

N° d'ordre:

RÉPUBLIQUE ALGERIENNE DÉMOCRATIQUE ET POPULAIRE
MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE
 UNIVERSITÉ MOULOUD MAMMERI DE TIZI OUZOU
FACULTÉ DES SCIENCES
DÉPARTEMENT DE MATHÉMATIQUES
LABORATOIRE LMPA

MÉMOIRE DE MASTER

Domaine : LMD Mathématiques-informatique
Filière : Mathématiques
Spécialité : Recherche Opérationnelle

Par

M^{LLE} SOUHILA BEN MOHAND SAID

OPTIMISATION GLOBALE : ALGORITHMES ET APPLICATIONS

Soutenu le 23-07-2023 devant le jury :

Dr.	TALEM DJAMEL	UMMTO	Président du jury
Pr.	OUANES MOHAND	UMMTO	Rapporteur
Dr.	ACHEMINE FARIDA	UMMTO	Examinatrice

Année Universitaire : 2022 - 2023

Remerciements

Il m'est agréable d'adresser mes remerciements à mon promoteur Monsieur Ouanes Mohand, professeur à l'UMMTO, pour l'honneur qu'il me fait, en assurant la direction et le suivi scientifique et technique du présent mémoire. Je le remercie pour sa grande contribution à l'aboutissement de ce travail, ainsi que pour sa disponibilité, tout au long de cette étude, malgré ses nombreuses activités. Qu'il retrouve ici ma profonde gratitude.

Je remercie vivement, Monsieur Talem, Maître de conférences à l'UMMTO, pour l'honneur qu'il me fait en acceptant de présider le jury de ce mémoire.

J'adresse mes vifs remerciement à Madame Achemine, Maître de conférences à l'UMMTO, d'avoir bien voulu faire partie de ce jury, et d'avoir pris le temps de lire et de jugé ce travail. Je remercie tous ceux qui ont contribués de près ou de loin à la réalisation de ce modeste travail.

Je ne peux pas me résoudre de remercier mes parents, ainsi que tous les membres de ma famille, pour leur soutien et encouragement quasi-quotidien.

Dédicaces

Je remercie le bon dieu de m'avoir donner la force d'accomplir ce travail pour aller plus loin.
Je dédie ce modeste travail, à celle qui a bercé mes rêves ma très chère mère ; à celui qui sans son aide je ne serais peut-être pas à ce niveau mon cher père.

À mes soeurs Assia, Saloua, Djouher. À mes deux frères, Zouhir, Tayeb, pour leurs encouragements tout au long de mes études.

À ma deuxième maman et ma chère adorée ma tante Hammama.

À la mémoire de mes grands parents que dieu les protège dans son vaste paradis.

À toute la famille Harouni, et spécialement à mon beau frère khellaf.

À toute la famille Ben Mohand Saïd, et à tous ceux qui m'aiment et que j'aime de près et de loin.

Table des matières

Table des matières	ii
Table des figures	iv
Liste des tableaux	v
Introduction générale	1
1 Rappels sur l'optimisation convexe et non convexe	4
1.1 Introduction	4
1.2 Ensembles convexes et non convexes	4
1.2.1 Définitions d'ensembles convexes et non convexes [1],[2],[3],[9] . . .	4
1.2.2 Propriétés des ensembles convexes [1],[2],[3],[11]	7
1.2.3 Projection sur un convexe fermé [2],[3]	9
1.2.4 Hyperplan d'appui, fonction d'appui [2],[3]	10
1.2.5 Théorème de la séparation par des hyperplans affines [2],[3],[8] . . .	12
1.3 Fonctions convexes et non convexes	14
1.3.1 Définitions de fonctions convexes, non convexes (concaves)[1],[2],[3],[9, 10]	14
1.3.2 Interprétation géométrique [3]	17
1.3.3 Propriétés des fonctions convexes et concaves [1],[2],[3]	19
1.3.4 Critère d'ordre (01) et d'ordre (02) de convexité [3]	23
1.3.5 Gradient-Différentiel [2],[3]	23
1.3.6 Sous-gradients. Sous-différentiel [2],[3]	25
1.3.7 Sous-gradient et dérivées directionnelles [1],[2],[3]	26
1.3.8 Caractérisation de la convexité [12]	27
1.3.9 Fonctions convexes "étendues" [1]	30
1.4 Programmes convexes et non convexes	30
1.4.1 Programme convexe [1],[2]	30
1.4.2 Programme non convexe [1]	31

1.5	Conditions d'optimalités	31
1.5.1	Condition d'optimalité du 1 ^{er} ordre [2],[9],[13]	31
1.5.2	Condition d'optimalité du 2 nd ordre [2],[13]	35
1.6	Conclusion	37
2	Optimisation globale	39
2.1	Introduction	39
2.2	Définition d'optimisation globale [3], [4], [6]	39
2.2.1	Optimum local-global [6]	40
2.3	Méthodes d'optimisation globale	40
2.3.1	Méthodes déterministes (exactes) [7], [19], [20]	41
2.3.2	Méthodes stochastiques [7], [17], [27]	55
2.3.3	La méthode Branch and Bound avec une fonction borne inférieure quadratique [21]	61
2.3.4	La méthode de Branch and Bound avec une nouvelle fonction borne inférieure pour les problèmes d'optimisation globale univariés [21], [25], [37], [38]	71
2.3.5	Les méthodes hybrides [19], [39]	80
2.4	Conclusion	82
3	Simulations numérique	83
3.1	Introduction	83
3.2	Qu'est-ce-que Matlab? [58]	83
3.3	Simulation numérique	84
3.3.1	Simulation numérique sur la méthode Branch and Bound avec une fonction borne inférieure quadratique [21]	84
3.3.2	Simulation numérique sur la méthode Branch and Bound avec une nouvelle fonction borne inférieure [21], [37], [59]	87
3.4	Conclusion	94
	Annexes	103

Table des figures

1.1	Convexité d'un ensemble	5
1.2	Enveloppe convexe.	6
1.3	Ensemble convexe, hyperplan d'appuis et demi espace.	11
1.4	Fonction convexe.	15
1.5	Fonction concave.	16
1.6	Fonction ni convexe ni concave.	17
1.7	Le gradient de la fonction $f(x, y) = (\cos^2 x + \cos^2 y)^2$ représenté dans le plan bas.	24
1.8	Sous-différentiel.	26
2.1	Procédure de Tunneling.	42
2.2	Principe de l'algorithme IBBA.	47
2.3	La fonction borne inférieure de f, cas 1.	64
2.4	La fonction borne inférieure de f, cas 2.	64
2.5	L'ensemble faisable de problème (PC) après la 1 ^{ère} itération $[a_0^1, b_0^2]$. . .	70
2.6	Instruction Commande Window pour avoir la comparaison entres les diffé- rentes fonctions borne inférieure de la fonction $y = \sin(x)$	76
2.7	Comparaison entre les différentes fonctions borne inférieure de la fonction $\sin(x)$	77
2.8	Les instruction commande window pour avoir le graphe de la fonction sous Matlab	78
2.9	La fonction $\sin(s)$ sous Matlab.	78
3.1	Fenêtre principale de Matlab 2007.	84
3.2	Instruction pour tracer le graphe de $b''(s)$ sous Matlab (Command Window)	88
3.3	La représentation graphique de $b''(s)$ sous Matlab.	88
3.4	Instruction pour tracer le graphe de $b''(s)$ sous Matlab (Command Window)	90
3.5	La représentation graphique de $g''(s)$ sous Matlab.	90
3.6	Tableau qui prouve l'efficacité de la méthode BB avec une nouvelle fonction borne inférieure par rapport à la méthode α BB.	93

3.7	1 ^{ère} itération de la méthode Branch and Bound avec une fonction borne inférieure quadratique simulée sous Matlab.	103
3.8	2 ^{ème} itération de la méthode Branch and Bound avec une fonction borne inférieure quadratique simulée sous Matlab.	106
3.9	3 ^{ème} itération de la méthode Branch and Bound avec une fonction borne inférieure quadratique simulée sous Matlab.	108

Liste des tableaux

3.1	Tableau de signe de $g''(s)$	90
3.2	Tableau de comparaison entre la méthode de BB avec une nouvelle fonction borne inférieure et la méthode α BB.	93

Introduction générale

Dans notre vie quotidienne, l'optimisation est toujours présente. On cherche tous à optimiser notre temps de travail ou d'études. Ou encore, le trajet que nous aurons à parcourir pour nous rendre quelque part. En industrie, on cherche continûment, à optimiser le processus de production ; de plus en informatique (plus exactement en algorithmique), on vise fréquemment à optimiser le temps d'exécution et le nombre d'instructions ... d'un algorithme, et beaucoup d'autres exemples d'applications d'actualité de l'optimisation. Donc, en générale, on cherche à trouver cette meilleure solution.

L'optimisation : vient du mot optimiser qui veut dire maximiser ou minimiser un profit (gain)/perte, d'un ou plusieurs objectif(s), avec ou sans contraintes.

L'optimisation locale s'applique pour la résolution des problèmes convexes. Elle cherche à trouver la meilleure solution localement, c'est à dire dans ce voisinage aucune solution n'est meilleure qu'elle ; en utilisant des méthodes locales (la méthode de Newton Raphson, la méthode de gradient...).

L'optimisation globale s'applique pour la résolution des problèmes non convexes. Elle consiste à trouver la meilleure solution du domaine tout entier ; c'est à dire, sur ce domaine il n'existe aucune autre solution meilleure qu'elle en respectant toutes les contraintes ; en utilisant des méthodes globales (la méthode de Branch and Bound, la méthode de recuit simulé...). Cette solution est appelée l'optimum global.

L'optimisation globale [3], [4], [6] attire l'attention des chercheurs non seulement par-ce-qu'elle survienne dans plusieurs applications réelles. Mais

aussi, car les méthodes de ces problèmes sont utiles pour les extensions aux cas multi-variables, ou en réduisant le cas multidimensionnel au cas unidimensionnel.

Dans nos jours l'optimisation globale, a connu une avancée fulgurante, en ce sens plusieurs papiers ont été développés [54, 57]..., pour trouver la meilleure solution dans le cadre de l'optimisation (programmation mathématique).

Dans ce mémoire, nous nous interrogeons sur les notions d'optimisation convexe locale, et pourquoi le passage à l'optimisation globale ? Pourquoi l'optimisation globale en général ? et comment choisir la meilleure méthode à appliquer pour chaque problème rencontré ? Comment hybrider ces méthodes, et avoir des algorithmes plus efficaces ?

Dans ce travail, nous nous sommes intéressés à donner des généralités sur l'optimisation convexe et non convexe. Citer les différentes méthodes, algorithmes, principes et théorèmes ... en optimisation globale ; en prouvant la convergence et l'efficacité de ces algorithmes à travers des simulations numériques.

Le travail accompli est présenté à travers les différents chapitres.

Dans le premier chapitre, nous allons donner des généralités sur l'optimisation convexe et non convexe [1, 3], [8, 13], (définition, propriétés, théorèmes ...), des ensembles, fonctions et programmes, ainsi que les conditions d'optimalités de 1^{er} et de 2nd ordre.

