Définition 2 [12] : Une solution réalisable est une *solution optimale* s'il n'existe pas d'autres solutions réalisables qui fournissent une plus grande valeur de la fonction objectif. À noter que dans un problème possédant des solutions réalisables, il se peut que la valeur optimale de la fonction objectif soit infinie. Dans ce cas, on parle de solution *optimale infinie*.

Remarque :

Dans le premier chapitre, nous avons vu que le problème d'optimisation est dit

convexe si sa fonction objectif et ses contraintes sont convexes, c'est-à-dire qu'elles vérifient :

$$f(\alpha x + \beta y) \leq \alpha f(x) + \beta f(y),$$

$$g_i(\alpha x + \beta y) \leq \alpha g_i(x) + \beta g_i(y), \quad i = 1, \ldots, m,$$

avec $\alpha = \lambda$ et $\beta = 1 - \lambda$.

Ainsi, d'après la définition d'un PL, on conclut qu'un programme linéaire est un programme d'optimisation convexe.

On peut considérer l'optimisation convexe comme une généralisation de la programmation linéaire.

2.2.2 Exemple d'un problème de programmation linéaire

La programmation linéaire est un outil mathématique puissant largement utilisé pour résoudre des problèmes d'optimisation dans divers domaines, notamment dans les entreprises industrielles et les usines de fabrication. Parmi les problèmes classiques rencontrés, on distingue :

Problème de transport

Il s'agit ici d'un problème que l'on peut résoudre par la programmation linéaire, c'est-à-dire un problème de transport [12]. Ce type de problème se définit comme suit :

Connaissant les quantités disponibles de chacune des unités de production, les quantités requises aux points de distribution et le coût de transport d'un bien d'une usine vers un point de vente, il s'agit de déterminer le plan de transport optimal, c'est-à-dire de déterminer les quantités de biens que chaque usine va envoyer vers chacun des points de vente afin que le coût de transport total soit minimum. On suppose qu'il est possible d'expédier des produits de n'importe quelle origine vers n'importe quelle destination.

L'exemple ci-dessous est le cas d'une fabrique de conserves qui expédie des caisses vers ses dépôts. Nous voulons que le plan d'expédition des caisses minimise le coût total de transport des usines aux dépôts. Pour simplifier, supposons qu'il

y a deux usines et trois dépôts. L'offre des usines et les demandes des dépôts sont les suivantes (en nombre de caisses) :

Usine 1 : 350, dépôt 1 : 200, Usine 2 : 450, dépôt 2 : 300 et dépot 3 : 50

Les coûts de transport de chaque origine vers chaque destination sont donnés dans le tableau ci-dessous (en DA par caisse) :

Usine	**Dépôt**		
	1	2	3
1	25	17	16
2	24	18	14

Ainsi, le coût pour transporter une caisse de l'usine 1 au dépôt 1 est de 25, le coût pour transporter une caisse de l'usine 2 vers le dépôt 1 est de 24, et ainsi de suite. Ce problème peut se formuler sous la forme d'un problème de programmation linéaire. Notons par c_{ij} le coût de transport d'une caisse de l'origine i vers la destination j. Nous avons donc, d'après le tableau précédent :

$$c_{11} = 25,\ c_{12} = 17,\ c_{13} = 16,\ c_{21} = 24,\ c_{22} = 18 \text{ et } c_{23} = 14$$

Soit a_i la quantité de caisses disponibles à l'origine i et b_j celle requise à la destination j. Nous pouvons représenter le problème sous forme d'un diagramme. Les lignes qui relient les usines aux dépôts peuvent être considérées comme des routes. On y indique leur coût de transport unitaire respectif. À noter que le nombre de caisses disponibles doit être supérieur ou égal au nombre de caisses requises :

$$a_1 + a_2 \geq b_1 + b_2 + b_3$$

Dans le cas contraire, le problème n'a pas de solutions réalisables. Si x_{ij} représente le nombre de caisses expédiées de l'origine i vers la destination j, le coût total de l'expédition se traduit alors par l'équation :

$$z = \sum_{i=1}^{2}\sum_{j=1}^{3} c_{ij}x_{ij} = c_{11}x_{11} + c_{12}x_{12} + c_{13}x_{13} + c_{21}x_{21} + c_{22}x_{22} + c_{23}x_{23}$$

Avec les coefficients spécifiques, cela devient

$$z = 25x_{11} + 17x_{12} + 16x_{13} + 24x_{21} + 18x_{22} + 14x_{23}$$

C'est la fonction objectif à minimiser.

Comme il est impossible d'expédier plus de caisses d'une origine donnée qu'il n'y en a de disponibles, nous sommes confrontés aux deux contraintes :

$$\sum_{j=1}^{3} x_{1j} = x_{11} + x_{12} + x_{13} \leq 350 \quad \text{(usine 1)}$$

$$\sum_{j=1}^{3} x_{2j} = x_{21} + x_{22} + x_{23} \leq 450 \quad \text{(usine 2)}$$

De plus, il faut approvisionner chacun des trois dépôts avec la quantité requise, ce qui nous donne trois nouvelles contraintes :

$$\sum_{i=1}^{2} x_{i1} = x_{11} + x_{21} = 200 \quad \text{(dépôt 1)}$$

$$\sum_{i=1}^{2} x_{i2} = x_{12} + x_{22} = 300 \quad \text{(dépôt 2)}$$

$$\sum_{i=1}^{2} x_{i3} = x_{13} + x_{23} = 50 \quad \text{(dépôt 3)}$$

Comme il n'est pas possible d'expédier des quantités négatives, nous avons encore les six contraintes de non-négativité suivantes :

$$x_{ij} \geq 0, \quad i = 1, 2 \quad \text{et} \quad j = 1, 2, 3$$

Finalement, le programme linéaire à résoudre est :

$$\min Z = 25x_{11} + 17x_{12} + 16x_{13} + 24x_{21} + 18x_{22} + 14x_{23}$$

$$\text{s.c.} \begin{cases} x_{11} + x_{12} + x_{13} \leq 350 \\ x_{21} + x_{22} + x_{23} \leq 450 \\ x_{11} + x_{21} = 200 \\ x_{12} + x_{22} = 300 \\ x_{13} + x_{23} = 50 \\ x_{ij} \geq 0, \quad i = 1, 2 \quad \text{et} \quad j = 1, 2, 3 \end{cases}$$

Comme ce problème présente plus de deux variables, nous ne pouvons pas le résoudre géométriquement.

2.2.3 Résolution du problème par la méthode Simplexe

2.2.3.1 Définition [12]

La méthode de simplexe est l'outil principal 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.

La forme matricielle permet de représenter un problème de programmation linéaire sous une forme plus concise.

$$\text{Max} \quad z = cx$$

$$\text{s.c.} \quad \begin{cases} Ax = b \\ x \geq 0 \end{cases}$$

où c est un vecteur-ligne de dimension $(1 \times n)$, x un vecteur-colonne de dimension $(n \times 1)$, A une matrice de dimension $(m \times n)$, b un vecteur-colonne de dimension $(m \times 1)$ et 0 le vecteur nul à n composantes. Un problème de programmation linéaire peut se présenter sous différentes formes. En voici la terminologie.

Forme canonique

Si la fonction objectif doit être maximisée et si toutes les contraintes sont des inéquations du type $\leq$, on dit que le programme linéaire se présente sous une

forme canonique. Matriciellement, un problème de programmation linéaire se présente sous sa forme canonique de la manière suivante :

$$\begin{array}{rl} \text{Maximiser} & z = cx \\ \text{sous contraintes} & \left\{ \begin{array}{l} Ax \leq b \\ x \geq 0 \end{array} \right. \end{array}$$

À noter que toute contrainte peut être transformée sous forme canonique.

Forme standard

Un problème de programmation linéaire se présente sous sa forme standard si toutes les contraintes sont des équations exprimées sous forme d'égalités, et la fonction objectif doit également être maximisée. Sous forme matricielle, la forme standard s'écrit :

$$\begin{array}{rl} \text{Maximiser} & z = cx \\ \text{sous contraintes} & \left\{ \begin{array}{l} Ax = b \\ x \geq 0 \end{array} \right. \end{array}$$

Transformation minimisation-maximisation

Tout problème de minimisation peut être transformé en un problème équivalent de maximisation. En effet, le problème :

$$\text{Minimiser} \quad z = cx$$

est équivalent à :

$$\text{Maximiser} \quad -z = -cx$$

Variables d'écart

La procédure que nous allons développer pour trouver la solution d'un problème de programmation linéaire s'appelle la méthode du simplexe. Cette méthode exige que le programme linéaire soit sous forme standard. Pour cette

raison, il faut transformer les inégalités rencontrées en égalités. Cette transformation se fait simplement en introduisant des variables non-négatives (qui vérifient les contraintes de non-négativité) appelées variables d'écart. Si les contraintes sont du type :

$$a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n \leq b_i$$

Nous introduisons une nouvelle variable $x_{n+i} \geq 0$ et écrivons :

$$a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n + x_{n+i} = b_i$$

De même, nous pouvons être contraints de transformer les contraintes de la forme :

$$a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n \geq b_i$$

en une égalité en soustrayant cette fois une variable d'écart $x_{n+i} \geq 0$. Nous écrivons alors :

$$a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n - x_{n+i} = b_i$$

Dans la fonction objectif, le réel c_j est souvent appelé le prix associé à la variable x_j. En assignant un prix nul à chaque variable d'écart, la transformation des contraintes en un système d'équations linéaires ne change pas la fonction objectif.

2.2.3.2 Exemple (Problème de transport précédent) [12]

Fonction objectif

Minimiser le coût total :

$$Z = 25x_{11} + 17x_{12} + 16x_{13} + 24x_{21} + 18x_{22} + 14x_{23}$$

Contraintes

– **Capacité des usines** :

$$\begin{cases} x_{11} + x_{12} + x_{13} \leq 350 \quad \text{(Usine 1)} \\ x_{21} + x_{22} + x_{23} \leq 450 \quad \text{(Usine 2)} \end{cases}$$

– **Demande des dépôts** :

$$\begin{cases} x_{11} + x_{21} = 200 \quad \text{(Dépôt 1)} \\ x_{12} + x_{22} = 300 \quad \text{(Dépôt 2)} \\ x_{13} + x_{23} = 50 \quad \text{(Dépôt 3)} \end{cases}$$

– **Non-négativité** :

$$x_{ij} \geq 0 \quad \forall i, j$$

Transformation en forme standard :

Pour appliquer la méthode du simplexe, nous introduisons des variables d'écart s_1 et s_2 pour transformer les inégalités en égalités.

– **Fonction objectif** :

$$\text{Minimiser } Z = 25x_{11} + 17x_{12} + 16x_{13} + 24x_{21} + 18x_{22} + 14x_{23}$$

– **Contraintes** :

$$\begin{cases} x_{11} + x_{12} + x_{13} + s_1 = 350 \quad \text{(Usine 1)} \\ x_{21} + x_{22} + x_{23} + s_2 = 450 \quad \text{(Usine 2)} \\ x_{11} + x_{21} = 200 \quad \text{(Dépôt 1)} \\ x_{12} + x_{22} = 300 \quad \text{(Dépôt 2)} \\ x_{13} + x_{23} = 50 \quad \text{(Dépôt 3)} \\ x_{ij}, s_1, s_2 \geq 0 \quad \forall i, j \end{cases}$$

Itérations du simplexe

Étape 1 : Transformation en maximisation

Pour utiliser la méthode du simplexe, nous transformons la minimisation en maximisation en multipliant la fonction objectif par -1 :

$$\text{Maximiser } -Z = -25x_{11} - 17x_{12} - 16x_{13} - 24x_{21} - 18x_{22} - 14x_{23}$$

Étape 2 : Tableau initial

Le tableau initial du simplexe est :

Base	x_{11}	x_{12}	x_{13}	x_{21}	x_{22}	x_{23}	s_1	s_2	Solution
s_1	1	1	1	0	0	0	1	0	350
s_2	0	0	0	1	1	1	0	1	450
$x_{11} + x_{21}$	1	0	0	1	0	0	0	0	200
$x_{12} + x_{22}$	0	1	0	0	1	0	0	0	300
$x_{13} + x_{23}$	0	0	1	0	0	1	0	0	50
$-Z$	-25	-17	-16	-24	-18	-14	0	0	0

Étape 3 : Choix de la variable entrante

Dans la ligne de $-Z$, les coefficients sont :

$$-25 \quad , \quad -17 \quad , \quad -16 \quad , \quad -24 \quad , \quad -18 \quad , \quad -14$$

Le coefficient le plus négatif est -25 pour x_{11}. Donc, x_{11} est la variable entrante.

Étape 4 : Choix de la variable sortante

Nous calculons les ratios $\frac{\text{Solution}}{\text{Coefficient de } x_{11}}$ pour chaque ligne où le coefficient de x_{11} est positif.