Dans le deuxième chapitre nous présenterons, les concepts et les méthodes de l'optimisation globale, déterministes [7], [19], [20] (Tunnel [7], [14, 15], [17, 18], Branch and Bound [4, 6], Branch and Bound par intervalles [39], [46] ...), stochastiques [7], [17], [27] (recuit simulé [28, 30], [33, 36], algorithmes génétiques [28, 32], recherche Tabou [29], [30], [47], ...), hybrides [19], [39] (algorithme génétique d'intervalle hybride [39]). En définissant, expliquant le principe, l'algorithme des différentes approches, suivie par des théorèmes et des preuves de convergence des algorithmes proposés.

Le troisième chapitre, sera consacré à faire des simulations numériques,

afin de prouvé l'efficacité de la méthode Branch and Bound avec une fonction borne inférieure quadratique [21]. Comparer l'algorithme Branch and Bound avec une nouvelle fonction borne inférieure [37], par rapport à l'algorithme proposé dans l'article [59], explicité ses calculs de ses différentes itérations, et prouvé son efficacité et sa rapidité de convergence vers la solution optimale.

On terminera notre travail, par une conclusion générale et des perspectives.

Rappels sur l'optimisation convexe et non convexe

1.1 Introduction

L'optimisation est la recherche d'un optimum (maximum ou minimum) sur un ensemble donné. Elle est découpée en plusieurs sous disciplines ; notamment l'optimisation sans contraintes et l'optimisation avec contraintes. Dans la première , la recherche de l'extremum se fera sur $\mathbb{R}$, plus généralement sur $\mathbb{R}^n$. Dans la seconde la recherche de l'optimum, se fera sur un domaine bien précis. Dans ces deux sous disciplines, on abordera les notions de la convexité et non convexité, qui est l'objet du premier chapitre qui sera dédié aux rappels de ces notions.

1.2 Ensembles convexes et non convexes

1.2.1 Définitions d'ensembles convexes et non convexes [1],[2],[3],[9]

Définition 1.2.1 *Un ensemble $S \subset \mathbb{R}^n$ est dit **convexe** si et seulement si :*

$$\forall x \in S, \forall y \in S \text{ et } \forall \lambda \in [0, 1] \Rightarrow \lambda x + (1 - \lambda)y \in S. \quad (\mathbf{A})$$

*D'une façon équivalente, on peut dire que S est **convexe** $\Leftrightarrow \forall x, y \in S$, le segment $[x, y]$ est tout entier contenu dans S . (cf.Figure 1.1) **(B)**.*

*Plus généralement, étant donné p points de $\mathbb{R}^n$ ($x^1, x^2, \dots, x^p$) on dit que $x \in \mathbb{R}^n$ est **une combinaison convexe** de ces p points, s'il $\exists$ des coefficients*

$\mu_1, \mu_2, \dots, \mu_p$ ($\mu_i \geq 0, \forall i = 1, 2, \dots, p$) tels que :

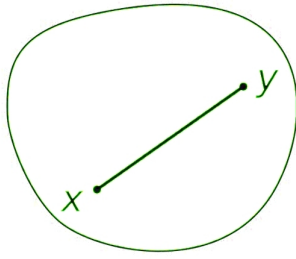
$$\sum_{i=1}^p \mu_i = 1 \text{ et } x = \sum_{i=1}^p \mu_i x^i \quad (\mathbf{C})$$

On vérifie aisément qu'un ensemble $S \subset \mathbb{R}^n$ est convexe si et seulement si tout point, combinaison convexe de points de S , est dans S .

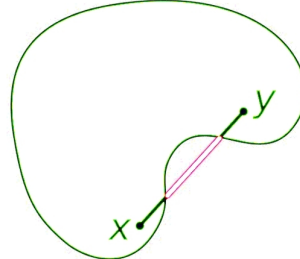
Remarque 1.2.1 Par convention, $\emptyset$ est convexe.

Exemple 1.2.1 Le sous espace vectoriel, les boules (ouvertes ou fermées) pour n'importe quelle norme, sont des parties convexes. Les parties convexes de $\mathbb{R}$ sont des intervalles.

Définition 1.2.2 Un ensemble S est dit **non convexe** s'il ne vérifie pas les définitions $(\mathbf{A}), (\mathbf{B}), (\mathbf{C})$ données dans la **définition (1.1)**.



Ensemble convexe



Ensemble non convexe

FIGURE 1.1 – Convexité d'un ensemble

Définition 1.2.3 L'enveloppe convexe de S : est le plus petit ensemble convexe contenant S . On le note par $\text{Conv}(S)$. (cf. Figure 1.2)

Il est aussi défini comme étant l'intersection de toutes les parties convexes de $\mathbb{R}^n$ qui contiennent S .

Remarque 1.2.2 Il en découle de ce qui précède que S est convexe si et seulement si

$$S = \text{Conv}(S).$$

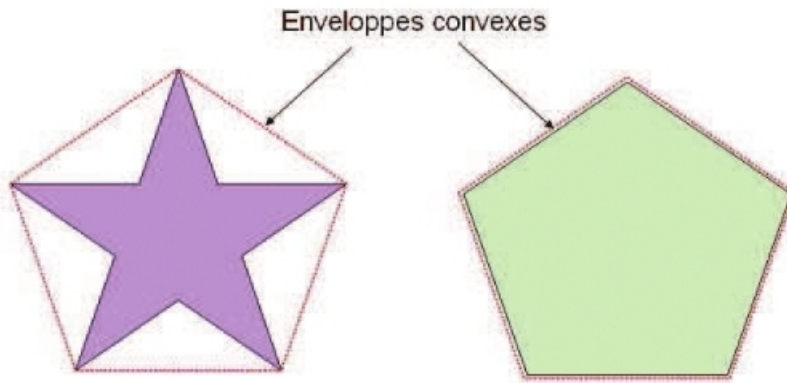


FIGURE 1.2 – Enveloppe convexe.

Il y a deux manières de construire un enveloppe convexe :

$\text{Conv}(S)$ = l'intersection de tous les convexe C contenant S .

$\text{Conv}(S)$ = l'ensemble de toutes les combinaisons convexes d'éléments de S .

Définition 1.2.4 *L'enveloppe convexe fermé de S : est le plus petit ensemble convexe fermé contenant S . On le note par $\overline{\text{Conv}(S)}$.*

Il y a aussi deux manières de construire un enveloppe convexe fermé :

$\overline{\text{Conv}(S)}$ = L'intersection de tous les convexes fermés contenant S .

$\overline{\text{Conv}(S)}$ = L'adhérence de $\text{Conv}(S)$.

Définition 1.2.5 *On appelle X un sommet (ou point extrême) de S , si X ne peut pas s'écrire comme combinaison convexe de deux points de S différents de X .*

Définition 1.2.6 *l'intérieure relatif d'un ensemble convexe $S \subset \mathbb{R}^n$, est l'intérieure de l'ensemble S relativement à la plus petite variété affine qui le contient. Il est noté par $\text{rint}(S)$.*

Définition 1.2.7 *Une partie F de S est appelée face ou partie extrême de S lorsque les propriétés suivantes sont vérifiées :*

$(x_1, x_2) \in S \times S$ et $\exists \alpha \in]0, 1[$ tel que $\alpha x_1 + (1 - \alpha)x_2 \in F \Rightarrow [x_1, x_2] \subset F$.

1.2.2 Propriétés des ensembles convexes [1],[2],[3],[11]

En pratique, on utilise généralement la notion d'ensembles convexes ; et

rarement celle d'ensembles non convexes. Pour cela, dans cette sous-section on n'abordera que les propriétés des ensembles convexes.

Théorème 1.2.1 *S est convexe si et seulement si : toute combinaison convexe de points de S appartient à S .*

Démonstration 1.2.1 *S est convexe $\Leftrightarrow \sum_{i=1}^n \lambda_i X_i \in S, \sum_{i=1}^n \lambda_i = 1, \lambda_i \geq 0, \forall i = 1, \dots, n \forall n \in \mathbb{N}^*$.*

1) $\Leftarrow$ *pour $n = 2$ c'est exactement la définition de la convexité donc S est convexe.*

2) $\Rightarrow$ *Par récurrence :*

Pour $n = 2$ c'est vérifié car S est convexe.

On suppose qu'elle est vraie à l'ordre n c'est à dire :

$$\sum_{i=1}^n \lambda_i X_i \in S, \sum_{i=1}^n \lambda_i = 1, \lambda_i \geq 0, i = 1, \dots, n;$$

et on la montre à l'ordre $n+1$ i.e. :

$$\sum_{i=1}^{n+1} \lambda_i X_i \in S, \sum_{i=1}^{n+1} \lambda_i = 1, \lambda_i \geq 0, i = 1, \dots, n+1.$$

$$\sum_{i=1}^{n+1} \lambda_i X_i = \sum_{i=1}^n \lambda_i X_i + \lambda_{n+1} X_{n+1}.$$

Cas (01) : $\lambda_{n+1} = 0$.

$$\sum_{i=1}^{n+1} \lambda_i X_i = \sum_{i=1}^n \lambda_i X_i \in S. \text{ (Par hypothèse de récurrence).}$$

Cas (02) : $\lambda_{n+1} \neq 0$.

$$\begin{aligned} \sum_{i=1}^{n+1} \lambda_i X_i &= \sum_{i=1}^n \lambda_i X_i + \lambda_{n+1} X_{n+1} = \sum_{j=1}^n \lambda_j \frac{\sum_{i=1}^n \lambda_i X_i}{\sum_{j=1}^n \lambda_j} + \lambda_{n+1} X_{n+1} \\ &= \sum_{j=1}^n \lambda_j \sum_{i=1}^n \frac{\lambda_i}{\sum_{j=1}^n \lambda_j} X_i + \lambda_{n+1} X_{n+1}. \end{aligned}$$

On pose $\mu_i = \frac{\lambda_i}{\sum_{j=1}^n \lambda_j}, \forall i = 1, \dots, n, \mu_i \geq 0, \forall i = 1, \dots, n$.

$$\sum_{i=1}^n \mu_i = \sum_{i=1}^n \frac{\lambda_i}{\sum_{j=1}^n \lambda_j} = \frac{\sum_{i=1}^n \lambda_i}{\sum_{j=1}^n \lambda_j} = \frac{\lambda_1 + \lambda_2 + \dots + \lambda_n}{\lambda_1 + \lambda_2 + \dots + \lambda_n} = 1.$$

$\sum_{i=1}^n \mu_i X_i$ est une combinaison convexe des points $(x_1, \dots, x_n)$ (n points) ,
donc :

$$\sum_{i=1}^n \mu_i X_i \in S \text{ (par hypothèse de récurrence).}$$

$$\text{On pose } Y = \sum_{i=1}^n \mu_i X_i.$$

$$\begin{aligned} \sum_{i=1}^{n+1} \lambda_i X_i &= \sum_{i=1}^n \lambda_i X_i + \lambda_{n+1} X_{n+1} \\ &= \sum_{j=1}^n \lambda_j \sum_{i=1}^n \mu_i X_i + \lambda_{n+1} X_{n+1} \\ &= \sum_{j=1}^n \lambda_j Y + \lambda_{n+1} X_{n+1} \end{aligned}$$

$$Y, X_{n+1} \in S, \sum_{j=1}^n \lambda_j \geq 0, \lambda_{n+1} \geq 0.$$

$$\sum_{j=1}^n \lambda_j + \lambda_{n+1} = \sum_{j=1}^{n+1} \lambda_j = 1$$

$$\sum_{i=1}^{n+1} \lambda_i X_i = \sum_{j=1}^n \lambda_j Y + \lambda_{n+1} X_{n+1} \text{ est une combinaison convexe des points } Y$$

$$\text{et } X_{n+1} \text{ comme } S \text{ est convexe, alors } \sum_{i=1}^{n+1} \lambda_i X_i = \sum_{j=1}^n \lambda_j Y + \lambda_{n+1} X_{n+1} \in S.$$

$$\underline{\text{Conclusion}} : \sum_{i=1}^n \lambda_i X_i \in S, \forall n \in \mathbb{N}^*, \sum_{i=1}^n \lambda_i = 1, \lambda_i \geq 0, i = 1, \dots, n.$$

• La stabilité de la convexité par certaines opérations sur les ensembles :

Théorème 1.2.2 Si S_1 et S_2 sont des convexes de $\mathbb{R}^n$, il est de même pour $S_1 + S_2$.

Démonstration 1.2.2 Montrons que $S_1 + S_2$ est un convexe, c'est à dire : $\forall z_1, z_2 \in S_1 + S_2, \forall \lambda \in [0, 1] : \lambda z_1 + (1 - \lambda) z_2 \in S_1 + S_2$, avec :
($z_1 = x_1 + y_1$ et $z_2 = x_2 + y_2$).

$$S_1 \text{ est convexe} \Leftrightarrow \forall x_1, x_2 \in S_1, \forall \lambda \in [0, 1] : \lambda x_1 + (1 - \lambda) x_2 \in S_1.$$

$$S_2 \text{ est convexe} \Leftrightarrow \forall y_1, y_2 \in S_2, \forall \lambda \in [0, 1], \lambda y_1 + (1 - \lambda) y_2 \in S_2.$$

$$\Rightarrow \lambda(x_1 + y_1) + (1 - \lambda)(x_2 + y_2) \in S_1 + S_2.$$

$$\Rightarrow \lambda z_1 + (1 - \lambda)z_2 \in S_1 + S_2 \Rightarrow S_1 + S_2 \text{ est convexe.}$$

Théorème 1.2.3 Si S_1 est un convexe de $\mathbb{R}^p$ et S_2 un convexe de $\mathbb{R}^q$, alors $S_1 \times S_2$ est un convexe de $\mathbb{R}^p \times \mathbb{R}^q$.

Démonstration 1.2.3 La démonstration se fera de la même façon que la démonstration de (Théorème 1.2.2).

Théorème 1.2.4 Soient $S_i, i = 1, \dots, n$ convexes, alors : $\bigcap_{i=1}^n S_i$ est convexe.

Démonstration 1.2.4 Soit S_i convexe ($i = 1, \dots, n$) ; montrons que $\bigcap_{i=1}^n S_i$ est convexe .i.e : $\forall X_1, X_2 \in \bigcap_{i=1}^n S_i, \forall \lambda \in [0, 1], \lambda X_1 + (1 - \lambda)X_2 \in \bigcap_{i=1}^n S_i$.

$$X_1, X_2 \in \bigcap_{i=1}^n S_i \Rightarrow X_1, X_2 \in S_i, \forall i = 1, \dots, n.$$

$$\lambda X_1 + (1 - \lambda)X_2 \in S_i, \forall i, \text{ car } S_i \text{ est convexe } \forall i = 1, \dots, n.$$

$$\Rightarrow \lambda X_1 + (1 - \lambda)X_2 \in \bigcap_{i=1}^n S_i.$$

Conclusion : $\bigcap_{i=1}^n S_i$ est convexe.

Théorème 1.2.5 Tout sous-ensemble convexe non vide de $\mathbb{R}^n$ a un intérieur relatif non vide.

Le théorème a une autre version plus simple figurant en [11]

Démonstration 1.2.5 Pour la preuve de théorème voir [11].

1.2.3 Projection sur un convexe fermé [2],[3]

Définition 1.2.8 Soit $X_0 \notin S$ convexe fermé. On définit la projection de X_0 sur S par $P_{X_0} \in S$, tel que :

$$\|P_{X_0} - X_0\| = \min_{X \in S} \|X - X_0\| \leq \|X - X_0\|, \forall X \in S.$$

Remarque 1.2.3 La projection est unique.

Définition 1.2.9 P_{X_0} est une projection de X_0 sur S convexe fermé si :

$$\langle X - P_{X_0}, X_0 - P_{X_0} \rangle \leq 0, \forall X \in S.$$

Géométriquement :

$$\langle X - P_{X_0}, X_0 - P_{X_0} \rangle = \|X - P_{X_0}\| \|X_0 - P_{X_0}\| \cos(X - P_{X_0}, X_0 - P_{X_0}) \leq 0,$$

donc :

$$\cos(X - P_{X_0}, X_0 - P_{X_0}) \leq 0.$$

$$\frac{\pi}{2} \leq (X - P_{X_0}, X_0 - P_{X_0}) \leq \pi.$$

C'est un angle obtus.

$$\begin{cases} \min f(X) \\ X \in S \quad \text{CNS} \quad \nabla f(X) = 0 \end{cases}$$

→ On calcule la projection de $\bar{X}$ sur S

$$\min_{X \in S} \|X - \bar{X}\|$$

Si S est un box :

$$S = [L_i, U_i]^n$$

$$\bar{X} \in \mathbb{R}^n (\bar{X} = (\bar{X}_1, \bar{X}_2, \dots, \bar{X}_n))$$

Pour trouver $P_{\bar{X}} = (\bar{X}_{1p}, \bar{X}_{2p}, \dots, \bar{X}_{np})$

$$\bar{X}_{ip} = \begin{cases} L_i & \text{si } \bar{X}_i < L_i \\ \bar{X}_i & \text{si } L_i < \bar{X}_i < U_i \\ U_i & \text{si } \bar{X}_i \geq U_i \end{cases}$$

1.2.4 Hyperplan d'appui, fonction d'appui [2],[3]

1.2.4.1 Hyperplan d'appui

Définition 1.2.10 On définit un hyperplan par :

$$H = \{X \in \mathbb{R}^n / \langle C, X \rangle = \alpha, \quad C \in \mathbb{R}^n, \quad \alpha \in \mathbb{R}\}.$$

Exemple 1.2.2 Dans $\mathbb{R}^2$, H est une droite ; dans $\mathbb{R}^3$ c'est est un plan.

Définition 1.2.11 On définit un demi-espace par :

$$E = \{X \in \mathbb{R}^n / \langle C, X \rangle \leq (\geq) \alpha\}. \quad (\text{cf. Figure 1.3})$$

Définition 1.2.12 Un hyperplan affine de $\mathbb{R}^n$, d'équation $\langle s, X \rangle = r$, est appelé hyperplan d'appui au convexe S en $\bar{X} \in S$ lorsque :

$$\langle s, \bar{X} \rangle = r \text{ et } \langle s, X \rangle \leq r \text{ pour tout } X \in S.$$

Autrement dit, H est appelé un hyperplan d'appui au convexe S au point $\bar{X}$ si tous les points de S se trouvent de même côté de H . (cf. Figure 1.3)

Géométriquement :

H coïncide avec l'hyperplan tangent au point $\bar{X}$ quand la fonction définissant S est différentiable en $\bar{X}$ et il est unique .

$$H = \{X \in \mathbb{R}^n / \langle (X - \bar{X}), \nabla f(\bar{X}) \rangle = 0\}.$$

Si f n'est pas différentiable en $\bar{X}$ alors, il existe une infinité d'hyperplans d'appuis.

Résultat important : Si S est un convexe et si $\bar{X} \in \text{fr}S$, alors il ya au moins un hyperplan d'appui à S en $\bar{X}$.

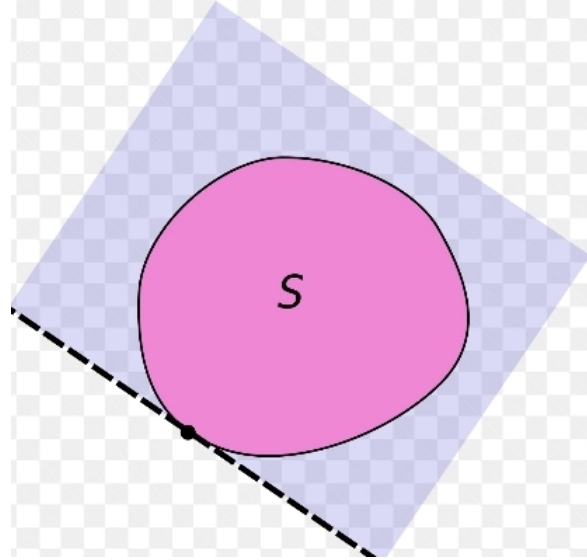


FIGURE 1.3 – Ensemble convexe, hyperplan d'appuis et demi espace.

L'ensemble convexe (S) (en rose), hyperplan d'appui de S (ligne pointillée), qui s'appuie et délimite un demi-espace qui contienne S (en bleu clair).

1.2.4.2 Fonction d'appui

Définition 1.2.13 Soit $S \subset \mathbb{R}^n \neq \emptyset$. On définit une fonction d'appui de S , la fonction notée par σ_S tel que $\sigma_S : \mathbb{R}^n \rightarrow \mathbb{R} \cup +\infty$ de la manière suivante :

$$d \in \mathbb{R}^n \mapsto \sigma_S(d) := \sup_{s \in S} \langle s, d \rangle .$$

σ_S est partout finie sur $\mathbb{R}^n$ si, et seulement si, S est borné.

Une fonction d'appui ne sait pas distinguer S de son enveloppe convexe

fermé : $\sigma_S = \sigma_{\overline{\text{conv}S}}$.

La version "fonctionnelle" de $\overline{\text{conv}S}$ peut se voir à partir de S -ou plutôt à partir de σ_S -de la manière suivante :

$$(s \in \overline{\text{conv}S}) \Leftrightarrow (\langle s, d \rangle \leq \sigma_S(d) \text{ pour tout } d \in \mathbb{R}^n).$$

Elle est de résultat "ensembliste" de la description de $\overline{\text{conv}S}$ suivant : $\overline{\text{conv}S}$ est l'intersection de tous les demi-espaces affines fermés contenant S .

Parmi les règles de calcul concernant les fonctions d'appuis : Si S_1 et S_2 sont deux parties non vides de $\mathbb{R}^n$, alors

$$\sigma_{S_1+S_2} = \sigma_{S_1} + \sigma_{S_2}.$$

Propriétés 1.2.1 de fonctions d'appuis : Soit f une fonction différentiable convexe. On a toujours :

$$f(X) \geq f(X_0) + \langle \nabla f(X_0), (X - X_0) \rangle, \forall X \in \mathbb{R}^n.$$

$$f_{ap(X_0)}(X) = f(X_0) + \langle \nabla f(X_0), (X - X_0) \rangle.$$

La fonction d'appuis vérifie les propriétés suivantes :

- 1) $f_{ap(X_0)}(X)$ est linéaire.
- 2) Elle coïncide avec f au point X_0 .
- 3) Elle reste inférieure à la fonction $f(X)$.
- 4) $f_{ap(X_0)}(X)$ est une fonction borne inférieure de f .