1. **Ligne de** s_1 :
$$\frac{350}{1} = 350$$
2. **Ligne de** $x_{11} + x_{21}$:
$$\frac{200}{1} = 200$$

Le ratio minimum est 200, donc la variable associée à cette ligne ($x_{11} + x_{21}$) sort de la base.

Étape 5 : Pivotage

1. **Ligne pivot** : La ligne de $x_{11} + x_{21}$ est choisie comme ligne pivot. Nous divisons cette ligne par 1 pour que le coefficient de x_{11} soit 1.

2. **Élimination de** x_{11} : Nous soustrayons cette ligne multipliée par 1 des autres lignes pour éliminer x_{11} des autres équations.

Étape 6 : Mise à jour du tableau

Après pivotage, le tableau devient :

Base	x_{11}	x_{12}	x_{13}	x_{21}	x_{22}	x_{23}	s_1	s_2	Solution
s_1	0	1	1	-1	0	0	1	0	150
s_2	0	0	0	1	1	1	0	1	450
x_{11}	1	0	0	1	0	0	0	0	200
$x_{12} + x_{22}$	0	1	0	0	1	0	0	0	300
$x_{13} + x_{23}$	0	0	1	0	0	1	0	0	50
$-Z$	0	-17	-16	1	-18	-14	0	0	5000

Étape 7 : Répétition des itérations

Nous répétons les étapes 4 à 7 jusqu'à ce que tous les coefficients de la ligne $-Z$ soient positifs ou nuls.

Étape 8 : Solution optimale

Après les itérations du simplexe, la solution optimale est :

$$\begin{cases} x_{11} = 200 \\ x_{12} = 150 \\ x_{13} = 0 \\ x_{21} = 0 \\ x_{22} = 150 \\ x_{23} = 50 \\ s_1 = 0 \\ s_2 = 250 \end{cases}$$

Valeur de la fonction objectif

La valeur minimale de Z est :

$$Z = 25 \times 200 + 17 \times 150 + 16 \times 0 + 24 \times 0 + 18 \times 150 + 14 \times 50 = 10950$$

Conclusion

La méthode du simplexe permet de résoudre efficacement le problème de transport en trouvant le plan d'expédition optimal qui minimise le coût total. La solution respecte toutes les contraintes de capacité et de demande.

2.3 Optimisation non linéaire

2.3.1 Définition

L'optimisation non linéaire est une branche des mathématiques appliquées qui traite des problèmes d'optimisation où la fonction objectif ou les contraintes sont **non linéaires**. Ces problèmes se divisent en deux catégories : **sans contraintes** et **avec contraintes**. Leur complexité, souvent supérieure à celle des problèmes linéaires, nécessite des outils mathématiques avancés pour leur résolution.
La forme générale d'un problème d'optimisation est la suivante :

$$\begin{aligned} &\min_{x\in\mathbb{R}^n} \quad f(x) \\ &\text{s.c.} \quad \begin{cases} g(x) \leq 0 \\ h(x) = 0 \end{cases} \end{aligned}$$

où les fonctions f, g et h sont typiquement non-linéaires .
Dans cette partie, nous présenterons les techniques permettant de résoudre des problèmes d'optimisation non linéaire, qu'ils soient **avec contraintes** ou **sans contraintes**[**18**].

2.3.2 Optimisation non linéaire sans contraintes

2.3.2.1 Définition [28, 34]

Ces problèmes d'optimisation constituent une formulation générale qui s'écrit comme suit :

$$\min_{x\in\mathbb{R}^n} f(x),$$

où f est une fonction non linéaire.
Il s'agit donc de déterminer un point $x_0 \in \mathbb{R}^n$ tel que

$$\forall x \in \mathbb{R}^n, \quad f(x_0) \leq f(x). \quad (*)$$

C'est-à-dire, un minimum global de f sur $\mathbb{R}^n$. Lorsque l'inégalité stricte

$$f(x_0) < f(x)$$

est vérifiée pour tout $x \in \mathbb{R}^n$ avec $x \neq x_0$, le minimum global est unique.

Pour beaucoup de problèmes d'optimisation sans contrainte, les principales méthodes de résolution ne permettent pas la détermination d'un minimum global, il faut alors se contenter de minima locaux, c'est-à-dire des points qui vérifient (*) seulement dans un voisinage de x_0. On verra maintenant comment de tels points peuvent être caractérisés.

2.3.2.2 Conditions d'optimalités (classiques)[28, 34]

- **Conditions nécessaires d'optimalité locale**
  Supposons que f soit suffisamment différentiable. Pour qu'un point x_0 soit un minimum local, il faut :
  a) $\nabla f(x_0) = 0$ (condition de stationnarité),
  b) La hessienne $\nabla^2 f(x_0)$ doit être une matrice semi-définie positive.
- **Conditions suffisantes d'optimalité locale**
  Sous les mêmes hypothèses de régularité, une condition suffisante pour que x_0 soit un minimum local est :
  a) $\nabla f(x_0) = 0$ (stationnarité),
  b) Le hessien $\nabla^2 f(x_0)$ est défini positif.

2.3.2.3 Existence d'une solution globale [28, 34]

Dans le cadre de l'optimisation non linéaire sans contraintes, l'existence d'une solution globale n'est pas garantie de manière générale. En effet, sans hypothèses particulières, la fonction $f : \mathbb{R}^n \to \mathbb{R}$ peut ne pas atteindre de minimum global, même si des minimums locaux existent.

Cependant, sous certaines conditions supplémentaires, l'existence d'un minimum global peut être assurée si on a :

– **Domaine compact :** Si le problème est défini sur un ensemble fermé et borné $S \subset \mathbb{R}^n$, le théorème de Weierstrass garantit que f atteint un minimum global sur S. Autrement dit, il existe un point $x^* \in S$ tel que

$$f(x^*) \leq f(x), \quad \forall x \in S.$$

– **Continuité et coercivité :** Si f est continue et coercitive, c'est-à-dire

$$\lim_{\|x\| \to \infty} f(x) = +\infty,$$

alors il existe au moins un point $x^* \in \mathbb{R}^n$ tel que

$$f(x^*) \leq f(x), \quad \forall x \in \mathbb{R}^n.$$

En outre, dans le cas où f est convexe, la stationnarité (c.à.d, $\nabla f(x_0) = 0$) constitue non seulement une condition nécessaire mais également suffisante pour qu'un point x_0 soit un minimum global. Pour une fonction strictement convexe, ce minimum global est unique.

2.3.2.4 Exemple pratique [28]

Considérons la fonction à deux variables définie par

$$f(x, y) = 2x^2 + xy + 1.5\, y^2 - 2x - 5y + 7.$$

Cette fonction peut être réécrite sous la forme quadratique standard (où la fonction quadratique est une fonction non linéaire)

$$f(x, y) = \frac{1}{2} \begin{pmatrix} x \\ y \end{pmatrix}^T \begin{pmatrix} 4 & 1 \\ 1 & 3 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} - \begin{pmatrix} 2 & 5 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + 7,$$

où

- $A = \begin{pmatrix} 4 & 1 \\ 1 & 3 \end{pmatrix}$ est une matrice symétrique définie positive.
- $b = \begin{pmatrix} 2 \\ 5 \end{pmatrix}$.
- $c = 7$.

Le **gradient** de f est donné par

$$\nabla f(x,y) = \begin{pmatrix} 4x + y - 2 \\ x + 3y - 5 \end{pmatrix}.$$

Pour trouver le minimum global, nous résolvons le système

$$\begin{cases} 4x + y - 2 = 0 \\ x + 3y - 5 = 0 \end{cases}$$

De la deuxième équation, on obtient $x = 5 - 3y$. En substituant dans la première :

$$4(5 - 3y) + y - 2 = 0 \quad \Longrightarrow \quad 20 - 12y + y - 2 = 0,$$

soit

$$18 - 11y = 0 \quad \Longrightarrow \quad y = \frac{18}{11}.$$

Ensuite,

$$x = 5 - 3\left(\frac{18}{11}\right) = 5 - \frac{54}{11} = \frac{55 - 54}{11} = \frac{1}{11}.$$

Ainsi, le minimum global est atteint en

$$x^* = \left(\frac{1}{11}, \frac{18}{11}\right).$$

2.3.3 Optimisation non linéaire avec contraintes

2.3.3.1 Conditions d'optimalité

Nous étudions les conditions d'optimalité pour un problème de minimisation sous contraintes définies par un ensemble d'inégalités et d'égalités [24, 25, 34].

1. **Définitions**

   Considérons le problème général d'optimisation non linéaire sous contraintes :

   $$\begin{aligned} &\text{minimiser } f(x) \\ &\text{sous contraintes } \begin{cases} g_i(x) \leq 0, & i = 1, \ldots, m \\ h_j(x) = 0, & j = 1, \ldots, r \end{cases} \end{aligned}$$

   où :

   - $\mathbf{x} \in \mathbb{R}^n$: vecteur de décision
   - $f : \mathbb{R}^n \to \mathbb{R}$: fonction objectif
   - $g_i, h_j : \mathbb{R}^n \to \mathbb{R}$: fonctions de contraintes de classe $\mathcal{C}^1$

Définition (Direction admissible) Soit $\mathcal{S}$ l'ensemble admissible et $x \in \mathcal{S}$. Une direction $d \in \mathbb{R}^n$ est dite **admissible** en x s'il existe $\alpha_0 > 0$ tel que :

$$x + \alpha d \in \mathcal{S}, \quad \forall \alpha \in [0, \alpha_0].$$

Condition nécessaire d'optimalité (géométrique)

Si x^* est un minimum local, alors aucune direction admissible en x^* ne peut être une **direction de descente**, c'est-à-dire :

$$\nabla f(x^*)^T d \geq 0 \quad \text{pour tout } d \text{ admissible.}$$

Cependant, cette condition est difficile à exploiter directement. Pour la rendre plus pratique, on introduit les **conditions de Lagrange** et **Karush-Kuhn-Tucker (KKT)**.

2. **Conditions de Lagrange**
   Considérons le problème avec uniquement des contraintes d'égalité :

$$\begin{aligned} &\text{minimiser } f(x) \\ &\text{sous contraintes } h_j(x) = 0, \quad j = 1, \ldots, r. \end{aligned}$$

Théorème (Condition nécessaire de Lagrange)

Si x^* est un minimum local et si les gradients $\nabla h_j(x^*)$ sont linéairement indépendants, alors il existe des multiplicateurs $\lambda_1, \ldots, \lambda_r$ tels que :

$$\nabla f(x^*) + \sum_{j=1}^{r} \lambda_j \nabla h_j(x^*) = 0.$$

Remarque : Cette condition n'est pas suffisante en général, mais elle l'est si f est convexe et les h_j sont linéaires.

3. **Conditions de Karush-Kuhn-Tucker (KKT)**
   Étendons maintenant le problème avec des contraintes d'inégalité :

$$\text{minimiser } f(x)$$
$$\text{sous contraintes } \begin{cases} g_i(x) \leq 0, & i = 1, \ldots, m \\ h_j(x) = 0, & j = 1, \ldots, r. \end{cases}$$

Théorème (Conditions KKT)

Si x^* est un minimum local et si les contraintes actives vérifient une condition de qualification (e.g., linéaire indépendance des gradients), alors il existe des multiplicateurs $\mu_1, \ldots, \mu_m \geq 0$ et $\lambda_1, \ldots, \lambda_r$ tels que :

(a) **Stationnarité :**

$$\nabla f(x^*) + \sum_{i=1}^{m} \mu_i \nabla g_i(x^*) + \sum_{j=1}^{r} \lambda_j \nabla h_j(x^*) = 0.$$

(b) **Conditions de complémentarité :**

$$\mu_i g_i(x^*) = 0, \quad \forall i.$$

(c) **Admissibilité :**

$$g_i(x^*) \leq 0, \quad h_j(x^*) = 0, \quad \forall i, j.$$

Cas suffisant : Les conditions KKT sont suffisantes si : f est convexe, les h_j sont linéaires, et les g_i sont concaves : c'est-à-dire que pour tout $x, y \in \mathbb{R}^n$ et $\lambda \in [0, 1]$, $g_i(\lambda x + (1 - \lambda)y) \geq \lambda g_i(x) + (1 - \lambda) g_i(y)$.

2.3.3.2 Exemple d'application [28]

Considérons le problème de minimisation suivant :

$$(P) \quad \begin{cases} \min\limits_{x \in \mathbb{R}^2} & f(x) = (x_1 - 1)^2 + (x_2 - 2)^2 \\ \text{s.c.} & g_1(x) = x_1 + x_2 - 1 \leq 0 \\ & g_2(x) = -x_1^2 + x_2 \leq 0 \end{cases}$$

où :

- f est non linéaire
- g_1 est linéaire (donc concave)
- g_2 est concave et non linéaire

1. **Formulation du Lagrangien**

$$\mathcal{L}(x, \mu) = f(x) + \mu_1 g_1(x) + \mu_2 g_2(x)$$

2. **Conditions d'optimalité KKT**

Pour tout minimum local x^*, $\exists \mu^* \geq 0$ tel que :

$$\begin{cases} \nabla_x \mathcal{L}(x^*, \mu^*) = 0 \\ \mu_i^* \, g_i(x^*) = 0, \quad i = 1, 2 \\ g_i(x^*) \leq 0, \quad i = 1, 2 \end{cases}$$

3. **Résolution du système KKT**

Cas 1 : $\mu_1 = \mu_2 = 0 \quad \nabla f(x) = 0 \Rightarrow x = (1, 2)$

Vérification contraintes :

$g_1(1, 2) = 2 > 0$, donc le point $x = (1, 2)$ est non admissible.

Cas 2 : $\mu_1 > 0$**,** $\mu_2 = 0$, on aura le système :

$$\begin{cases} 2(x_1 - 1) + \mu_1 = 0 \\ 2(x_2 - 2) + \mu_1 = 0 \\ x_1 + x_2 = 1 \end{cases}$$

Solution : $x = (0, 1)$, $\mu_1 = 2$

Vérification g_2 : $g_2(0, 1) = 1 > 0$, donc le point $x = (0, 1)$ est non admissible.

Cas 3 : $\mu_1 = 0$**,** $\mu_2 > 0$, on aura le système :

$$\begin{cases} 2(x_1 - 1) - 2\mu_2 x_1 = 0 \\ 2(x_2 - 2) + \mu_2 = 0 \\ x_2 = x_1^2 \end{cases}$$

Solutions :

- $x = (1, 1)$, $\mu_2 = 2 \Rightarrow g_1(1, 1) = 1 > 0$, donc le point $x = (1, 1)$ est non admissible.
- $x = (0.5, 0.25)$, $\mu_2 = 3.5 \Rightarrow g_1(0.5, 0.25) = -0.25 \leq 0$, donc le point $x = (0.5, 0.25)$ est admissible.

Cas 4 : $\mu_1 > 0$**,** $\mu_2 > 0$, le système est non linéaire complexe, donc il n'admet pas de solution admissible simple.

4. **Solution optimale**

   Le seul point admissible (un minimum local (et même global)) est $x^* = (0.5, 0.25)$ avec $\mu^* = (0, 3.5)$

2.3.4 Méthode classique d'optimisation non linéaire

2.3.4.1 Définition [38]

La méthode *Branch and Bound* est un algorithme d'optimisation globale utilisé pour résoudre des problèmes non linéaires complexes, notamment ceux impliquant des variables mixtes (continues et discrètes). Son principe repose sur une décomposition hiérarchique du problème initial en sous-problèmes plus simples (*branching*), associés à l'évaluation de bornes inférieure et supérieure des solutions optimales (*bounding*).

L'algorithme est généralement représenté sous la forme d'un arbre, où chaque nœud correspond à un sous-problème généré par une partition de l'espace des solutions. À chaque itération, un nœud actif est sélectionné et évalué selon une stratégie de recherche adaptée. Un nœud est exploré si sa borne inférieure indique une solution potentiellement meilleure que celle obtenue jusqu'à présent. À l'inverse, il est élagué dans trois cas :

- Il est prouvé qu'il ne contient pas de solution réalisable.
- Sa borne inférieure est supérieure à la meilleure solution connue.
- Une solution optimale y est trouvée.

2.3.4.2 Principe de l'algorithme de Branch and Bound [38]

Considérons le problème d'optimisation non linéaire suivant :

$$\min_{x \in D} f(x)$$

où $f : A \to \mathbb{R}$ est une fonction non linéaire, et $D \subseteq A \subseteq \mathbb{R}^n$ est l'ensemble des solutions admissibles.

L'algorithme repose sur une partition récursive du domaine D.

Définition 1 : Partition d'un domaine non linéaire [45]

Soit P un sous-ensemble compact de $\mathbb{R}^n$ contenant D, et soit I un ensemble fini d'indices. Une famille de sous-ensembles P_i est une partition de P si :

$$P = \bigcup_{i \in I} P_i, \quad P_i \cap P_j = \emptyset \quad \text{pour } i \neq j.$$

Dans le cas non linéaire, les sous-ensembles P_i peuvent être définis par des intervalles ou des approximations convexes.

Définition 2 : Évaluation des bornes (Bounding) [45]

Pour chaque sous-problème généré, on calcule :

- Une **borne inférieure** $L(P_i)$, obtenue via une relaxation du problème non linéaire. Exemples de relaxations :
  - Relaxation linéaire : linéarisation locale des contraintes.
  - Relaxation convexe : approximation convexe des fonctions non linéaires.
  - Analyse intervalle : encadrement des valeurs possibles de $f(x)$ sur P_i.
- Une **borne supérieure** $U(P_i)$, obtenue en évaluant $f(x)$ sur un point $x \in P_i$.

Si $U(P_i) - L(P_i)$ est inférieur à un seuil de tolérance ϵ, alors P_i est considéré comme résolu.

Définition 3 : Critères d'élagage (Pruning) [45]

L'élagage consiste à éliminer certains sous-problèmes de l'arbre de recherche afin de réduire l'espace de calcul et d'accélérer la convergence vers la solution optimale. Un sous-problème P_i est éliminé si :

- **Infeasibility** : un sous-problème ne contient aucune solution réalisable, il est immédiatement supprimé, c'est-à-dire :
  $P_i \cap D = \emptyset$.
- **Dominance** : la borne inférieure d'un sous-problème est supérieure à la meilleure solution connue jusqu'à présent, alors ce sous-problème ne peut pas conduire à une meilleure solution et est donc éliminé.
  $L(P_i) > U(P^*)$, où P^* est le meilleur sous-problème trouvé.
- **Solution optimale atteinte** : la borne inférieure et la borne supérieure d'un sous-problème sont égales, alors ce sous-problème est résolu de manière optimale et ne nécessite plus d'exploration.
  $U(P_i) = L(P_i)$.

2.3.4.3 Algorithme Branch and Bound

L'algorithme Branch and Bound suit plusieurs étapes pour résoudre un problème d'optimisation non linéaire [26, 45] :

Étape 0 : Initialisation

- Définir P_0 comme le domaine initial D.
- Calculer la borne inférieure $L(P_0)$ et la borne supérieure $U(P_0)$.
- Initialiser la meilleure solution $U^* = \infty$, indiquant qu'aucune solution optimale n'a encore été identifiée.

Étape 1 : Sélection du sous-problème à explorer

- Parmi les sous-problèmes actifs, choisir celui qui a la plus petite borne inférieure.

Étape 2 : Décomposition du sous-problème sélectionné

- Diviser P_i en plusieurs sous-problèmes $P_{i1}, P_{i2}, \ldots$.
- Calculer pour chaque sous-problème P_{ij} ses bornes inférieure $L(P_{ij})$ et supérieure $U(P_{ij})$.

Étape 3 : Élagage des sous-problèmes non prometteurs

- Supprimer un sous-problème P_{ij} si :
  - Il ne contient pas de solutions admissibles.
  - Sa borne inférieure est supérieure à la meilleure solution trouvée.

– Une solution optimale a été identifiée dans ce sous-problème.

– Mettre à jour U^* si une meilleure solution est trouvée.

Étape 4 : Arrêt de l'algorithme

– L'algorithme s'arrête lorsque tous les sous-problèmes restants ont une borne inférieure $L(P_i)$ supérieure ou égale à la meilleure solution trouvée U^*.

2.3.4.4 Exemple numérique : [21]

Pour illustrer cet algorithme, nous allons résoudre le problème suivant :

$$\min\ (x_1 - 4)^2 + (x_2 - 4)^2$$

sous les contraintes :

$$\begin{cases} 2x_1 + x_2 \leq 6.5 \\ x_1 + 2x_2 \leq 6 \\ x_1 \geq 0, \quad x_2 \geq 0 \\ x_1, x_2 \in \mathbb{Z} \end{cases}$$

Solution :

$$\boxed{\begin{array}{c} \min\ (x_1 - 4)^2 + (x_2 - 4)^2 \\ \\ \begin{cases} 2x_1 + x_2 \leq 6.5 \\ x_1 + 2x_2 \leq 6 \\ x_1 \geq 0, \quad x_2 \geq 0 \end{cases} \\ \textbf{Problème initial} \end{array}}$$

$$\Downarrow$$

$$\boxed{\begin{array}{c} x_1 = 2.3,\ x_2 = 1.8,\ z^* = 7.4 \\ \textbf{Solution non entière} \\ \rightarrow \text{Séparation} \end{array}}$$

$$\boxed{x_2 \geq 2} \qquad \boxed{x_2 \leq 1}$$

$$\Downarrow \qquad \Downarrow$$

$$\min\ (x_1-4)^2+(x_2-4)^2$$

$$\begin{cases} 2x_1+x_2\le 6.5\\ x_1+2x_2\le 6\\ x_2\ge 2\\ x_1\ge 0,\ x_2\ge 0 \end{cases}$$

Solution : $x_1=2,\ x_2=2,\ z^*=8$

Arrêt de cette branche

$$\min\ (x_1-4)^2+(x_2-4)^2$$

$$\begin{cases} 2x_1+x_2\le 6.5\\ x_1+2x_2\le 6\\ x_2\le 1\\ x_1\ge 0,\ x_2\ge 0 \end{cases}$$

Solution :
$x_1=2.75,\ x_2=1,\ z^*=10.5$

Solution non entière $\rightarrow$
Séparation

$$\boxed{x_1\ge 3} \qquad \boxed{x_1\le 2}$$
$$\Downarrow \qquad\qquad \Downarrow$$

$$\min\ (x_1-4)^2+(x_2-4)^2$$

$$\begin{cases} 2x_1+x_2\le 6.5\\ x_1+2x_2\le 6\\ x_2\le 1\\ x_1\ge 3\\ x_1\ge 0,\ x_2\ge 0 \end{cases}$$

Solution :
$x_1=3,\ x_2=0,\ z^*=17$

$$\min\ (x_1-4)^2+(x_2-4)^2$$

$$\begin{cases} 2x_1+x_2\le 6.5\\ x_1+2x_2\le 6\\ x_2\le 1\\ x_1\le 2\\ x_1\ge 0,\ x_2\ge 0 \end{cases}$$

Solution :
$x_1=2,\ x_2=1,\ z^*=13$

Comparaison des solutions : $17>13>8$
Solution entière optimale : $x_1=2,\ x_2=2,\ z^*=8$

Chapitre 3

Programmation dynamique

3.1 Introduction

La **programmation dynamique** (DP), est une méthode révolutionnaire d'optimisation des processus décisionnels, a été formalisée par Richard Bellman en 1957. Initialement développée pour résoudre des problèmes de chemins optimaux (recherche de trajectoires minimales ou maximales), cette approche s'est révélée particulièrement efficace pour une classe spécifique de problèmes mathématiques possédant une propriété de décomposabilité remarquable.

Le cœur de cette méthode repose sur un principe fondamental la décomposition systématique d'un problème complexe en une série de sous-problèmes plus simples. L'originalité de l'approche réside dans la combinaison intelligente des solutions optimales de ces sous-problèmes pour construire la solution globale optimale, évitant ainsi les calculs redondants grâce à des techniques de mémorisation.

Contrairement aux limitations des méthodes d'optimisation classiques, la programmation dynamique a démontré son efficacité bien au-delà du cadre initial des problèmes linéaires convexes. Elle s'applique avec succès à des problématiques non linéaires et non convexes complexes, là où les approches traditionnelles échouent souvent. Cette adaptabilité remarquable explique son adoption massive dans des domaines aussi variés que la recherche opérationnelle, l'intelligence artificielle ou l'économie mathématique.

3.2 Rappels de programmation dynamique

3.2.1 Définitions

Définition.1. (Systéme dynamique)[19]

La **dynamique** est un processus évolutif dans le temps. L'ensemble des équations décrivant cette dynamique est appelé un **système dynamique**.

Un système dynamique peut être représenté **à temps continu** ou **à temps discret**.

1. **Définition : (Systèmes dynamiques continus)**
   Un système dynamique continu est un triplet $(X, \mathbb{R}, \{f_t\}_{t\in\mathbb{R}})$ où :
   - X est un espace (souvent une variété différentielle).
   - $\{f_t : X \to X\}_{t\in\mathbb{R}}$ est une famille d'applications vérifiant :
     (a) $f_{t+s} = f_t \circ f_s$ (propriété de semi-groupe).
     (b) $f_0 = \mathrm{Id}_X$.
     (c) L'application $(t, x) \mapsto f_t(x)$ est continue.