1.2.5 Théorème de la séparation par des hyperplans affines [2],[3],[8]

Théorème 1.2.6 "de séparation" (d'un point de l'ensemble convexe) :

Soit $\bar{X} \notin S$ convexe fermé alors : il existe un hyperplan $H = \{X \in \mathbb{R}^n / \langle C, X \rangle = \alpha\}$; tel que :

$$\langle C, \bar{X} \rangle = \alpha \text{ et } \langle C, X \rangle < \alpha, \forall X \in S.$$

Démonstration 1.2.6 $\bar{X} \notin S \Rightarrow \exists P_{\bar{X}} \in S$ qui est sa projection.

On pose $C = \bar{X} - P_{\bar{X}}$ et $\alpha = \langle \bar{X} - P_{\bar{X}}, \bar{X} \rangle$. Donc $\langle C, \bar{X} \rangle = \alpha$.

Il reste à vérifier que $\langle C, X \rangle < \alpha, \forall X \in S$.

On a la 2^{ème} définition de la projection : $\langle X - P_{\bar{X}}, \bar{X} - P_{\bar{X}} \rangle \leq 0, \forall X \in S ;$
 $\langle C, X \rangle - \langle C, P_{\bar{X}} \rangle \leq 0 \Rightarrow \langle C, X \rangle \leq \langle C, P_{\bar{X}} \rangle, \forall X \in S. \quad (*)$

D'autre part :

$$0 < \|C\|^2 = C.C = \langle C, \bar{X} - P_{\bar{X}} \rangle = \langle C, \bar{X} \rangle - \langle C, P_{\bar{X}} \rangle \Rightarrow \langle C, P_{\bar{X}} \rangle < \langle C, \bar{X} \rangle. \quad (**)$$

(*) et (**) $\Rightarrow \langle C, X \rangle \leq \langle C, P_{\bar{X}} \rangle < \bar{X}, \forall X \in S$, donc $\langle C, X \rangle < \langle C, P_{\bar{X}} \rangle < \alpha$,
donc $\langle C, X \rangle < \alpha, \forall X \in S$.

Maintenant, si on fait tendre $\bar{X}$ vers $P_{\bar{X}}$ qui se trouve sur la frontière de S ; on obtient un hyperplan $H = \{X \in \mathbb{R}^n / \langle C, X \rangle = \alpha\}$.

On a : $\langle C, \bar{X} \rangle = \alpha (\bar{X} = P_{\bar{X}}), \langle C, X \rangle \leq \alpha, \forall X \in S$.

Une autre démonstration est donnée en [8]

Théorème 1.2.7 "de séparation"(de deux ensembles convexes "au sens large") :

Si S_1 et S_2 sont deux convexes disjoints de $\mathbb{R}^n$, il existe un hyperplan affine qui les sépare, i.e. un vecteur $s \neq 0$ de $\mathbb{R}^n$ tel que :

$$\langle s, x_1 \rangle \leq \langle s, x_2 \rangle \text{ pour tout } x_1 \in S_1 \text{ et tout } x_2 \in S_2.$$

Michel Minoux a exprimé le théorème (1.2.7) par une autre manière en [8].

Démonstration 1.2.7 Ce théorème est démontré en [8]

Théorème 1.2.8 "de séparation"(de deux ensembles convexes "au sens strict") :

Si S_1 est un convexe fermé et S_2 un convexe compact¹ de $\mathbb{R}^n$, et s'ils sont disjoints, il existe un hyperplan affine qui les sépare strictement², i.e. un vecteur $s \neq 0$ de $\mathbb{R}^n$ et $r \in \mathbb{R}$ tel que :

$$\langle s, x_1 \rangle < r \text{ pour tout } x_1 \in S_1$$

et

$$\langle s, x_2 \rangle > r \text{ pour tout } x_2 \in S_2$$

1. On dit qu'une partie S_2 d'un espace métrique est compacte si toute suite de S_2 possède une sous suite (suite extraite) dans S_2 .

2. Le problème de la séparation d'ensemble par des hyperplan remonterait au moins à Paul Kirchberger (1878 – 1945) en 1902. Celui de remplacer les hyperplans par des surfaces sphérique fut d'abord posé par Federick Albert Valentine en 1964. Cette question ainsi que d'autres sujets s'y rapportant, incluant la séparation des moutons stationnaires de chèvres par une clôture droite, sont très bien traité dans le chapitre sur le théorème de type Kirchberger de S.R.Lay[10].

Démonstration 1.2.8 *Ce théorème se démontre de la même façon que le théorème (1.2.7) (voir [8]).*

1.3 Fonctions convexes et non convexes

1.3.1 Définitions de fonctions convexes, non convexes (concaves)[1],[2],[3],[9, 10]

Définition 1.3.1 *Soit S un convexe de $\mathbb{R}^n$; $f : S \rightarrow \mathbb{R}$ est dite **convexe** sur S si : $\forall X_1, X_2 \in S$ et $\forall \lambda \in [0, 1]$*

$$f(\lambda X_1 + (1 - \lambda)X_2) \leq \lambda f(X_1) + (1 - \lambda)f(X_2) \quad (\star) \quad (\text{cf. Figure 1.4})$$

*f est dite **strictement convexe** sur S quand l'inégalité $(\star)$ est stricte dès que $X_1 \neq X_2$.*

*Une propriété encore plus forte est donnée comme suit : f est dite **fortement convexe** sur S , de module de forte convexité $s > 0$, lorsque*

$$f(\lambda X_1 + (1 - \lambda)X_2) \leq \lambda f(X_1) + (1 - \lambda)f(X_2) - \frac{1}{2}s\lambda(1 - \lambda)\|X_2 - X_1\|^2$$

pour tout $(X_1, X_2) \in S \times S$ et tout $\lambda \in]0, 1[$.

Exemple 1.3.1 EXEMPLES DE FONCTIONS CONVEXES :

-Si $\|\cdot\|$ est une norme sur $\mathbb{R}^n$ et S est un convexe de $\mathbb{R}^n$, alors la fonction distance

$$d_C : x \in \mathbb{R}^n \mapsto d_C(x) := \inf\{\|x - c\|, c \in C\}$$

est convexe sur $\mathbb{R}^n$.

-Si $f : x \in \mathbb{R}_*^+ \mapsto f(x)$ est convexe sur $\mathbb{R}_*^+$ il en est de même de la fonction

$$g : x \in \mathbb{R}_*^+ \mapsto g(x) := xf\left(\frac{1}{x}\right)$$

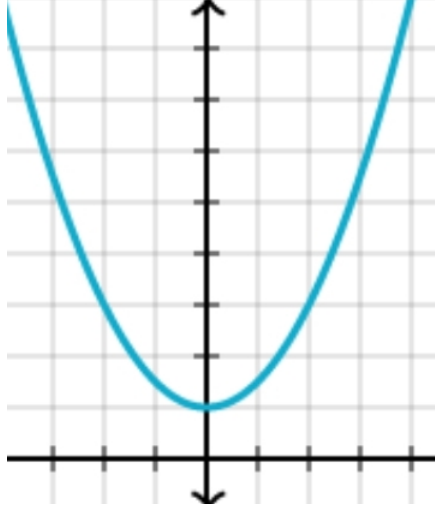


FIGURE 1.4 – Fonction convexe.

Remarque 1.3.1 *Il existe une notion moins forte et moins restrictive que la convexité, c'est la quasi-convexité.*

En quasi-convexité³ on a juste besoin d'avoir que $\forall k \in \mathbb{R}$ l'ensemble

$$\{x \in S : f(x) \leq k\}$$

soit convexe. Or, il suffit que

$$\forall x, y \in S \text{ tel que } f(x) \leq k \text{ et } f(y) \leq k$$

On ait

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

Définition 1.3.2 *Une fonction $f : \mathbb{R}^n \rightarrow \mathbb{R} \cup \{+\infty\}$ est quasi convexe (quasi-concave) dans un convexe $S \subset \mathbb{R}^n$ si $\forall \lambda \in]0, 1[, \forall x, y \in S, f(\lambda x + (1 - \lambda)y) \leq (\geq) \max\{f(x), f(y)\}$.*

Remarque 1.3.2 *Une fonction convexe est quasi convexe ; mais la réciproque n'est pas vraie.*

Par exemple la fonction f définie comme suit :

3. La quasi convexité fut introduite à l'origine pour comprendre les questions liées à la semi continuité et l'existence dans le calcul des variations. Récemment, elle est utilisée en analyse microstructures, le passage de modèles atomiques aux modèles continus, les approches d'hybrides...etc.

$$f(x) = \begin{cases} \|x\| + 1, & \text{si } x \neq 0 \\ 0, & \text{si } x = 0 \end{cases}$$

est une fonction quasi-convexe mais pas convexe.

Définition 1.3.3 Une fonction f est dite **concave**(non convexe) si $-f$ est convexe c'est à dire : $\forall X_1, X_2 \in S, \forall \lambda \in [0, 1]$:

$$f(\lambda X_1 + (1 - \lambda)X_2) \geq \lambda f(X_1) + (1 - \lambda)f(X_2) \quad (\star\star) \text{ (cf.Figure 1.5)}$$

f est dite **strictement concave** si l'inégalité $(\star\star)$ est stricte dès que $X_1 \neq X_2$.

Une propriété encore plus forte est comme suit : f est dite **fortement concave** sur S , de module de forte concavité s' , lorsque

$$f(\lambda X_1 + (1 - \lambda)X_2) \geq \lambda f(X_1) + (1 - \lambda)f(X_2) - \frac{1}{2}s'\lambda(1 - \lambda)\|X_2 - X_1\|^2$$

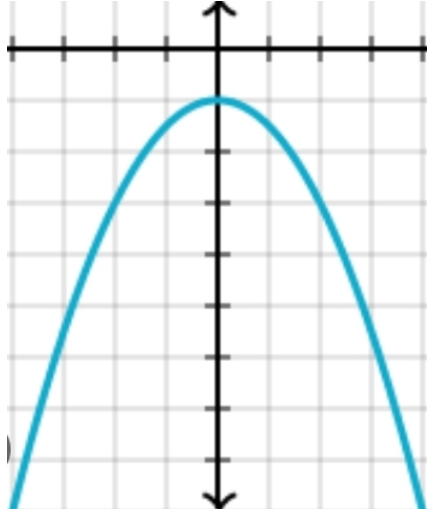


FIGURE 1.5 – Fonction concave.