   La variable t représente le temps.

   Un système dynamique continu est dit :
   – **Autonome** : si f_t ne dépend pas explicitement du temps (i.e. f_t est définie par un flot constant).
   – **Non autonome** : si f_t dépend explicitement du temps.
2. **Définition : (Systèmes dynamiques discrets)**
   Soit $E \subset \mathbb{C}$. Un système dynamique discret est un couple (E, f) où :
   - $f : E \to E$ est une application.
   - La trajectoire issue de $x_0 \in E$ est la suite $\{x_n\}_{n\in\mathbb{N}}$ définie par :

   $$x_{n+1} = f(x_n) \quad \text{pour tout } n \geq 0.$$

   On note f^n la n-ième itérée de f :

   $$f^n = \underbrace{f \circ f \circ \cdots \circ f}_{n \text{ fois}}$$

Définition.2. (Fonction décomposable)[29]

On dit que f est décomposable en f_1 et f_2 si f peut se mettre sous la forme :

$$f(x, y) = f_1(x, f_2(y))$$

et si, de plus, la fonction f_1 est monotone non décroissante par rapport à **son deuxième argument** $f_2(y)$ (le résultat de l'application de f_2 à y) ; c'est-à-dire que si $f_2(y)$ augmente, alors $f_1(x, f_2(y))$ ne diminue pas (elle reste constante ou augmente).

On peut alors énoncer **le théorème d'optimalité de Bellman** :

Théorème. ***[29]*** *: Soit f une fonction réelle de x et de $y = (y_1, y_2, \ldots, y_k)$. Supposons f décomposable avec $f(x, y) = f_1(x, f_2(y))$ et (*Opt $=$ min *ou* max*) et qu'il existe $\bar{y} \in \mathbb{R}^k$ et $\bar{x} \in \mathbb{R}$ tels que :*

$$f_2(\bar{y}) = \mathrm{Opt}_y\{f_2(y)\}$$
$$f_1(\bar{x}, f_2(\bar{y})) = \mathrm{Opt}_x\, f_1(x, f_2(\bar{y})).$$

Alors on a :

$$\mathrm{Opt}_{(x,y)}\, f(x, y) = \mathrm{Opt}_x\{f_1(x, \mathrm{Opt}_y\{f_2(y)\})\}$$

3.2.2 Décomposition de Bellman

Le théorème précédent permet de transformer un problème complexe en une suite de sous problèmes plus simples.

3.2.2.1 Cas particulier à 2 variables :

Considérons le problème :

$$(P) \quad F = \mathrm{Opt}\ f(x, y) \quad \text{s.c. } g(x, y) \in E_t$$

Supposons que f et g soient décomposables :

$$g(x, y) = g_2(y, g_1(x))$$

On pose :

$$E_1 = g_1(x), \quad E_2 = g_2(y, E_1)$$

On résout d'abord un sous-problème :

$$(P_2(E_1)) \quad F_2(E_1) = \text{Opt}_{y \in \Omega_2(E_1)} f_2(y)$$

Le problème (P) devient :

$$F = \text{Opt}_x f_1(x, F_2(g_1(x)))$$

Remarques :

– Si $g_1(x)$ n'est pas défini ou si $\Omega_2(E_1) = \emptyset$, on pose :

$$F_2(E_1) = \pm\infty$$

Cela signifie que le sous-problème n'a pas de solution admissible, et est donc écarté de l'optimisation globale.

– La **décomposition** de f et g est une **condition cruciale** pour pouvoir appliquer la méthode de programmation dynamique. Sans cette propriété, il n'est pas possible de ramener le problème global à une suite de sous-problèmes indépendants.

3.2.2.2 Généralisation à n variables :

Équation de Bellman (récurrence d'ordre k)

$$F_k(E_{k-1}) = \text{Opt}_{x_k} f_{k-1}(x_k, F_{k+1}(g_k(x_k, E_{k-1}))) \quad \text{pour } k < n$$

$$F_n(E_{n-1}) = \text{Opt}_{x_n / g_n(x_n, E_{n-1}) \in E_t} f_n(x_n)$$

Avec :

$$\text{États} : E_1 = g_1(x_1), \quad E_2 = g_2(x_2, E_1), \quad \ldots$$

3.2.2.2 Exemple :[15]

Problème Original (P)

$$(P)\begin{cases}\max & f(x) = 2x_1 + 4x_2 + 7x_3 + 10x_4 \\ \text{s.c.} & g(x) = 3x_1 + 5x_2 + 8x_3 + 10x_4 \leq 15 \\ & x_j \in \{0,1\} \quad \forall j \in \{1,2,3,4\}\end{cases}$$

Décomposition Additive : On sépare le problème en deux sous-problèmes indépendants :

– **Sous-problème 1 (P1)** : Variables x_1, x_2

$$\begin{cases} f_1(x_1, x_2) = 2x_1 + 4x_2 \\ g_1(x_1, x_2) = 3x_1 + 5x_2 \end{cases}$$

– **Sous-problème 2 (P2)** : Variables x_3, x_4

$$\begin{cases} f_2(x_3, x_4) = 7x_3 + 10x_4 \\ g_2(x_3, x_4) = 8x_3 + 10x_4 \end{cases}$$

Relation entre sous-problèmes : La contrainte globale se décompose en :

$$g(x) = \underbrace{3x_1 + 5x_2}_{g_1(x_1,x_2)} + \underbrace{8x_3 + 10x_4}_{g_2(x_3,x_4)} \leq 15$$

Pour résoudre le problème (P), nous présentons maintenant des méthodes exactes garantissant l'optimalité, ainsi que des méthodes approchées pour les cas complexes.

3.3 Position du problème

Nous considérons le programme mathématique non linéaire, en général non convexe, non séparable en variables entières, noté $(P_{N,M})$, qui s'énonce de la manière suivante :

$$(P_{N,M})\begin{cases} \max\limits_{(x_1,\ldots,x_N)} f(x) = \sum\limits_{i=1}^{N} \left[p_i - c_i \cdot g\left(\sum\limits_{k=1}^{i} x_k \right) \right] x_i \\ \text{s.c.} \sum\limits_{i=1}^{N} x_i = M \\ \forall i,\ 0 \leq x_i \leq M \\ \forall i,\ x_i \in \mathbb{N} \end{cases}$$

où

- g est une fonction croissante de $\mathbb{N}$ dans $\mathbb{R}^+$, est appelée fonction de pénalité.
- Les coefficients p_i et c_i sont des réels strictement positifs.
- Les x_i, pour i allant de 1 à N, sont les N variables de décisions du problème. Elles prennent leurs valeurs parmi les entiers naturels.
- Le problème n'a qu'une contrainte linéaire d'égalité, de second membre M, où M est une constante positive.

Le problème $P_{N,M}$, considéré ici sous une forme déterministe et discrétisée, correspond à un modèle de *liquidation optimale de portefeuille* (ou *écoulement de larges blocs d'actifs*). Il consiste à vendre une quantité totale M d'un actif financier en N étapes, en décidant des quantités x_i à écouler à chaque instant i. L'objectif est de maximiser le revenu global de la vente, tout en prenant en compte l'impact négatif de cette liquidation sur le marché : plus la quantité cumulée vendue augmente, plus le prix unitaire est pénalisé. Cet effet est modélisé par une fonction de pénalité croissante appliquée au volume cumulé, ce qui reflète la perte de valeur due à l'effet de marché.

3.4 Résolution exacte

Nous établissons ici formellement l'équation de Bellman et décrivons l'algorithme de programmation dynamique proposée par Bellman, permettant une résolution exacte du problème $(P_{N,M})$.

Considérant le problème $P_{n,m}$ suivant :

$$(P_{n,m}) \begin{cases} \max\limits_{(x_1,\ldots,x_n)} f(x) = \sum\limits_{i=1}^{n} \left[p_i - c_i g\left(\sum\limits_{k=1}^{i} x_k \right) \right] x_i \\ \text{s.c} \ : h(x) = \sum\limits_{i=1}^{n} x_i - m = 0 \\ \forall i, x_i \in \mathbb{N} \\ \forall i, 0 \leq x_i \leq m \end{cases}$$

Le problème $(P_{n,m})$, qui est équivalent à $(P_{N,M})$ lorsque $n = N$ et $m = M$; est un problème en variables entières, dont les variables de décisions sont bornées. Il admet donc des solutions pour $n \leq m$. Soit $O_{n,m}$ sa valeur optimale.

Théorème. ***[Equation de Bellman [6]]*** *Pour tout n, m avec $1 \leq n \leq m$:*

$$O_{n,m} = \max_{i \in [0;m]} \{O_{n-1,m-i} + (p_n - c_n g(m))\, i\}$$

Sous les conditions initiales $O_{n,0} = O_{0,m} = 0$.

L'équation de Bellman du théorème fournit un algorithme récursif de calcul de la valeur optimale $O_{n,m}$. Nous présentons l'algorithme de résolution exacte par programmation dynamique (Algorithme 1) [8] :

Algorithm 1 Programmation dynamique exacte

Construction de la matrice $O_{n,m}$
for $n = 1$ **to** N **do**
  for $m = 1$ **to** M **do**
    for $k = 0$ **to** m **do**
      $O_{n,m} = \max(O_{n,m}, O_{n-1,m-k} + (p_n - c_n g(m)) \cdot k)$
    end for
  end for
end for
Calcul de la stratégie optimale
$m = M$
for $i = N$ **to** 1 **do**
  if $O_{i,m} = O_{i-1,m}$ **then**
    $x_i \leftarrow 0$
  else
    $k \leftarrow m$
    while $(O_{i,m} \neq O_{n-1,m-k} + (p_n - c_n g(m)) \cdot k)$ **and** $(k > 0)$ **do**
      $k \leftarrow k - 1$
    end while
    $x_i \leftarrow k$
    $m \leftarrow m - k$
  end if
end for

Exemple

Revenons à l'exemple précédent : en appliquant cet algorithme de programmation dynamique, nous trouvons que la valeur optimale du problème est 14, obtenue avec la solution $\mathbf{x} = (0, 1, 0, 1)^t$.

3.5 Résolutions approchées

3.5.1 Cas du problème discret

Il existe plusieurs heuristiques naïves pour générer des solutions admissibles dans des problèmes non convexes et non linéaires en variables entières, comme des répartitions fixes ou aléatoires. Cependant, notre objectif ici est de proposer une méthode de résolution plus structurée et efficace, fondée sur la programmation dynamique.

3.5.1.1. Méthode DP en deux étapes (TSDP)

La méthode TSDP se décompose en deux étapes indépendantes visant à résoudre le problème de manière efficace tout en conservant une qualité de solution acceptable.

1. **Première étape – Programmation dynamique à gros grain (heuristique rapide)[37]** :

   Lorsque le nombre de périodes N est grand, l'algorithme de *programmation dynamique exacte* devient trop coûteux, avec une complexité temporelle en $\mathcal{O}(NM^2)$. Pour réduire le coût de calcul, on regroupe les décisions en **paquets de P unités**, appelés **grains** P, et on applique l'algorithme de DP sur ces blocs.

   – Cela revient à changer d'échelle : la nouvelle capacité maximale devient $M' = \lfloor M/P \rfloor$.
   – On applique alors exactement le même algorithme de DP, mais avec des unités de taille P.
   – La complexité devient alors :

   $$\mathcal{O}\left(\frac{NM^2}{P^2}\right)$$

   offrant ainsi une accélération d'un facteur P^2.

   En contrepartie :
   – Plus P est grand, plus l'algorithme est rapide,
   – mais la solution est moins précise.

Il y a donc un compromis à trouver entre la qualité de la solution et le temps de calcul.

2. **Deuxième étape – Programmation dynamique avec bornes [17]** :
   Dans cette étape, on part d'une solution admissible obtenue dans la première étape (approximative), supposée proche de l'optimum. L'objectif est de *raffiner* cette solution en explorant son *voisinage* via une programmation dynamique exacte, mais en imposant des **bornes** sur les solutions admissibles.
   - Soit $\mathbf{x_0} = (x_0^1, x_0^2, \ldots, x_0^N)$ la solution initiale.
   - Pour chaque i, on impose des bornes inférieure et supérieure $l_i \leq x_i \leq u_i$, avec $0 \leq l_i \leq x_i \leq u_i \leq M$.
   - On définit :
   $$L_i = \min\left(\sum_{k=1}^{i} l_k, M\right), \quad U_i = \min\left(\sum_{k=1}^{i} u_k, M\right)$$
   - L'équation de Bellman restreinte devient :
   $$O_n(m) = \max_{l_n \leq k \leq u_n} \left(O_{n-1}(m-k) + p_n - c_n g_m(k)\right), \quad \text{sous contrainte } L_n \leq m = y_n \leq U_n$$

Algorithm 2 Programmation dynamique avec bornes

```
/* Construction de la matrice (On,m) */
for n = 1 to N do
  for m = Ln to Un do
    for k = ln to un do
      O_{n,m} = max(O_{n,m}, O_{n-1,m-k} + p_t − c_t · g_m(k))
    end for
  end for
end for
/* Partie retours-arrière (backtracking) identique à celle de l'algorithme
(1) */
/* Trouve la solution par retours-arrière en O_{N,M} matrix */
```

3.5.2 Cas du problème continu

Nous considérons le problème d'optimisation continue noté ($\overline{P_{N,M}}$), dont l'objectif principal est le calcul de bornes inférieures de qualité (solutions admissibles approchées). En l'absence d'hypothèse de convexité, les solutions obtenues

correspondent uniquement à des extremums locaux, sans garantie d'optimalité globale.