Remarque 1.3.3 Il existe des fonctions qui sont ni convexes ni concaves par exemple : la fonction $f(X) = X^3$. (cf.Figure 1.6)

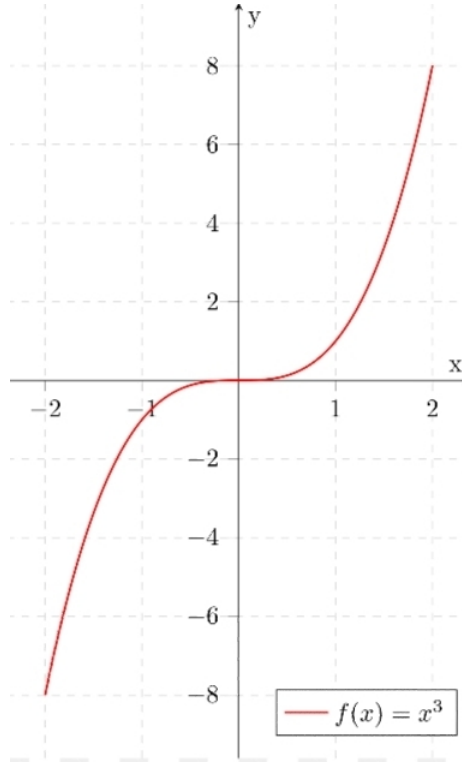


FIGURE 1.6 – Fonction ni convexe ni concave.

Remarque 1.3.4 *Par convention, la fonction identiquement égale à $+\infty$ ou $-\infty$ est convexe et concave au même temps.*

Remarque 1.3.5 *On a les implications suivantes :*

fortement convexe $\Rightarrow$ strictement convexe $\Rightarrow$ convexe $\Rightarrow$ quasi-convexe.

Exemple 1.3.2 *La fonction $f(x) = x$ est convexe, mais pas strictement convexe dans $\mathbb{R}$.*

Exemple 1.3.3 *La fonction $f(x) = e^{-x}$ est strictement convexe, mais pas fortement convexe dans $\mathbb{R}_+$. (Pour la preuve on renvoie le lecteur à consulter l'ouvrage [9]).*

1.3.2 Interprétation géométrique [3]

1.3.2.1 Interprétation géométrique des fonctions convexes

Soit $S = [a, b]$ un intervalle. On a l'équation de la droite qui passe par

$$(X_1, f(X_1)) \text{ et } (X_2, f(X_2)) \quad \frac{y - f(X_1)}{f(X_2) - f(X_1)} = \frac{X - X_1}{X_2 - X_1}$$

$$Y = \frac{X - X_1}{X_2 - X_1}(f(X_2) - f(X_1)) + f(X_1)$$

Soit le point $\lambda X_1 + (1 - \lambda)X_2 \in]X_1, X_2[$

$$\begin{aligned} Y_{\lambda X_1 + (1-\lambda)X_2} &= \frac{\lambda X_1 + (1-\lambda)X_2 - X_1}{X_2 - X_1}(f(X_2) - f(X_1)) + f(X_1) \\ &= \frac{(\lambda - 1)X_1 + (1 - \lambda)X_2}{X_2 - X_1}(f(X_2) - f(X_1)) + f(X_1) \\ &= \frac{(1 - \lambda)(X_2 - X_1)}{X_2 - X_1}(f(X_2) - f(X_1)) + f(X_1) \\ &= (1 - \lambda)(f(X_2) - f(X_1)) + f(X_1) \\ &= (1 - \lambda)f(X_2) - (1 - \lambda)(f(X_1) + f(X_1)) \\ &= (1 - \lambda)f(X_2) + \lambda f(X_1) \\ &\geq f(\lambda X_1 + (1 - \lambda)X_2) \end{aligned}$$

La corde joignant les points $(X_1, f(X_1))$ et $(X_2, f(X_2))$ se trouve au dessus du graphe de f .

1.3.2.2 Interprétation géométrique des fonctions concaves

L'interprétation géométrique des fonctions concaves se fera de la même façon que celle des fonctions convexes ; et on trouve que la corde joignant les deux points $(X_1, f(X_1))$ et $(X_2, f(X_2))$ est toujours au dessous de graphe de f .

Dans tout ce qui suit minimiser une fonction convexe $f \Leftrightarrow$ maximiser une fonction concave $-f$

1.3.3 Propriétés des fonctions convexes et concaves [1],[2],[3]

Proposition 1.3.1 f est convexe sur S convexe si et seulement si ($\Leftrightarrow$) :

$$f\left(\sum_{i=1}^n \lambda_i X_i\right) \leq \sum_{i=1}^n \lambda_i f(X_i), \quad \sum_{i=1}^n \lambda_i = 1, \quad \lambda_i \geq 0, \quad X_i \in S, \quad i = 1, \dots, n, \quad \forall n \in \mathbb{N}^*$$

Démonstration 1.3.1 1) ($\Leftarrow$) si c'est vrai pour $\forall n \in \mathbb{N}^*$, en particulier c'est vérifié pour $n = 2$ c'est exactement la convexité de f .

$$\text{Pour } n = 2 \quad f\left(\sum_{i=1}^2 \lambda_i X_i\right) \leq \sum_{i=1}^2 \lambda_i f(X_i), \quad \sum_{i=1}^2 \lambda_i = 1, \quad \lambda_i \geq 0, \quad \forall X_1, X_2 \in S$$

$$f(\lambda_1 X_1 + \lambda_2 X_2) \leq \lambda_1 f(X_1) + \lambda_2 f(X_2)$$

$$\Rightarrow f(\lambda_1 X_1 + (1 - \lambda_1) X_2) \leq \lambda_1 f(X_1) + (1 - \lambda_1) f(X_2) \quad (\text{avec } \lambda_2 = 1 - \lambda_1)$$

2) ($\Rightarrow$) On utilise la récurrence pour $n = 2$ c'est vérifié car f est convexe.

On suppose vraie à l'ordre n c'est-à-dire :

$$f\left(\sum_{i=1}^n (\lambda_i X_i)\right) \leq \sum_{i=1}^n \lambda_i f(X_i), \quad \sum_{i=1}^n \lambda_i = 1, \quad \lambda_i \geq 0, \quad X_i \in S, \quad i = 1, \dots, n.$$

On montre que c'est vraie à l'ordre $n + 1$ i.e. :

$$f\left(\sum_{i=1}^{n+1} \lambda_i X_i\right) \leq \sum_{i=1}^{n+1} \lambda_i f(X_i), \quad \sum_{i=1}^{n+1} \lambda_i = 1, \quad \lambda_i \geq 0, \quad i = 1, \dots, n, \quad X_i \in S, \quad i = 1, \dots, n + 1.$$

$$\sum_{i=1}^{n+1} \lambda_i X_i = \sum_{i=1}^n \lambda_i X_i + \lambda_{n+1} X_{n+1}$$

Cas(01) : $\lambda_{n+1} = 0$.

$$\sum_{i=1}^{n+1} \lambda_i X_i = \sum_{i=1}^n \lambda_i X_i \quad f\left(\sum_{i=1}^{n+1} \lambda_i X_i\right) = f\left(\sum_{i=1}^n \lambda_i X_i\right) \leq \sum_{i=1}^n \lambda_i f(X_i) \leq \sum_{i=1}^{n+1} \lambda_i f(X_i)$$

(Hypothèse
de récurrence)

Cas(02): $\lambda_{n+1} \neq 0$.

$$\begin{aligned} \sum_{i=1}^{n+1} \lambda_i X_i &= \sum_{i=1}^n \lambda_i X_i + \lambda_{n+1} X_{n+1} \\ &= \sum_{j=1}^n \lambda_j \sum_{i=1}^n \frac{\lambda_i}{\sum_{j=1}^n \lambda_j} X_i + \lambda_{n+1} X_{n+1}. \end{aligned}$$

$$= \sum_{j=1}^n \lambda_j \sum_{i=1}^n \mu_i X_i + \lambda_{n+1} X_{n+1} \quad (\text{avec } \mu = \frac{\lambda_i}{\sum_{j=1}^n \lambda_j}, \mu > 0, \sum_{i=1}^n \mu_i = 1.)$$

$$\text{On a : } \sum_{i=1}^n \mu_i X_i = Y \in S, \text{ donc : } \sum_{i=1}^{n+1} \lambda_i X_i = \sum_{i=1}^n \lambda_j Y + \lambda_{n+1} X_{n+1}$$

$$\begin{aligned} f\left(\sum_{i=1}^{n+1} \lambda_i X_i\right) &= f\left(\sum_{j=1}^n \lambda_j Y + \lambda_{n+1} X_{n+1}\right) \\ &\leq \sum_{j=1}^n \lambda_j f(Y) + \lambda_{n+1} f(X_{n+1}) \\ &= \sum_{j=1}^n \lambda_j f\left(\sum_{i=1}^n (\mu_i X_i) + \lambda_{n+1} f(X_{n+1})\right) \\ &\leq \sum_{j=1}^n \lambda_j \sum_{i=1}^n \mu_i f(X_i) + \lambda_{n+1} f(X_{n+1}) \quad (\text{Par hypothèse de récurrence}). \end{aligned}$$

$$\begin{aligned} &\leq \sum_{j=1}^n \lambda_j \sum_{i=1}^n \frac{\lambda_i}{\sum_{j=1}^n \lambda_j} f(X_i) + \lambda_{n+1} f(X_{n+1}) \\ &= \sum_{i=1}^{n+1} \lambda_i f(X_i). \text{ Donc le résultat est vrai } \forall n. \end{aligned}$$

C'est l'inégalité de JENSEN.⁴

Définition 1.3.4 On appelle *épigraphe* de f est l'ensemble noté par $\text{épi}(f)$, avec $f : S \in \mathbb{R}^n \rightarrow \mathbb{R}$, définit tel que : $\text{épi}(f) = \{(x, r) \in S \times \mathbb{R} / f(x) \leq r\}$.

Proposition 1.3.2 f est convexe si et seulement si $\text{épi}(f)$ est un ensemble convexe.

Démonstration 1.3.2 On doit montrer la double implication ((i) f est convexe $\Leftrightarrow$ (ii) $\text{épi}(f)$ est convexe.)

(i) $(\Rightarrow) f$ est convexe $\Rightarrow \text{épi}(f)$ est un ensemble convexe.

Soient (X_1, r_1) et $(X_2, r_2) \in \text{épi}(f)$, $\lambda \in [0, 1]$, on montre que : $\lambda(X_1, r_1) + (1 - \lambda)(X_2, r_2) \in \text{épi}(f)$.

4. Johan Ludwig William Valdemar Jensen, mieux connu comme Johan Jensen (1859-1925).

$$\begin{aligned}\lambda(X_1, r_1) + (1 - \lambda)(X_2, r_2) &= (\lambda X_1, \lambda r_1) + ((1 - \lambda)X_2, (1 - \lambda)r_2) \\ &= \lambda X_1 + (1 - \lambda)X_2, \lambda r_1 + (1 - \lambda)r_2 \in \text{épi}(f)(?)\end{aligned}$$

(Convexité

$$\begin{aligned}f(\lambda X_1 + (1 - \lambda)X_2) &\leq \lambda f(X_1) + (1 - \lambda)f(X_2) \leq \lambda r_1 + (1 - \lambda)r_2 \\ &\text{de } f)\end{aligned}$$

$\Rightarrow (\lambda X_1 + (1 - \lambda)X_2, \lambda r_1 + (1 - \lambda)r_2) \in \text{épi}(f)$; donc $\text{épi}(f)$ est convexe.

ii)($\Leftarrow$) $\text{épi}(f)$ convexe $\Rightarrow f$ est convexe.