Lorsque la condition $M \geq N$ est satisfaite, les solutions continues du problème $(\overline{P_{N,M}})$ constituent des approximations pertinentes pour la version en variables entières. Cette propriété motive la recherche de solutions entières performantes dans le voisinage de ces extrema continus. D'un point de vue théorique, le problème $(\overline{P_{N,M}})$ présente une fonction objectif de classe $\mathcal{C}^1$ définie sur un compact, ce qui assure l'existence d'un maximum global. Cependant, la détermination effective de ce maximum reste aussi complexe que pour le cas en variables entières.

Dans ce contexte, nous étudierons la méthode de gradient projeté pour résoudre ce problème d'optimisation.

3.5.2.1. Méthode de gradient projeté [42, 43]

La méthode de gradient projeté est une approche itérative permettant de résoudre des problèmes sous contraintes. Elle combine une étape de descente de gradient (pour minimiser la fonction objectif) et une projection orthogonale (pour garantir que les itérés restent dans l'ensemble des solutions admissibles).

1. **Projection sur l'hyperplan admissible**
   Pour la contrainte linéaire $C = \{x \in \mathbb{R}^N \mid \sum_{i=1}^{N} x_i = M\}$:
   - Soit x un vecteur de $\mathbb{R}^N$ et $P_C(x)$ sa projection orthogonale sur C.
   - Par définition, $x - P_C(x)$ est orthogonal à C.
   - Le gradient de la contrainte $(\sum_{i=1}^{N} x_i - M)$ est le vecteur $(1, \ldots, 1)$.
   - Nous en déduisons $\forall i,\ P_C(x)_i = x_i + \mu \cdot 1$.

     $P_C(x)$ est une stratégie admissible. En sommant sur i, nous obtenons :

     $$\mu = \frac{1}{T}\left(N - \sum_{i=1}^{T} x_i\right)$$

     Nous en concluons pour l'opérateur P_C :

     $$\forall i,\ P_C(x)_i = x_i + \frac{1}{T}\left(N - \sum_{j=1}^{T} x_j\right)$$

2. **Algorithme Itératif**

   (a) **Initialisation** : Partir d'une solution admissible $x_0 \in C$.

   (b) **Mise à jour du gradient** :
   - Calculer la direction de descente : $d_0 = \alpha \cdot \nabla f(x_0)$,
   - Où $\alpha > 0$ est le pas de gradient (constant)
   - Notation simplifiée : $\nabla f(x_0)_i \equiv \nabla f_i^0$.

   (c) **Projection orthogonale** :

$$x_1 = \mathcal{P}_C\left(x_0 + \alpha \nabla f^0\right)$$
$$\text{avec } \mathcal{P}_C(x_1)_i = x_{0i} + \alpha\left(\nabla f_i^0 - \frac{1}{T}\sum_{j=1}^{T} \nabla f_j^0\right)$$

   (d) **Itération** : Répéter pour x_k jusqu'à convergence :

$$\|\nabla f_k\| \leq \varepsilon$$

   où $\varepsilon > 0$ est un seuil de tolérance prédéfini.

Conclusion

La résolution exacte par programmation dynamique (DP) garantit une solution optimale pour le problème $\mathrm{P}_{N,M}$, mais sa complexité en $\mathcal{O}(NM^2)$ la rend inapplicable aux grandes instances. Pour ces cas, des méthodes approchées offrent un compromis entre précision et efficacité.Pour les problèmes de structure **discrète**, la méthode TSDP (Two-Step Dynamic Programming) fournit une solution rapide en deux phases, une approximation gros grain suivie d'un affinement local. Bien qu'elle ne garantisse pas l'optimalité, elle préserve une structure computationnelle efficace. Dans le cas **continu**, la méthode du gradient projeté est employée pour explorer efficacement l'espace des solutions. Parallèlement, ces techniques d'optimisation peuvent également être utilisées dans des configurations où les méthodes discrètes classiques atteignent leurs limites. Leur flexibilité en fait un complément utile aux approches purement discrètes.

3.6 Approche de séparation pour la borne supérieure

Lorsqu'on résout des problèmes d'optimisation (en variables entières ou continues), il est souvent utile d'encadrer la solution optimale entre :

– Une **borne inférieure** (solution réalisable).

– Une **borne supérieure** (limite théorique du meilleur résultat possible).

Dans certains cas, la résolution exacte par programmation dynamique (PD) est trop coûteuse en temps ou en mémoire. Nous proposons ici une méthode efficace pour calculer une borne supérieure.

3.6.1 Simplification du problème

L'idée principale est de remplacer le problème complexe par un problème plus simple qui le majore. Pour cela, on utilise deux propriétés :

– La fonction de pénalité g est **strictement croissante**.

– Les variables x_i sont **positives**.

Cela permet d'établir l'inégalité :

$$g\left(\sum_{k=1}^{i} x_k\right) \geq g(x_i),$$

et donc de majorer la fonction objectif originale par :

$$f(x) \leq \sum_{i=1}^{N} \left[p_i - c_i \times g(x_i)\right] x_i = U(x).$$

Propriétés : [14, 32, 39]

– **Séparabilité** : $U(x)$ se décompose en une somme de fonctions à une seule variable.

– **Convexité** : Si $x \times g(x)$ est convexe, alors $U(x)$ est concave et donc le problème est bien posé.

Les propriétés établies ci-dessus nous offrent un cadre favorable pour approximer le problème initial par un programme non linéaire plus simple. Par la suite, nous utilisons cette reformulation pour construire une borne supérieure, en résolvant un problème séparé et analytiquement plus tractable.

3.6.2 Calcul d'une première borne supérieure

Une première borne supérieure est proposée en majorant le problème initial par un problème séparé, plus simple à résoudre analytiquement, sous certaines hypothèses de régularité sur la fonction de pénalité g.

En supposant que g est **strictement croissante** et que les variables de décision x_i sont **positives**, on peut minorer $g\left(\sum x_k\right)$ par $g(x_i)$, ce qui permet de séparer la fonction objectif.

Le problème obtenu est un programme non linéaire continu, défini par :

$$UB_2 = \max_{x \in C} U(x), \text{ avec } \quad U(x) = \sum_{n=1}^{N} u_i(x_i) = \sum_{n=1}^{N} \left(p_i - c_i \times g(x_i)\right) x_i$$

Ce problème constitue une **majoration** des problèmes initiaux, et présente l'avantage d'être **séparable**, c'est-à-dire que la fonction objectif s'écrit comme une somme de fonctions univariées.

Une condition suffisante sur g est ensuite donnée pour garantir la **convexité** de ce nouveau problème.

3.6.3 Résolution d'un problème séparé

Nous nous intéressons au problème non linéaire, en variables continues, précédant :

$$UB_2 = \max_{x \in C} U(x), \text{ avec } \quad U(x) = \sum_{i=1}^{N} \left[p_i - c_i \times g(x_i)\right] x_i$$

La résolution de ce problème repose sur le calcul du Lagrangien et l'analyse des conditions de Karush-Kuhn-Tucker (KKT). Sous les hypothèses de régularité sur la fonction de pénalité g (croissance stricte, continuité) et la linéarité de la contrainte, les conditions de KKT sont satisfaites, et les points stationnaires correspondent à des optima globaux.

Lorsque les vecteurs de prix p et de coûts c sont constants, la solution optimale est donnée par :

$$\hat{x} = \left(\frac{M}{N}, \ldots, \frac{M}{N}\right), \quad \text{et} \quad UB_2 = M\left(p - c \times g\left(\frac{M}{N}\right)\right)$$

Cette solution découle de l'injectivité et de la stricte convexité de la fonction $x \mapsto x \times g(x)$.

Dans le cas plus général où p et c varient, une borne supérieure reste accessible en posant $p = \max_i p_i$ et $c = \min_i c_i$, ce qui conduit à :

$$UB_2 \leq M \left(p - c \times g \left(\frac{M}{N} \right) \right)$$

Une résolution analytique est également envisageable en supposant $x_i > 0$ pour tout i. En sommant les conditions de KKT, on obtient l'équation implicite suivante :

$$\sum_{i=1}^{N} g^{-1} \left(\frac{p_i - \lambda}{c_i} \right) = M$$

Cette équation peut être résolue numériquement (par exemple via la méthode de Newton). Pour toute solution λ, on déduit une solution optimale :

$$x_i^{\lambda} = g^{-1} \left(\frac{p_i - \lambda}{c_i} \right)$$

3.7 Convexification d'un problème

Le problème considéré est non convexe, ce qui signifie que les optima locaux ne sont pas nécessairement globaux. Dans ce contexte, une convexification est nécessaire : on majore le problème initial par un problème convexe plus simple à résoudre, ce qui permet d'obtenir une borne supérieure plus fine. En répétant ce procédé, on peut construire une suite de problèmes convexes de plus en plus précis, espérant atteindre l'optimum global ou s'en rapprocher à ε-près. Pour cela, il faut préalablement transformer le problème pour isoler ses termes non convexes d'où l'intérêt de la factorisation.

3.8 Factorisation d'un problème

La factorisation consiste à transformer un problème en un autre équivalent en introduisant des variables auxiliaires pour décomposer la fonction objectif en sommes de produits de fonctions univariées (notamment linéaires ou bilinéraires).

Cette méthode permet d'isoler les sources de non-convexité, ce qui facilite la convexification.

Exemple

Prenons par exemple le problème (P1) défini par :

$$\begin{array}{ll} \min\limits_{x\in\mathbb{R}^2} & f(x) = \cos(x_1 + x_2) + x_1^2 + 7x_2^2 - 2x_1x_2 \\ \text{s.c.} & \left\{ \begin{array}{l} a_1 \leq x_1 \leq b_1 \\ a_2 \leq x_2 \leq b_2 \end{array} \right. \end{array}$$

Nous introduisons les variables auxiliaires suivantes :

$$\begin{aligned} x_3 &= x_1 + x_2 \\ x_4 &= \cos(x_3) - x_3^2 \\ x_5 &= x_4 + 2x_1^2 + 8x_2^2 \end{aligned}$$

Le problème (P2) transformé s'écrit alors :

$$\begin{array}{ll} & \min\ x_5 \\ \text{s.c.} & \left\{ \begin{array}{l} x_1 + x_2 - x_3 = 0 \\ \cos(x_3) - x_3^2 - x_4 = 0 \\ x_5 = x_4 + 2x_1^2 + 8x_2^2 \\ a_1 \leq x_1 \leq b_1 \\ a_2 \leq x_2 \leq b_2 \\ a_1 + a_2 \leq x_3 \leq b_1 + b_2 \\ a_4 \leq x_4 \leq b_4 \end{array} \right. \end{array}$$

Les bornes a_4 et b_4 sont déterminées par une analyse réelle de la fonction univariée $x \mapsto \cos(x) - x^2$.

Les problèmes (P1) et (P2) sont équivalents et tous deux non convexes. Toutefois, (P2) présente une structure séparable qui facilite l'analyse. La non-convexité est concentrée dans le terme $\cos(x_3) - x_3^2$, qui est une fonction concave sur $\mathbb{R}$. En utilisant une fonction minorante convexe de ce terme, on peut obtenir une relaxation convexe du problème, utile pour les méthodes de type *Branch and Bound*.

Définition (Fonction factorisable)[27]

Une fonction $f : \mathbb{R}^n \to \mathbb{R}$ est dite **factorisable** si elle peut s'écrire comme une somme de produits de fonctions univariées continues provenant d'un ensemble donné O, et dont les arguments sont des variables, des constantes ou d'autres fonctions factorisables.

Un ensemble typique est :

$$O = \{+,\ \times,\ /,\ \wedge,\ \exp,\ \log,\ \sin,\ \cos\}$$

3.8.1 Problèmes factorisables

Un problème d'optimisation est dit **factorisable** si la fonction objectif ainsi que toutes ses contraintes peuvent être exprimées comme des combinaisons de fonctions factorisables, c'est-à-dire des sommes de produits de fonctions univariées issues d'un ensemble donné [**27**].