Soient $X_1, X_2 \in S$, $\lambda \in [0, 1]$. On montre que :

$$f(\lambda X_1 + (1 - \lambda)X_2) \leq \lambda f(X_1) + (1 - \lambda)f(X_2).$$

Soient $(X_1, f(X_1))$ et $(X_2, f(X_2)) \in \text{épi}(f)$.

$\text{épi}(f)$ est convexe $\Rightarrow \lambda(X_1, f(X_1)) + (1 - \lambda)(X_2, f(X_2)) \in \text{épi}(f)$

$$(\lambda X_1, \lambda f(X_1)) + ((1 - \lambda)X_2, (1 - \lambda)f(X_2)) \in \text{épi}(f)$$

$$\Rightarrow (\lambda X_1 + (1 - \lambda)X_2, \lambda f(X_1) + (1 - \lambda)f(X_2)) \in \text{épi}(f).$$

$$\Rightarrow f(\lambda X_1 + (1 - \lambda)X_2) \leq \lambda f(X_1) + (1 - \lambda)f(X_2), \text{ donc } f \text{ est convexe.}$$

Définition 1.3.5 On appelle *domaine effectif de définition* d'une fonction convexe f l'ensemble noté par $\text{dom}(f)$, qui est la projection de $\text{l'épi}(f)$ sur $\mathbb{R}^n$, tel que : $\text{dom}(f) = \{x \in \mathbb{R}^n / \exists \mu < +\infty : (x, \mu) \in \text{épi}(f)\}$.

Il est défini par : $\{x \in \mathbb{R}^n / f(x) < +\infty\}$.

Définition 1.3.6 On appelle *l'ensemble de niveau* α pour $f : S \in \mathbb{R}^n \rightarrow \mathbb{R}$ l'ensemble : $L^\alpha f = \{X \in S / f(x) \leq \alpha\}$.

Proposition 1.3.3 f est convexe $\Rightarrow L^\alpha$ est convexe (1), mais l'autre implication ($L^\alpha f$ est convexe $\Rightarrow f$ est convexe (2)) n'est pas vraie.

Démonstration 1.3.3 1) f est convexe $\Leftrightarrow \forall X_1, X_2 \in S, \forall \lambda \in [0, 1]$:

$$f(\lambda X_1 + (1 - \lambda)X_2) \leq \lambda f(X_1) + (1 - \lambda)f(X_2).$$

Montrons que $L^\alpha f$ est convexe $\Leftrightarrow \forall X_1, X_2 \in L^\alpha f, \forall \lambda \in [0, 1]$

$$\lambda X_1 + (1 - \lambda)X_2 \in L^\alpha f, f(\lambda X_1 + (1 - \lambda)X_2) \leq \alpha(?)$$

$$f(\lambda X_1 + (1 - \lambda)X_2) \leq \lambda f(X_1) + (1 - \lambda)f(X_2) \quad (\text{Par la convexité de } f)$$

$$X_1 \in L^\alpha f \Leftrightarrow f(X_1) \leq \alpha$$

$$X_2 \in L^\alpha f \Leftrightarrow f(X_2) \leq \alpha$$

$$f(\lambda X_1 + (1 - \lambda)X_2) \leq \lambda f(X_1) + (1 - \lambda)f(X_2) \leq \lambda \alpha + (1 - \lambda)\alpha \leq \alpha$$

$$\Rightarrow \lambda X_1 + (1 - \lambda)X_2 \in L^\alpha f. \text{ Donc } L^\alpha f \text{ est convexe.}$$

2) $L^\alpha f$ convexe $\Rightarrow f$ convexe (?).

Cette implication n'est pas vraie ; car on peut avoir l'ensemble de niveau (L^α) convexe sans que la fonction f soit convexe.

Exemple 1.3.4 Une fonction dont les ensembles de niveaux sont (des segments) convexes n'est pas convexe en générale par exemple : en $1D : f(x) = \sqrt{|x|}$. Une telle fonction est appelée quasi-convexe.

Proposition 1.3.4 Soient f_1, f_2 deux fonctions convexes alors : $f_1 + f_2$ est convexe.

Démonstration 1.3.4 On pose $F = f_1 + f_2$ et on montre que F est convexe sur S .

Soient X et $Y \in S, \forall \lambda \in [0, 1]$; on doit montrer que :

$$F(\lambda X + (1 - \lambda)Y) \leq \lambda F(X) + (1 - \lambda)F(Y).$$

$$F(\lambda X + (1 - \lambda)Y) = f_1(\lambda X + (1 - \lambda)Y) + f_2(\lambda X + (1 - \lambda)Y)$$

(Convexité

$$\leq \lambda f_1(X) + (1 - \lambda)f_1(Y) + \lambda f_2(X) + (1 - \lambda)f_2(Y)$$

de f_1 et f_2)

$$\leq \lambda(f_1(X) + f_2(X)) + (1 - \lambda)(f_1(Y) + f_2(Y))$$

$$\leq \lambda F(X) + (1 - \lambda)F(Y).$$

Donc $F = f_1 + f_2$ est convexe.

Exemple 1.3.5 $f_1(X) = X, f_2(X) = X^2$.

$F(X) = f_1(X) + f_2(X) = X + X^2$. Donc F est convexe.

Remarque 1.3.6 La différence de deux fonctions convexes n'est pas convexe.

Exemple 1.3.6 $f_1(X) = X, f_2(X) = X^2$.

$f_1(X) - f_2(X) = X - X^2$ n'est pas convexe.

Proposition 1.3.5 Soient f_1 et f_2 convexes sur S . Alors $\max\{f_1(X), f_2(X)\}$ est aussi convexe.

Démonstration 1.3.5 On pose : $F(X) = \max\{f_1(X), f_2(X)\}$.

$\forall X, Y \in S, \forall \lambda \in [0, 1]$

$$\begin{aligned}
 F(\lambda X + (1 - \lambda)Y) &= \max\{f_1(\lambda X + (1 - \lambda)Y), f_2(\lambda X + (1 - \lambda)Y)\} \\
 &\leq \max\{\lambda f_1(X) + (1 - \lambda)f_1(Y), \lambda f_2(X) + (1 - \lambda)f_2(Y)\} \\
 &\leq \max\{\lambda(f_1(X), f_2(X)) + (1 - \lambda)(f_1(Y), f_2(Y))\} \\
 &\leq \max\{\lambda\{f_1(X), f_2(X)\} + (1 - \lambda)\{f_1(Y), f_2(Y)\}\} \\
 &\leq \lambda \max\{f_1(X), f_2(X)\} + (1 - \lambda) \max\{f_1(Y), f_2(Y)\} \\
 &\leq \lambda F(X) + (1 - \lambda)F(Y) \\
 &\Rightarrow F = \max\{f_1, f_2\} \text{ est convexe.}
 \end{aligned}$$

Remarque 1.3.7 Si f_1 et f_2 sont convexes, alors $\min\{f_1, f_2\}$ n'est pas convexe.

1.3.4 Critère d'ordre (01) et d'ordre (02) de convexité [3]

1.3.4.1 Critère d'ordre 01 de convexité

Soit f de classe C^1 sur S .

f est convexe $\Leftrightarrow f(X_1) \geq f(X_2) + \langle \nabla f(X_2), X_1 - X_2 \rangle, \forall X_1, X_2 \in S$.

Remarque 1.3.8 L'inégalité de critère d'ordre (01), serait la même avec $(\leq)$ s'il s'agit d'une fonction concave de classe C^1 sur S .

1.3.4.2 Critère d'ordre 02 de convexité

Soit f de classe C^2 sur S .

f est convexe $\Leftrightarrow Hf(x)$ est semi définie positive.

Remarque 1.3.9 Dans le cas d'une fonction concave, de classe C^2 ⁵, $Hf(x)$ est semi définie négative.

1.3.5 Gradient-Différentiel [2],[3]

1.3.5.1 Gradient

Soit $f : \mathcal{O} \subset \mathbb{R}^n \rightarrow \mathbb{R}$ une fonction convexe différentiable en point $\bar{x}$.

5. Une fonction de classe C^n est une fonction continue sur S et sa dérivée d'ordre n est aussi continue sur S .

On dit que la fonction f est continue sur S lorsque sa courbe représentative se trace "sans lever le crayon".

Définition 1.3.7 On appelle **gradient** de f en $\bar{x}$ le vecteur de $\mathbb{R}^n$ noté par $\nabla f(\bar{x})$ (ça se lit "nabla de f " en $\bar{x}$), de composantes $\partial_1 f(\bar{x}), \dots, \partial_n f(\bar{x})$ avec $\partial_i f(\bar{x})$ est la dérivée partielle de f au point $\bar{x}$; tel que :

$$f(x) = f(\bar{x}) + \langle \nabla f(\bar{x}), x - \bar{x} \rangle \quad (\text{cf. Figure 1.7}).$$

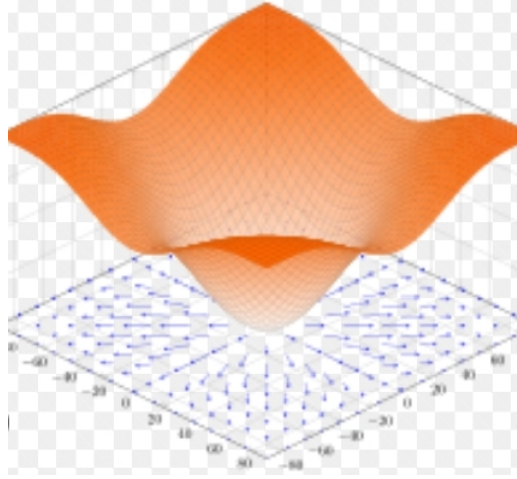


FIGURE 1.7 – Le gradient de la fonction $f(x, y) = (\cos^2 x + \cos^2 y)^2$ représenté dans le plan bas.

1.3.5.2 Différentiel

Soit $f : \mathcal{O} \subset \mathbb{R} \rightarrow \mathbb{R}$ définie sur un ouvert $\mathcal{O}$ de $\mathbb{R}$.

Définition 1.3.8 Quand il s'agit d'une fonction à une seule variable; la différentielle coïncide avec la dérivée.

Soit $f : \mathcal{O} \subset \mathbb{R}^n \rightarrow \mathbb{R}$ définie sur un ouvert⁶ $\mathcal{O}$ de $\mathbb{R}^n$

Définition 1.3.9 f est dite "différentiable" en $\bar{x} \in \mathcal{O}$ si :

$\exists l : \mathbb{R}^n \rightarrow \mathbb{R}$ une application linéaire :

$$f(\bar{x} + h) = f(\bar{x}) + l(h) + \|h\|\epsilon(h), \text{ avec } \epsilon(h) \rightarrow 0 \text{ quand } h \xrightarrow{\neq} 0. \quad (1.1)$$

$$(1.1) \Leftrightarrow \lim_{t \rightarrow 0^+} \frac{f(\bar{x} + td) - f(\bar{x})}{t} = \langle \nabla f(\bar{x}), d \rangle. \quad (1.2)$$

Plus généralement, s'il s'agit de $F : \mathcal{O} \subset \mathbb{R}^n \rightarrow \mathbb{R}^m$, avec $F(x) = \begin{pmatrix} f_1(x) \\ \vdots \\ f_m(x) \end{pmatrix}$

6. Plus fréquemment l'ouvert, l'ensemble ouvert ou encore dit la partie ouverte : est un sous ensemble d'un espace topologique qui ne contient aucun point de sa frontière.

Définition 1.3.10 On dira que , F est *différentiable* en $\bar{x} \in \mathcal{O}$, si chacune de ses fonctions composantes $f_1, f_2, \dots, f_m$ l'est.

Remarque 1.3.10 On associé à F une matrice, qu'on appelle la *matrice jacobienne*, on la note par $JF(x)$, avec $JF(x) = \begin{pmatrix} (\nabla f_1(\bar{x}))^T \\ \vdots \\ (\nabla f_m(\bar{x}))^T \end{pmatrix}$

Définition 1.3.11 Une fonction convexe est dite *deux fois différentiable* en $\bar{x} \in \mathcal{O}$, lorsque son gradient est différentiable en $\bar{x}$.

Remarque 1.3.11 Pour une fonction numérique à plusieurs variables, à valeurs dans $\mathbb{R}$ et deux fois différentiable; on associe une matrice appelée la *matrice hessienne* qu'on note par $\nabla^2 f(\bar{x})$ qui est symétrique; avec : $\nabla^2 f(\bar{x}) = \partial_{ij}^2 f(\bar{x})$; (i : est l'indice de la ligne , j : est l'indice de la colonne ; $\forall i, j = 1, \dots, m$).

1.3.6 Sous-gradients. Sous-différentiel [2],[3]

1.3.6.1 Sous-gradient

Pour une fonction f convexe non partout différentiable, on définit la notion de sous gradient

Définition 1.3.12 On appelle *sous gradient* de f au point X_0 tout vecteur $\xi \in \mathbb{R}^n$, vérifiant : $f(X) \geq f(X_0) + \langle \xi, X - X_0 \rangle$, $\forall X \in \mathbb{R}^n$.

1.3.6.2 Sous-différentiel

Définition 1.3.13 L'ensemble de sous gradients de f en X_0 est appelé le *sous différentiel* de f en X_0 . Il est noté par $\partial f(X_0)$. (cf.Figure 1.8)

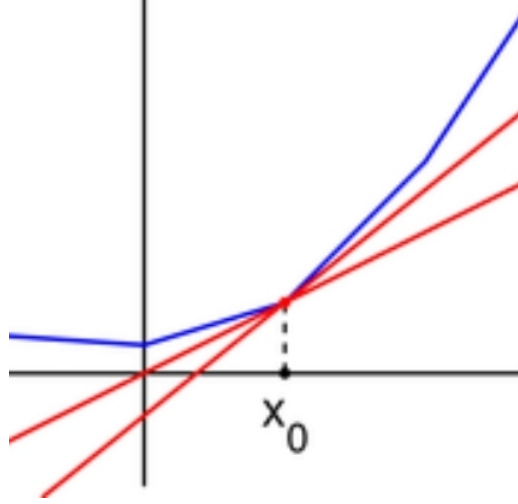


FIGURE 1.8 – Sous-différentiel.

Ici chaque droite (en rouge) représente un sous-gradient, puisque cette fonction n'est pas différentiable. L'ensemble des droites en rouge représente le sous-différentiel.

1.3.7 Sous-gradient et dérivées directionnelles [1],[2],[3]

1.3.7.1 Dérivée directionnelle

Définition 1.3.14 La dérivée directionnelle de f au point x^0 dans la direction y est notée par $\delta f(x^0, y)$, est définie comme suit :

$$\delta f(x^0, y) = \lim_{\lambda \rightarrow 0^+} \frac{f(x^0 + \lambda y) - f(x^0)}{\lambda}$$

1.3.7.2 Remarque : (la relation entre le sous-gradient et la dérivée directionnelle)

Si f est une fonction convexe et pour x^0 et y fixé on a :

$$g(\lambda) = \frac{f(x^0 + \lambda y) - f(x^0)}{\lambda}$$

qui est décroissante, quand λ diminue et restant positif, (avec $f(x^0)$ finie).

D'autre part, si $\partial f(x^0) \neq \emptyset$ et si $\gamma \in \partial f(x^0)$, on a :

$$\frac{f(x^0 + \lambda y) - f(x^0)}{\lambda} \geq \langle \gamma, y \rangle \quad (\forall \lambda) \quad (*)$$

$\Rightarrow g(\lambda)$ a une limite finie. Donc, $\delta f(x^0, y)$ existe en tout x^0 et $\delta f(x^0) \neq \emptyset$ et $\delta f(x^0, y) \geq \langle \gamma, y \rangle \quad \forall \gamma \in \delta f(x^0)$

En générale, $(*)$ est réalisée avec l'égalité pour au moins un sous gradient en x^0 , ce qui s'exprime par le théorème suivant :

Théorème 1.3.1 [1] Si $\text{dom}(f)$ a un intérieur relatif non vide, alors pour tout $x^0 \in \text{rint}[\text{dom}(f)]$, $\delta f(x^0, y)$ existe et :

$$\delta f(x^0, y) = \max_{\gamma \in \partial f(x^0)} \{\langle \gamma, y \rangle\}$$

1.3.8 Caractérisation de la convexité [12]

1.3.8.1 Caractérisation de la convexité

Proposition 1.3.6 Soit $\Omega \in \mathbb{R}$ un ouvert, $S \subset \Omega$ avec S convexe et f une fonction de Ω dans $\mathbb{R}$, de classe C^1 ; alors :

a) les trois propositions suivantes sont équivalentes :

1) f est convexe sur S .

2) $f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle \forall x, y \in S$.

3) ∇f est monotone sur S :

$$\langle \nabla f(y) - \nabla f(x), y - x \rangle \geq 0, \forall x, y \in S.$$

b) Si de plus f est de classe C^2 sur Ω , alors f est convexe sur S si et seulement si : $\langle \nabla^2 f(x)(y - x), y - x \rangle \geq 0, \forall x, y \in S$.

Démonstration 1.3.6 1) $\Rightarrow$ 2). Soit $\lambda \in [0, 1]$, $(x, y) \in S^2$. Alors par la convexité de f , $f(\lambda x + (1 - \lambda)y) \leq (1 - \lambda)f(x) + \lambda f(y)$, d'où $f(x + \lambda(y - x)) \leq \lambda[f(y) - f(x)]$, puis on divise par λ et on fait tendre λ vers 0.

2) $\Rightarrow$ 3 On écrit 2) avec (x, y) puis avec (y, x) , et on somme.

C'est à dire on procède de la façon suivante :

On écrit 2) avec (x, y) ça veut dire : $f(x) \geq f(y) + \langle \nabla f(y), x - y \rangle \forall x, y \in S$ (**A**).

Écrire 2) avec (y, x) signifie : $f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle \forall y, x \in S$ (**B**).

(**A+B=C**) : $f(x) + f(y) \geq f(y) + f(x) + \langle \nabla f(y), x - y \rangle + \langle \nabla f(x), y - x \rangle$.

Donc $\forall x, \forall y \langle \nabla f(x), y - x \rangle + \langle \nabla f(y), x - y \rangle \leq 0$.

Donc $\forall x, \forall y - \langle \nabla f(x), y - x \rangle - \langle \nabla f(y), x - y \rangle \geq 0$.

Ainsi $\forall x, \forall y \langle \nabla f(y) - \nabla f(x), y - x \rangle \geq 0$.

Enfin, ∇f est monotone i.e. : $\forall x, \forall y \langle \nabla f(y) - \nabla f(x), y - x \rangle \geq 0$.

Montrons enfin que 3) $\Rightarrow$ 1) Soient x, y deux éléments de S et $\lambda \in [0, 1]$ et soit $g : I \rightarrow \mathbb{R}$; et $\lambda \rightarrow g(\lambda) = f(\lambda y + (1 - \lambda)x)$, avec $I \supset [0, 1]$. Comme f est C^1 donc g est C^1 et $g'(\lambda) = \langle \nabla f(\lambda y + (1 - \lambda)x), y - x \rangle$.

Soient λ_1 et λ_2 deux éléments de $[0, 1]$ avec $\lambda_1 < \lambda_2$:

$$\begin{aligned} g'(\lambda_2) - g'(\lambda_1) &= \langle \nabla f(\lambda_2 y + (1 - \lambda_2)x), y - x \rangle - \langle \nabla f(\lambda_1 y + (1 - \lambda_1)x), y - x \rangle \\ &= \langle \nabla f(\lambda_2 y + (1 - \lambda_2)x) - \nabla f(\lambda_1 y + (1 - \lambda_1)x), y - x \rangle \\ &= \frac{1}{\lambda_2 - \lambda_1} \langle \nabla f(\lambda_2 y + ((1 - \lambda_2)x) - \nabla f(\lambda_1 y + (1 - \lambda_1)x), (\lambda_2 - \lambda_1)(y - x) \rangle \end{aligned}$$

D'après (c)

donc :

$$\frac{1}{\lambda_2 - \lambda_1} \langle \nabla f(\lambda_2 y + (1 - \lambda_2)x) - \nabla f(\lambda_1 y + (1 - \lambda_1)x), (\lambda_2 - \lambda_1)(y - x) \rangle \geq 0$$

donc : $g'(\lambda_2) - g'(\lambda_1) \geq 0$.

$$\lim_{|x| \rightarrow +\infty} g(x) = +\infty$$

$$\Rightarrow g'(\lambda_2) \geq g'(\lambda_1)$$

g' est donc monotone donc g est convexe sur $[0, 1]$ donc $g(\lambda \times x + (1 - \lambda) \times y) \leq \lambda g(x) + (1 - \lambda)g(y)$ et $g(\lambda) = f(\lambda y + (1 - \lambda)x)$

Posons $x = 1, y = 0$

$$g(1) = (1 \times y) + 0 \times x = f(y)$$

$$g(0) = (0 \times y + (1 - 0)x = f(x)$$

$$\text{donc : } g(\lambda \times 1 + (1 - \lambda) \times 0) \leq (\geq) \lambda g(1) + (1 - \lambda)g(0)$$

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

donc f est convexe .

(b) Si de plus f est C^2 f est convexe $\Leftrightarrow \langle \nabla^2 f(x)(y - x), y - x \rangle \geq 0$
 $\forall x, \forall y \in S$

$\Rightarrow$ (Condition nécessaire) Supposons que f est convexe , $\forall x, \forall y, \forall \lambda \in [0, 1]$
 $f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda)f(y)$

Soient x, y dans S , $\lambda \in [0, 1]$, $g : \Omega \rightarrow \mathbb{R}$, $x \mapsto g(x) = \langle \nabla f(x), h \rangle$.

On a : $\nabla f(x), h = \nabla \langle \nabla f(x), h \rangle$, donc : $\langle \nabla^2 f(x)h, h \rangle = \langle \nabla g(x), h \rangle = \frac{\partial g}{\partial h}(x)$,

donc : $\langle \nabla^2 f(x), h.y \rangle = \lim_{\lambda \rightarrow 0} \frac{g(x + \lambda h) - g(x)}{\lambda}$ dérivé directionnelle de g par

rapport à x dans la direction de $h = \lim_{\lambda \rightarrow 0} \frac{\langle \nabla f(x + h), h \rangle - \langle \nabla f(x), h \rangle}{\lambda}$.