3.8.1.1. Factorisation dans le cas à deux variables (cas convexe)[27]

Dans ce cas simple, la contrainte $x_1 + x_2 = M$ permet de réécrire la fonction objectif comme une fonction univariée :

$$f(x_1) = (p_1 - c_1 \times g(x_1))x_1 + (p_2 - c_2 \times g(M))(M - x_1)$$

La seule non-linéarité réside dans le terme $-c_1 \times g(x_1) \times x_1$. En supposant que la fonction g est croissante et concave, et que l'application $x \mapsto x \times g(x)$ est convexe, ainsi que :

$$\sum_{i=1}^{N} g^{-1}\left(\frac{p_i}{c_i}\right) = M,$$

alors la fonction f est concave et le problème est donc convexe. Dans ce contexte, la factorisation n'apporte aucun avantage particulier : le problème peut être résolu directement, soit par analyse réelle dans le cas continu, soit par programmation dynamique dans le cas discret.

3.8.1.2. Factorisation dans le cas à trois variables (apparition d'un produit bilinéaire) [27]

Pour le problème $(P_3; M)$ la factorisation devient utile, sa fonction objectif s'écrit :

$$f(x) = (p_1 - c_1 g(x_1))x_1 + (p_2 - c_2 g(x_1 + x_2))x_2 + (p_3 - c_3 g(M))x_3$$

On introduit les variables auxiliaires (qui sont des variables artificielles liées aux variables originales par des équations de calcul, introduites pour simplifier tout en préservant l'équivalence du problème) suivantes :

$$X_1 = x_1, \quad X_2 = x_2, \quad X_3 = x_3$$
$$X_4 = X_1 + X_2, \quad X_5 = X_1 + X_2 + X_3 = M$$
$$X_6 = p_1 X_1 - c_1 g(X_1)X_1 + p_2 X_2 - c_2 g(X_4)X_2 + (p_3 - c_3 g(M))X_3$$

Pour alléger l'écriture, on pose $H = g(M)$, qui représente le seuil de calibration de la fonction de pénalité. En pratique, on prend $H = 0.99$.

Le problème factorisé devient :

$$\max \ X_6$$
$$\text{s.c.} \quad \begin{cases} X_4 = X_1 + X_2 \\ X_5 = X_1 + X_2 + X_3 = M \\ 0 \leq X_i \leq M, \quad \forall i \in \{1, \ldots, 5\} \\ X_i \in \mathbb{N}, \quad \forall i \in \{1, \ldots, 6\} \end{cases}$$

Toutes les contraintes sont linéaires (donc concaves). Sous les hypothèses usuelles sur g, le seul terme non concave est $-c_2 X_2 g(X_4)$. Il appartient à une catégorie de produits bilinéaires pouvant être convexifiés. Cette structure va maintenant être généralisée.

3.8.1.3. Factorisation dans le cas général [27]

On généralise la factorisation à un nombre arbitraire N de variables. On pose $X_i = x_i$ pour $1 \leq i \leq N$.

Chaque terme de la forme $g\left(\sum_{k=1}^{i} x_k\right) x_i$ est représenté à l'aide d'une variable auxiliaire $X_{N+i-1} = \sum_{k=1}^{i} x_k$, pour $2 \leq i \leq N-1$.
La somme totale $X_{2N-1} = \sum_{i=1}^{N} x_i = M$ est également explicitée.

Le problème factorisé s'écrit :

$$\max\ X_{2N} = \sum_{i=1}^{N} p_i X_i - (c_1 g(X_1) X_1 + c_N g(M) X_N) + \sum_{i=2}^{N-1} (-c_i) X_i g(X_{N+i-1})$$

$$\text{s.c.} \quad \begin{cases} X_i = x_i, \quad \forall i \in \{1, \ldots, N\} \\ X_{N+i-1} = \sum_{j=1}^{i} X_j, \quad \forall i \in \{2, \ldots, N-1\} \\ X_{2N-1} = \sum_{j=1}^{N} X_j = M \\ 0 \leq X_i \leq M, \quad \forall i \in \{1, \ldots, 2N-1\} \\ X_i \in \mathbb{N}, \quad \forall i \in \{1, \ldots, 2N\} \end{cases}$$

Ce problème comprend :

- $N-2$ produits non concaves du type $X_i \times g(X_{N+i-1})$.
- $N-1$ contraintes d'égalité linéaires.
- $2N$ variables au total (dont $N-1$ auxiliaires).

Remarque 1 : Lorsque les N premières variables de décision sont fixées, toutes les variables auxiliaires ainsi que la fonction objectif sont automatiquement déterminées.

Remarque 2 : Réciproquement, si les variables auxiliaires X_{N+i-1} sont fixées de $i = 2$ à $N-2$, alors les variables initiales sont toutes déterminées à l'exception de X_1 et X_2, qui sont liées par $X_1 + X_2 = X_4$. Il reste donc un seul degré de liberté.

3.9 Relaxation convexe

Pour résoudre efficacement le problème non convexe $P_{N,M}$, il est essentiel de construire une relaxation convexe basée sur les enveloppes convexes et concaves. Cette étape joue un rôle central dans le bon fonctionnement des algorithmes de *Branch and Bound*, en fournissant des bornes utiles à l'exploration de l'espace des solutions.

3.9.1 Définitions

On considère un programme mathématique en variables entières défini par :

$$(P) \quad \min_{x \in \mathbb{R}^n} f(x)$$

$$\text{s.c.} \quad \begin{cases} c_j(x) \leq 0, & j = 1, \ldots, p \\ x_i \in \mathbb{N}, & \forall i \end{cases}$$

Dans un contexte de maximisation, on adopte la définition symétrique :

$$(P) \quad \max_{x \in \mathbb{R}^n} f(x)$$

$$\text{s.c.} \quad \begin{cases} c_j(x) \geq 0, & j = 1, \ldots, p \\ x_i \in \mathbb{N}, & \forall i \end{cases}$$

Dans ce cas, (P) est un problème convexe si les fonctions f et c_j, pour tout j, sont **concaves**.

Dans la suite, et conformément à l'usage, seul le cadre des problèmes et relaxations convexes sera utilisé, sans perte de généralité. Toutefois, pour un problème de maximisation, la construction d'une fonction majorante concave (appelée *surestimation concave*) de la fonction objectif est nécessaire [27].

3.9.2 Cas des fonctions factorisables

La factorisation du problème étudié précédemment permet de ne considérer que des fonctions univariées, issues de produits du type $X \times g(Y)$, où g est une fonction réelle de classe $\mathcal{C}^2$. On se ramène alors à l'étude de la convexité d'une fonction univariée f sur un intervalle réel fermé $[L, U]$.

3.9.1.1. Identification des intervalles de convexité

On cherche à déterminer les intervalles de convexité monotone de f, sur lesquels elle est strictement convexe ou concave. Deux méthodes classiques permettent de localiser les points d'inflexion :

- Étudier les annulations et changements de signe de f''.
- Repérer les points où la tangente traverse la courbe, c'est-à-dire les solutions de :

$$\frac{f(x) - f(a)}{x - a} = f'(x), \quad \text{avec } a < x,\ a \in [L, U].$$

Cette dernière équation peut être résolue analytiquement ou numériquement. Par exemple par une méthode simple basée sur la méthode de Newton, appliquée à la fonction :

$$F(x) = f(x) - f(a) - (x - a)f'(x) \quad \Rightarrow \quad F'(x) = -(x - a)f''(x)$$

d'où la récurrence :

$$x_{k+1} = x_k + \frac{1}{f''(x_k)} \left(\frac{f(x_k) - f(a)}{x_k - a} - f'(x_k) \right)$$

Cette approche permet de subdiviser l'intervalle $[L, U]$ en sous-intervalles $[l_i, l_{i+1}]$ sur lesquels la convexité est monotone.

3.9.1.2. Construction de l'enveloppe concave

Dans un problème de maximisation :

- Sur les intervalles concaves, la fonction est sa propre enveloppe concave.
- Sur les intervalles convexes, l'enveloppe concave est la sécante reliant $(l_i, f(l_i))$ à $(l_{i+1}, f(l_{i+1}))$.

Ce principe est discuté par Falk et Soland. Une interprétation géométrique, proposée par Horst et Tuy, lie l'enveloppe convexe à la fermeture convexe de l'épigraphe de f :

$$\text{epi}(\tilde{f}) = \text{conv}(\text{epi}(f))$$

3.10 Convexification des problèmes factorisables : Cas général

Dans le cadre des problèmes d'optimisation factorisables, l'une des stratégies utilisées consiste à "convexifier" les problèmes non convexes afin de les rendre plus faciles à résoudre. Cependant, cette approche n'est pas toujours simple et doit être appliquée avec soin. Le processus de convexification consiste à remplacer une fonction non convexe par une approximation convexe, ce qui permet d'utiliser des méthodes d'optimisation plus efficaces.

Définition [33] :

Soit $T : \mathbb{R} \to \mathbb{R}$ une fonction continue représentant la fonction objectif d'un problème d'optimisation (P).
Soit $S \subset \mathbb{R}^n$ l'ensemble des solutions admissibles de (P).
Soit $t : \mathbb{R}^n \to \mathbb{R}$ une fonction continue.

Soient $c : \mathbb{R}^n \to \mathbb{R}$ une fonction continue et convexe, et $C : \mathbb{R}^n \to \mathbb{R}$ une fonction continue et concave.

On introduit les notations suivantes :

- $e_T(S)$: enveloppe convexe de T, c'est-à-dire la plus grande fonction convexe qui minore T sur S.
- $E_T(S)$: enveloppe concave de T, c'est-à-dire la plus petite fonction concave qui majore T sur S.
- $\mathrm{mid}(x, y, z)$: valeur médiane parmi x, y, z.

On suppose :

$$\begin{aligned}
&\forall x \in S, \quad c(x) \leq t(x) \leq C(x),\\
&\exists a_t, b_t \in \mathbb{R} \text{ tels que } a_t \leq t(x) \leq b_t,\\
&\exists z_{\min T} \in [a_t, b_t] \text{ tel que } T(z_{\min T}) = A_T = \inf_{z \in [a_t, b_t]} T(z),\\
&\exists z_{\max T} \in [a_t, b_t] \text{ tel que } T(z_{\max T}) = B_T = \sup_{z \in [a_t, b_t]} T(z).
\end{aligned}$$

Théorème. ***[27]***

Sous les hypothèses précédentes, pour tout $x \in S \cap \{x \mid a_t \leq t(x) \leq b_t\}$, on a :

$$\begin{aligned}
T[t(x)] &\geq e_T\left(\mathrm{mid}(z_{\min T}, c(x), C(x))\right),\\
T[t(x)] &\leq E_T\left(\mathrm{mid}(z_{\max T}, c(x), C(x))\right).
\end{aligned}$$

Théorème. ***[27]*** *Soient U et V deux fonctions réelles univariées sur un intervalle $[a, b]$, bornées par :*

$$U_a \leq U(x) \leq U_b, \quad V_a \leq V(x) \leq V_b.$$

Alors, pour tout $x \in [a, b]$, les inégalités suivantes sont vraies :

$$\begin{aligned}
UV &\geq UV_b + U_bV - U_bV_b,\\
UV &\geq UV_a + U_aV - U_aV_a,\\
UV &\leq U_bV + UV_a - U_bV_a,\\
UV &\leq U_aV + UV_b - U_aV_b.
\end{aligned}$$

En conséquence :

$$UV \geq \max\left(UV_b + U_bV - U_bV_b,\ UV_a + U_aV - U_aV_a\right),$$
$$UV \leq \min\left(U_bV + UV_a - U_bV_a,\ U_aV + UV_b - U_aV_b\right).$$

Corollaire 1.[**27**] *Sous les hypothèses des deux théorèmes précédents* :

$$U(u(x)) \cdot V(v(x)) \geq \max\big(eU \circ g_u(x)\, V_b + U_b\, eV \circ g_v(x) - U_b\, V_b,$$
$$eU \circ g_u(x)\, V_a + U_a\, eV \circ g_v(x) - U_a\, V_a\big)$$

et

$$U(u(x)) \cdot V(v(x)) \leq \min\big(U_b\, EV \circ g_v(x) + EU \circ g_u(x)\, V_a - U_b\, V_a,$$
$$U_a\, EV \circ g_v(x) + EU \circ g_u(x)\, V_b - U_a\, V_b\big)$$

Le corollaire ci-dessous fournit un résultat général. Nous rappelons que dans notre cas, u et v correspondent à l'identité, ce qui simplifie les expressions précédentes :

Corollaire 2.[**27**] Sous les hypothèses des deux théorèmes précédents :

$$U(x)\cdot V(x) \geq \max\left\{eU(x)\cdot Vb + Ub\cdot eV(x) - Ub\cdot Vb,\ eU(x)\cdot Va + Ua\cdot eV(x) - Ua\cdot Va\right\}$$

et

$$U(x)\cdot V(x) \leq \min\left\{Ub\cdot EV(x) + EU(x)\cdot Va - Ub\cdot Va,\ Ua\cdot EV(x) + EU(x)\cdot Vb - Ua\cdot Vb\right\}$$

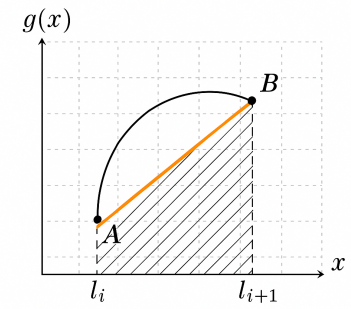
3.10.1 Application au problème posé

Dans cette partie, nous appliquons les résultats de convexification obtenus précédemment au problème $P_{N,M}$, en construisant une relaxation convexe qui pourra être utilisée comme fonction d'évaluation dans le cadre d'un algorithme de type Branch and Bound.

3.10.1.1. Cas particulier N = 3 :

Dans le cas général, on décomposerait le segment $[0, M]$ en intervalles de convexité monotone pour la fonction g. Nous en déduirions une subdivision $[l_0 = 0, a_1, \ldots, l_n = M]$, telle que g soit strictement convexe ou concave sur chaque intervalle $[l_i, l_{i+1}]$ pour tout $i \leq n - 1$.

Sur les intervalles où g est concave, elle est sa propre enveloppe concave. Sur les intervalles où elle est convexe, son enveloppe concave correspond à l'approximation affine $\beta_i x + \gamma_i$ qui passe par les points $A = (l_i, g(l_i))$ et $B = (l_{i+1}, g(l_{i+1}))$, ce qui permet de déterminer β_i et γ_i de manière unique [33].



Enveloppe convexe d'une fonction concave

Considérons tout d'abord le cas particulier où $N = 3$. Le problème factorisé de (3.7.1) consiste à déterminer trois variables entières X_1, X_2, X_3 dans $[0, M]$, telles que $\mathcal{P}_1$:

$$\text{Maximiser} \quad X_6$$

$$\text{sous les contraintes} \quad \begin{cases} X_6 = p_1 X_1 - c_1 g(X_1) X_1 + p_2 X_2 - c_2 g(X_4) X_2 + (p_3 - c_3 g(M)) X_3 \\ X_4 = X_1 + X_2 \\ X_5 = X_1 + X_2 + X_3 = M \\ 0 \leq X_i \leq M, \quad \forall i \\ X_i \in \mathbb{N}, \quad \forall i \end{cases}$$

En supposant que g est concave,donc de convexité monotone sur $[0, M]$, donc sur tous les intervalles $[a, b] \subseteq [0, M]$.
On peut utiliser le **théorème** et le **corollaire 1** précédents, on obtient notamment

les inégalités suivantes :

$$A_2 \leq c_2 H \times \min(0, M - X_2 - X_4),$$

$$A_1 \leq c_1 H \times \min(0, M - 2X_1),$$

ce qui conduit à une majoration de la fonction objectif du problème initial sous la forme :

$$f(x) \leq \sum_{i=1}^{3} p_i X_i - c_3 H X_3 + c_1 H \min(0, M - 2X_1) + c_2 H \min(0, M - X_2 - X_4).$$

Avec $g(M) = H$, où H est la constante de calibration qui est égale à 0,99.

En remplaçant la fonction objectif d'origine par cette expression dans la formulation du problème $\mathcal{P}_1$, on obtient un nouveau problème $\mathcal{P}_2$, qui constitue une **relaxation convexe** du problème initial appelé **problème relaché convexe**. Ce problème, bien que toujours défini sur des variables entières, possède une fonction objectif concave et des contraintes linéaires, ce qui le rend plus accessible au traitement algorithmique.

$$\text{Maximiser} \quad X_6$$

$$\text{sous les contraintes} \quad \begin{cases} X_6 = \sum_{i=1}^{3} p_i X_i - c_1 X_1 g(X_1) - c_3 H X_3 + c_2 H \min(0, M - X_2 - X_4) \\ X_4 = X_1 + X_2 \\ X_5 = X_1 + X_2 + X_3 = M \\ 0 \leq X_i \leq M, \quad \forall i \\ X_i \in \mathbb{N}, \quad \forall i \end{cases}$$

3.10.1.2. Cas général : N quelconque [33]

Nous nous intéressons maintenant au problème $(P_{N,M})$ avec des contraintes de bornes indépendantes pour chaque variable de décision :

$$\max \; X_{2N} = \left(\sum_{i=1}^{N} p_i X_i \right) - (c_1 g(X_1) X_1 + c_N g(M) X_N) + \sum_{i=2}^{N-1} (-c_i)[X_i g(X_{N+i-1})]$$

$$\text{s.c.} \quad \begin{cases} X_i = x_i, \quad \text{pour } i = 1, \dots, N \\ X_{N+i-1} = \sum_{j=1}^{i} X_j, \quad \text{pour } i = 2, \dots, N-1 \\ X_{2N-1} = \sum_{j=1}^{N} X_j = M \\ 0 \leq L_i \leq X_i \leq U_i \leq M, \quad \text{pour } i = 1, \dots, N \\ 0 \leq a_i \leq X_{N+i-1} \leq b_i \leq M, \quad \text{pour } i = 2, \dots, N-1 \\ X_i \in \mathbb{N}, \quad \text{pour tout } i \end{cases}$$

Par rapport au $(P_{N,M})$ standard, on ajoute les contraintes de bornes :

$$0 \leq L_i \leq X_i \leq U_i \leq M, \quad 0 \leq a_i \leq X_{N+i-1} \leq b_i, \text{ avec } a_i < b_i.$$

On définit les termes non concaves :

$$A_i = -c_i X_i g(X_{N+i-1}), \quad 2 \leq i \leq N-1.$$

Remarque : Le cas $a_i = b_i$ correspond à une variable X_{N+i-1} fixée, ce qui rend A_i linéaire. On suppose donc $a_i < b_i$ pour tout i.

Relaxation convexe :

Sous les bornes :

$$\begin{cases} 0 \leq L_i \leq X_i \leq U_i \leq M \\ 0 \leq a_i \leq X_{N+i-1} \leq b_i \leq M, \quad a_i < b_i \\ -H \leq -g(b_i) \leq -g(X_{N+i-1}) \leq -g(a_i) \leq 0 \\ V_{a_i} = -g(b_i), \quad V_{b_i} = -g(a_i) \end{cases}$$

On majore $-g(X_{N+i-1})$ par son enveloppe concave sur $[a_i, b_i]$, ce qui donne :

$$\begin{aligned} A_i &\leq c_i \min(-U_i g(X_{N+i-1}) + X_i V_{ai} - U_i V_{ai}, \\ &\quad - L_i g(X_{N+i-1}) + X_i V_{bi} - L_i V_{bi}) \\ &\leq c_i \min(U_i[\alpha_i X_{N+i-1} + \beta_i] + X_i V_{ai} - U_i V_{ai}, \\ &\quad L_i[\alpha_i X_{N+i-1} + \beta_i] + X_i V_{bi} - L_i V_{bi}) \\ &\leq c_i \min(U_i \alpha_i X_{N+i-1} + X_i V_{ai} + U_i(\beta_i - V_{ai}), \\ &\quad L_i \alpha_i X_{N+i-1} + X_i V_{bi} + L_i(\beta_i - V_{bi})) \end{aligned}$$

avec

$$\alpha_i = \frac{g(a_i) - g(b_i)}{b_i - a_i} < 0, \quad \beta_i = \frac{a_i g(b_i) - b_i g(a_i)}{b_i - a_i}.$$

On en déduit une surestimation concave de la fonction objectif f du problème factorisé :

$$f(x) \leq \left[\sum_{i=1}^{N} p_i X_i - c_1 g(X_1) X_1 + c_N g(M) \times X_N + \sum_{i=2}^{N-1} c_i \times \min \left\{ \begin{array}{l} U_i \alpha_i X_{N+i-1} - g(b_i) X_i - U_i b_i \alpha_i, \\ L_i \alpha_i X_{N+i-1} - g(a_i) X_i - L_i a_i \alpha_i \end{array} \right. \right.$$

Par conséquent, le problème majorant noté $(PRC_{N,M})$

$$\left\{ \begin{array}{rl} \max X_{2N} = & \left(\sum_{i=1}^{N} p_i X_i \right) - (c_1 g(X_1) X_1 + c_N g(M) X_N) \\ & + \sum_{i=2}^{N-1} c_i \times \min \left(U_i \frac{g(a_i) - g(b_i)}{b_i - a_i} X_{N+i-1} - g(b_i) X_i + U_i b_i \frac{g(b_i) - g(a_i)}{b_i - a_i}, \right. \\ & \left. L_i \frac{g(a_i) - g(b_i)}{b_i - a_i} X_{N+i-1} - g(a_i) X_i + L_i a_i \frac{g(b_i) - g(a_i)}{b_i - a_i} \right) \\ \text{s.c.} & X_i = x_i, \quad 1 \leq i \leq N \\ & X_{N+i-1} = \sum_{j=1}^{i} X_j, \quad 2 \leq i \leq N-1 \\ & X_{2N-1} = \sum_{j=1}^{N} X_j = M \\ & 0 \leq L_i \leq X_i \leq U_i \leq M, \quad 1 \leq i \leq N \\ & 0 \leq a_i \leq X_i \leq b_i \leq M, \quad N+1 \leq i \leq 2N-2 \\ & \forall i, \ X_i \in \mathbb{N} \end{array} \right.$$

Cas sans borne :

Pour $a_i = L_i = 0$ et $b_i = U_i = M$, on a :

$$V_{a_i} = -H, \quad V_{b_i} = 0,$$

ce qui donne finalement :

$$\max X_{2N} = \left(\sum_{i=1}^{N} p_i X_i\right) - (c_1 g(X_1) X_1 + c_N g(M) X_N) + H \times \sum_{i=2}^{N-1} c_i \min(0, M - X_{N+i-1} - X_i)$$

$$\begin{cases} \text{s.c.} \quad X_i = x_i, \quad 1 \leq i \leq N \\ \qquad X_{N+i-1} = \sum_{j=1}^{i} X_j, \quad 2 \leq i \leq N-1 \\ \qquad X_{2N-1} = \sum_{j=1}^{N} X_j = M \\ \qquad 0 \leq L_i \leq X_i \leq U_i \leq M, \quad 1 \leq i \leq N \\ \qquad 0 \leq a_i \leq X_i \leq b_i \leq M, \quad N+1 \leq i \leq 2N-2 \\ \qquad \forall i, \quad X_i \in \mathbb{N} \end{cases}$$

Remarque finale : Comme pour le cas $N = 3$, cette relaxation convexe ne fournit pas directement une bonne borne supérieure. Il faudra donc utiliser des algorithmes comme **Branch & Bound** pour atteindre l'optimum.

3.11 Linéarisation du minimum

L'objectif de cette section est de **linéariser les expressions du type** $\min(A, B)$ qui apparaissent dans le problème relaxé convexe non linéaire $(PRC_{N,M})$, présenté précédemment comme relaxation convexe du problème initial $(P_{N,M})$. L'idée est de transformer ce problème en une **formulation linéaire en nombres mixtes** (MIP), plus facile à résoudre numériquement.

Théorème (Linéarisation du minimum)**.** ***[27]*** *Soient $U_1(x), \ldots, U_k(x)$ des fonctions affines bornées sur un ensemble $S \subset \mathbb{R}^n$, avec :*

$$U_i^A \leq U_i(x) \leq U_i^B, \quad \forall x \in S, \ \forall i = 1, \ldots, k$$

Alors, poser $e = \min_{1 \leq i \leq k} U_i(x)$ est équivalent à introduire des variables binaires

$y_i \in \{0,1\}$ *et à résoudre :*

$$\begin{cases} e \leq U_i(x), \forall i \\ e \geq U_i(x) + \lambda_i y_i, \forall i \\ \sum_{i=1}^{k} y_i = k-1 \end{cases}$$

avec :

$$\lambda_i = \max_j (U_j^B - U_i^A)$$

Remarque. Cette reformulation découle directement d'un résultat connu sur la linéarisation du *maximum*, appliqué au *minimum* par symétrie.

Corollaire 3 (Billionnet)[27]

Si $U_1(x), \ldots, U_k(x)$ sont bornées sur $S \subset \mathbb{R}^N$, on peut écrire :

$$e = \min_i U_i(x)$$

en introduisant des variables binaires y_i et en écrivant le système suivant :

$$\begin{cases} e \leq U_i(x), \forall i \\ e - \left(\min_i U_i^A - U_i^B\right) y_i \geq U_i(x), \forall i \\ \sum_i y_i = k-1 \end{cases}$$

3.11.1 Application au problème $(PRC_{N,M})$ [33]

On définit pour chaque i, le terme :

$$E_i = \min\left(U_i\alpha_i X_{N+i-1} - g(b_i)X_i - U_i b_i \alpha_i,\ L_i\alpha_i X_{N+i-1} - g(a_i)X_i - L_i a_i \alpha_i\right)$$

avec $\alpha_i < 0$.

On encadre les termes pour garantir les bornes, puis par le corollaire 3, on introduit des binaires $Y_{1,i}, Y_{2,i}$:

$$\begin{cases} E_i \leq U_i\alpha_i X_{N+i-1} - g(b_i)X_i - U_i b_i \alpha_i \\ E_i \leq L_i\alpha_i X_{N+i-1} - g(a_i)X_i - L_i a_i \alpha_i \\ E_i \geq U_i\alpha_i X_{N+i-1} - g(b_i)X_i - U_i b_i \alpha_i + \lambda_{1,i} Y_{1,i} \\ E_i \geq L_i\alpha_i X_{N+i-1} - g(a_i)X_i - L_i a_i \alpha_i + \lambda_{2,i} Y_{2,i} \\ Y_{1,i} + Y_{2,i} = 1 \end{cases}$$

avec :

$$\gamma_i = \min\left(U_i\alpha_i b_i - g(b_i)U_i - U_i b_i\alpha_i,\ L_i\alpha_i b_i - g(a_i)U_i - L_i a_i\alpha_i\right)$$

et

$$\lambda_{1,i} = \gamma_i - \left(U_i\alpha_i a_i - g(b_i)L_i - U_i b_i\alpha_i\right), \quad \lambda_{2,i} = \gamma_i - \left(L_i\alpha_i a_i - g(a_i)L_i - L_i a_i\alpha_i\right)$$

3.11.2 Modélisation finale $(PL_{N,M})$ [33]

Pour linéariser le problème $(PRC_{N,M})$, nous appliquons le Corollaire 3. On obtient alors le problème linéaire mixte $(PL_{N,M})$ suivant :

$$\max X_{2N} = \left(\sum_{i=1}^{N} p_i X_i\right) - \left(c_1 g(X_1)X_1 + c_N g(M)\times X_N\right) + \sum_{i=2}^{N-1} c_i E_i$$

$$\begin{cases}
\text{s.c.} & X_i = x_i, \quad 1 \le i \le N \\
& X_{N+i-1} = \sum_{j=1}^{i} X_j, \quad 2 \le i \le N-1 \\
& X_{2N-1} = \sum_{j=1}^{N} X_j = M \\
\text{Pour } 2 \le i \le N-1 : & E_i \le U_i\alpha_i X_{N+i-1} - g(b_i)X_i - U_i b_i\alpha_i \\
& E_i \le L_i\alpha_i X_{N+i-1} - g(a_i)X_i - L_i a_i\alpha_i \\
& E_i \ge U_i\alpha_i X_{N+i-1} - g(b_i)X_i - U_i b_i\alpha_i + \lambda_{1,i}Y_{1,i} \\
& E_i \ge L_i\alpha_i X_{N+i-1} - g(a_i)X_i - L_i a_i\alpha_i + \lambda_{2,i}Y_{2,i} \\
& Y_{1,i} + Y_{2,i} = 1 \\
\text{Avec} & \alpha_i = \dfrac{g(a_i) - g(b_i)}{b_i - a_i} < 0 \\
& \gamma_i = \min\left(U_i\alpha_i b_i - g(b_i)U_i - U_i b_i\alpha_i,\ L_i\alpha_i b_i - g(a_i)U_i - L_i a_i\alpha_i\right) \\
& \lambda_{1,i} = \gamma_i - \left(U_i\alpha_i a_i - g(b_i)L_i - U_i b_i\alpha_i\right) \\
& \lambda_{2,i} = \gamma_i - \left(L_i\alpha_i a_i - g(a_i)L_i - L_i a_i\alpha_i\right) \\
& 0 \le L_i \le X_i \le U_i \le M, \quad 1 \le i \le N \\
& 0 \le a_i \le X_i \le b_i \le M, \quad N+1 \le i \le 2N-2 \\
& \forall i, \quad X_i \in \mathbb{N}, \quad Y_{1,i}, Y_{2,i} \in \{0,1\}
\end{cases}$$

Remarque finale

Bien que la linéarisation affine le problème, elle s'accompagne d'une complexité accrue. Elle nécessite souvent des méthodes comme le *Branch & Bound* pour garantir l'intégralité des solutions, mais reste une étape clé vers une résolution efficace.

Conclusion générale

Ce mémoire s'est inscrit dans une démarche approfondie d'analyse et de résolution des problèmes d'optimisation, en mobilisant une variété d'approches théoriques et méthodologiques issues de la recherche opérationnelle. L'optimisation, omniprésente dans notre quotidien comme dans les systèmes complexes, exige une compréhension fine des structures de problèmes, allant des cas convexes bien maîtrisés aux situations non convexes, plus réalistes mais aussi plus ardues.

Dans un premier volet, les fondations conceptuelles ont été posées en opposant clairement optimisation convexe et non convexe. Il a mis en évidence les limites théoriques et pratiques rencontrées dans les problèmes non convexes, notamment à travers la présence de minima locaux multiples et l'absence de garanties sur l'optimalité globale.

Le développement suivant a ensuite approfondi cette distinction dans le cadre linéaire et non linéaire, montrant que si l'optimisation linéaire bénéficie d'algorithmes efficaces comme le simplexe, l'optimisation non linéaire, plus proche des cas concrets, requiert des outils spécialisés comme la méthode Branch and Bound, les relaxations convexes ou d'autres stratégies de linéarisation.

Pour clore cette progression, la dernière partie a mis en lumière la puissance de la programmation dynamique dans la résolution de problèmes complexes, en particulier lorsqu'une structure récursive ou décomposable peut être exploitée. L'intégration des techniques de convexification et de décomposition y a joué un rôle clé, notamment pour aborder les problèmes non convexes avec plus d'efficacité.

L'ensemble de ce travail souligne que l'optimisation non convexe, bien que théoriquement et computationnellement difficile, n'est pas un domaine fermé. Au contraire, elle ouvre un champ de recherche riche, où les avancées méthodologiques,

la combinaison d'approches et l'évolution des outils algorithmiques permettent de progresser vers des solutions plus robustes, même pour les problèmes les plus complexes. Ces contributions constituent une base solide pour de futures explorations, tant sur le plan académique que dans des applications concrètes à fort enjeu.

Bibliographie

[1] Adams, R. A., *Calculus : A Complete Course*, 8th edition, Pearson, 2013.

[2] Allaire, G., *Analyse numérique et optimisation*, Dunod, 2002.

[3] Al Kharboutly, M., *Thèse : Résolution d'un problème quadratique non convexe avec contraintes mixtes par les techniques de l'optimisation D.C.*, Optimisation et contrôle [math.OC], Normandie Université, 2018.

[4] Bazaraa, M. S., Sherali, H. D., Shetty, C. M., *Nonlinear Programming*, 2nd edition, John Wiley and Sons, 1993.

[5] Bazaraa, M. S., Sherali, H. D., Shetty, C. M., *Nonlinear Programming : Theory and Algorithms*, 3rd edition, John Wiley & Sons, 2006.

[6] Bellman, R., *Dynamic Programming*, Princeton University Press, 1957.

[7] Bergounioux, M., *Optimisation et contrôle des systèmes linéaires*, Dunod, Paris, 2001.

[8] Bertsekas, D., *Dynamic Programming and Optimal Control*, Vol. I, 4th edition, Athena Scientific, 2017.

[9] Bonnans, J.-F., Gilbert, J.-C., Lemaréchal, C., Sagastizábal, C., *Optimisation Numérique : Aspects théoriques et pratiques*, Springer, 1997.

[10] Borwein, J., Lewis, A., *Convex Analysis*, Springer, 2006.

[11] Boudiaf, N., *Optimisation sans contraintes : aspect théorique et algorithmique. Cours et exercices corrigés*, 2017.

[12] Boucherit, S., *Cours d'optimisation*, année universitaire 2017–2018, Université 8 Mai 1945 Guelma.

[13] Boyd, S., Vandenberghe, L., *Convex Optimization*, Cambridge University Press, 2004.

[14] Boyd, S., Vandenberghe, L., *Convex Optimization*, Cambridge University Press, 2004. (Séparabilité : Sec. 4.2.3)

[15] Boyer, V., *Contribution à la programmation en nombre entier*, Thèse, INSA de Toulouse, 2007.

[16] Cohen, G., *Convexité et optimisation*, École nationale des ponts et chaussées, 2000.

[17] Cormen, T. et al., *Introduction to Algorithms*, MIT Press, 2009.

[18] El Ouafdi, A. F., *Mémoire : Méthodes primales-duales pour la programmation non linéaire non convexe*, Mémoire de maîtrise, Université de Sherbrooke, Canada, 2003.

[19] Haddad, C., *Mémoire : Systèmes dynamiques : applications en dynamique des populations*, 2023.

[20] Jain, P., Kar, P., *Non-convex Optimization for Machine Learning*, 2017.

[21] Kessi, K., Sidennas, D., *Mémoire : Optimisation non linéaire discrète*, Université Mouloud Mammeri de Tizi-Ouzou (UMMTO), s.d.

[22] Kerboub, R., *Mémoire : Introduction sur les problèmes semi-définis positifs*, Université de Guelma, s.d.

[23] Li, Y., *Introduction à l'optimisation non convexe*, Université Carnegie Mellon, s.d.

[24] Leslous, F., *Problème d'optimisation non convexe et optimisation DC*, Thèse, Université Mouloud Mammeri de Tizi-Ouzou (UMMTO), 2023.

[25] Leslous, F., Goubi, M., Ouanes, M., *A new approach for non-convex optimisation problems applied to Hump and benchmark functions*, *Journal of Applied Mathematics and Computation*, vol. 25, no 3, 2023.

[26] Leslous, F., Ouanes, M., *Determining a global optimum of a Nonconvex function in $\mathbb{R}^n$ Box*, Conférence internationale, Paris-Saclay, 2019.

[27] McCormick, G. P., "Computability of Global Solutions to Factorable Nonconvex Programs : Part I—Convex Underestimating Problems", *Mathematical Programming*, vol. 10, no. 1, pp. 147–175, 1976.

[28] Minoux, M., *Programmation mathématique*, Dunod, 1983.

[29] Minoux, M., *Programmation mathématique : Théorie et algorithmes*, 2e édition, Dunod, 2008.

[30] Mikhalevich, V. S., Gupal, A. M., Norkin, V. I., *Methods of Non-convex Optimization*, Moscow, 1987.

[31] Nocedal, J., Wright, S. J., *Numerical Optimization*, 2nd edition, Springer, 2006.

[32] Nocedal, J., Wright, S. J., *Numerical Optimization*, Springer, 2006. (Problème bien posé : Sec. 12.3)

[33] Nizard, D., *Programmation mathématique non convexe non linéaire en variables entières : un exemple d'application au problème de l'écoulement de larges blocs d'actifs*, Thèse, Université Paris-Saclay, 2023.

[34] Ouali, N., *Mémoire : Optimisation non linéaire et application*, année universitaire 2017–2018, Université Mouloud Mammeri, Tizi-Ouzou.

[35] *Optimisation avec contraintes (Chapitre 3)*, Support de cours, Université de Guelma, 2021.

[36] Picard, R. T., *Convexité et applications*, Notes de cours, Université de Rennes 1, 2004.

[37] Powell, W. B., *Approximate Dynamic Programming*, Wiley, 2007.

[38] Rairon, *Une méthode d'optimisation non linéaire en variables mixtes pour la conception de procédés*, *Operations Research*, 1988.

[39] Rockafellar, R. T., *Convex Analysis*, Princeton University Press, 1970. (Convexité : Th. 5.1)

[40] Royer, J., *Calcul différentiel et intégral*, Université Toulouse 3, 2013–2014.

[41] Rosen, J. B., *The Gradient Projection Method for Nonlinear Programming. Part I : Linear Constraints*, *Journal of the Society for Industrial and Applied Mathematics*, vol. 8, no. 1, pp. 181–217, 1960.

[42] Rosen, J. B., *The Gradient Projection Method for Nonlinear Programming. Part II : Nonlinear Constraints*, *SIAM Journal on Applied Mathematics*, vol. 9, no. 3, pp. 514–532, 1961.

[43] Sanghavi, S., Netrapalli, P., *Understanding Non-convex Optimization*, Microsoft Research, 2016.

[44] Stewart, J., *Calculus : Early Transcendentals*, 8th edition, Cengage Learning, 2015.

[45] Trystram, D., *Algorithmique avancée : Branch and Bound*, Presses Universitaires de Grenoble, 2000.