Soit alors x et y dans S tel que : $h = y - x$ donc : $\lambda x + (1 - \lambda)y \in S$

comme S est convexe f est convexe donc ∇f est monotone c'est à dire :

$$\langle \nabla f(y) - \nabla f(x), y - x \rangle \geq 0$$

$$\lim_{\lambda \rightarrow 0} \frac{\langle \nabla f(\lambda y + (1 - \lambda)x - \nabla f(x), y - x \rangle}{\lambda} \geq 0,$$

donc $\langle \nabla^2 f(x)(y - x), y - x \rangle \geq 0, \forall x, y \in S$.

$\Leftarrow$ (Condition suffisante). Supposons que $\forall x, \forall y \langle \nabla^2 f(x), y - x \rangle \geq 0$.

$f(x) - f(y) = \nabla g(y + \theta(y - x))$ et montrons f convexe.

Soit alors $x \in S, y \in S$ et $\lambda \in [0, 1]$

Montrons que f existe. On pose $g : \Omega \rightarrow \mathbb{R}, z \rightarrow g(z) = \langle \nabla f(z), x - y \rangle$ alors

$$\begin{aligned} \langle \nabla f(x) - \nabla f(y), x - y \rangle &= g(x) - g(y) \\ &= \langle \nabla g(y + \theta(y - x)), x - y \rangle \end{aligned}$$

avec $\theta \in [0, 1]$, c'est un développement de Taylor où la suite est incluse dans $\mathbb{R}$, qui est une suite de premier ordre où $g(z) = \nabla \langle \nabla f(z), x - y \rangle = \langle \nabla^2 f(z), x - y \rangle = \langle \nabla^2 f(x).h, h \rangle \quad \forall h \geq 0 \quad \forall x, y \in S$, donc : $\langle \nabla f(x) - \nabla f(y), x - y \rangle = \langle \nabla^2 f(y + \theta(x - y)), x - y \rangle \geq 0$ d'où f est convexe.

1.3.8.2 Caractérisation de la stricte convexité

Proposition 1.3.7 Soit une partie de $\mathbb{R}$ ouverte $S \subset \Omega$ convexe et f une fonction de Ω dans $\mathbb{R}$ de classe C^1 ; alors :

a) Les trois propositions suivantes sont équivalentes :

1) f est strictement convexe sur S .

2) $f(y) > f(x) + \langle \nabla f(x), y - x \rangle$

3) ∇f est strictement monotone sur S $\langle \nabla f(y) - \nabla f(x), y - x \rangle > 0, \forall x, y \in S$.

3) Si de plus f est de classe C^2 sur Ω , alors : $\langle \nabla^2 f(x)(y - x), y - x \rangle > 0. \forall x, y \in S$.

Démonstration 1.3.7 La démonstration se fera de la même manière que celle de la proposition (1.3.6) ; juste avec une légère modification (toutes les inégalités larges deviennent strictes). (Voir [12])

1.3.8.3 Caractérisation de la forte convexité

Proposition 1.3.8 Soit $\Omega \in \mathbb{R}$, un ouvert $S \subset \Omega$ convexe et f une fonction de Ω dans $\mathbb{R}$ de classe C^1 ; alors :

a) Les trois propositions sont équivalentes :

1) f est fortement convexe sur S .

2) $\exists \beta > 0$, tel que : $f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle + \beta \|y - x\|^2$, $\forall x, y \in S$.

3) $\exists \gamma > 0$, tel que : $\langle \nabla f(y) - \nabla f(x), y - x \rangle \geq \gamma \|y - x\|^2$, $\forall x, y \in S$.

b) Si de plus f est de classe $C^2(\Omega)$, alors f est fortement convexe si et seulement si $\exists \beta > 0$, tel que : $\langle \nabla^2 f(x)(y - x), y - x \rangle \geq \beta \|y - x\|^2$ $\forall x, y \in S$.

Démonstration 1.3.8 On peut la déduire à partir de la démonstration donnée en [12]

1.3.9 Fonctions convexes "étendues" [1]

Définition 1.3.15 On dira que f est fonction convexe ("étendue") avec $f : \mathbb{R}^n \rightarrow \bar{\mathbb{R}}$ ($\bar{\mathbb{R}} = \mathbb{R} \cup \{+\infty\} \cup \{-\infty\}$), si et seulement si :

$\forall x$ tel que $f(x) \neq +\infty$, $\forall y$ tel que $f(y) \neq +\infty$, $\forall \lambda \in [0, 1]$:

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

1.4 Programmes convexes et non convexes

1.4.1 Programme convexe [1],[2]

Définition 1.4.1 Un problème de la programmation mathématique est dit *programme convexe* s'il consiste à minimiser une fonction convexe (resp : maximiser une fonction concave) sur un domaine convexe fermé.

Théorème 1.4.1 Pour un programme convexe, tout optimum local est un optimum global.

Démonstration 1.4.1 On peut toujours considéré (sans restriction) un programme convexe de la forme :

$$\begin{cases} \text{Minimiser } f(x) \\ \text{sous la contrainte :} \\ x \in S \end{cases}$$

où f est une fonction convexe, et S est un ensemble convexe.

Soit x^0 est un optimum (minimum ou maximum) local. Pour montrer x^0 est un minimum global, prenant $y \in S (y \neq x^0)$ quelconque et montrons qu'on a nécessairement : $f(x^0) \leq f(y)$.

Raisonnons par absurde, en supposant que : $f(x^0) > f(y)$. En vertu de la convexité de f , ceci implique que :

$$\forall \theta \in]0, 1], f(x^0 + \theta(y - x^0)) \leq (1 - \theta)f(x^0) + \theta f(y) < f(x^0)$$

d'où la contradiction avec le fait que x^0 est un minimum local.

Donc, pour tout $y \in S$, on a bien : $f(x^0) \leq f(y)$, ce qui prouve que x^0 est un minimum global.

1.4.2 Programme non convexe [1]

Définition 1.4.2 Un programme de la programmation mathématique est dit *programme non convexe* si sa fonction objectif ou l'une de ses contraintes n'est pas convexe.

1.5 Conditions d'optimalités

1.5.1 Condition d'optimalité du 1^{er} ordre [2],[9],[13]

1.5.1.1 Cas d'optimisation sans contraintes

Soit $\mathcal{O}$ un ouvert de $\mathbb{R}^n$ et $f : \mathcal{O} \rightarrow \mathbb{R}$.

- Si $\bar{x} \in \mathcal{O}$ est un minimum local de f et si f est différentiable en $\bar{x}$, alors $\nabla f(\bar{x}) = 0$.

- On suppose $\mathcal{O}$ convexe et f convexe sur $\mathcal{O}$. Alors les conditions suivantes relatives à $\bar{x} \in \mathcal{O}$ sont équivalentes :

(i) $\bar{x}$ est un minimum global(absolu) de f sur $\mathcal{O}$.

(ii) $\bar{x}$ est un minimum local de f .

Et si f est différentiable en $\bar{x}$, on a la troisième condition équivalente :

(iii) $\nabla f(\bar{x}) = 0$.

1.5.1.2 Cas d'optimisation avec contraintes

Soit $f : \mathcal{O} \subset \mathbb{R}^n \rightarrow \mathbb{R}$ différentiable sur un ouvert $\mathcal{O}$ de $\mathbb{R}^n$, soit S un ensemble-contraintes décrit par les m égalités et p inégalités

$$S := \{x \in \mathbb{R}^n \mid h_1(x) = 0, \dots, h_m(x) = 0, g_1(x) \leq 0, \dots, g_p(x) \leq 0\}$$

où les $m + p$ fonctions $h_i, g_j : \mathbb{R}^n \rightarrow \mathbb{R}$ sont supposées continûment différentiable sur $\mathbb{R}^n$.

Théorème 1.5.1 (F. John).⁷ Si $\bar{x} \in \mathcal{O} \cap S$ est un minimum local de f sur S , alors il existe $\bar{\lambda}_1, \dots, \bar{\lambda}_m, \bar{\mu}_0, \bar{\mu}_1, \dots, \bar{\mu}_p$ non tous nuls tel que :

$$(i) \bar{\mu}_0 \nabla f(\bar{x}) + \sum_{i=1}^m \bar{\lambda}_i \nabla h_i(\bar{x}) + \sum_{j=1}^p \bar{\mu}_j \nabla g_j(\bar{x}) = 0 ;$$

$$(ii) \bar{\mu}_j \geq 0 \text{ pour tout } j = 0, 1, \dots, p ;$$

$$(iii) \bar{\mu}_j = 0 \text{ si } g_j(\bar{x}) < 0.$$

On écarte l'éventualité $\mu_0 = 0$ (peu informative sur $\bar{x}$) en faisant une hypothèse de **qualification des contraintes** en $\bar{x}$, c'est à dire une hypothèse sur les fonctions-contraintes h_i et g_j représentant S . On considère par exemple la condition $(QC)_{\bar{x}}$ de Mangasarian-Fromowitz⁸ suivante :

$$QC_{\bar{x}} \left\{ \begin{array}{l} \text{Il existe } d_0 \text{ tel que } \langle \nabla h_i(\bar{x}), d_0 \rangle = 0 \text{ pour tout } i = 1, \dots, m \\ \text{et } \langle \nabla g_j(\bar{x}), d_0 \rangle < 0 \text{ pour tout } j \text{ avec } g_j(\bar{x}) = 0; \\ \text{les vecteurs } \nabla h_1(\bar{x}), \dots, \nabla h_m(\bar{x}) \text{ sont linéairement indépendant.} \end{array} \right.$$

Démonstration 1.5.1 Première étape :

On considère la pénalisation suivante :

$$f_N(x) := f(x) + \|x - \bar{x}\|^2 + \frac{N}{2} \left[\sum_{i \in I} h_i(x)^2 + \sum_{j \in J} \max(0, g_j(x))^2 \right].$$

La fonction f_N vérifie $f_N(\bar{x}) = f(\bar{x})$. De plus, la fonction f_N est de classe C^1 en utilisant que $x \mapsto \max(0, g_j(x))^2$ est de classe C^1 (pour ce dernier point utiliser que g_j est de classe C^1 et que $\nabla(g_j(x)^2) = 2g_j(x)\nabla g_j(x)$, ce qui implique le résultat aux éventuels points x où $g_j(x) = 0$).

7. Fritz John : mathématicien américain d'origine allemand, connu par l'équation de John. Il est spécialisé dans les équations aux dérivées partielles et les problème bien posés.

8. En anglais Mangasarian-Fromowitz constraint qualification ; ou la qualification de contraintes de Mangasarian-formowitz ou encore MFCQ en abrégé : est une condition préalable importante pour que les critère d'optimalité nécessaires s'appliquent à l'optimisation non linéaire.

Deuxième étape :

Pour démontrer la deuxième étape de ce théorème, on a besoin de ce lemme qui est à son tour prouvé dans [13].

Lemme 1.5.1 *Il existe $\varepsilon_0 > 0$ tel que pour tout $\varepsilon \in]0, \varepsilon_0[$ il existe $N_\varepsilon > 0$ tel que tout $x \in \mathbb{R}^n$ vérifiant $\|x - \bar{x}\| = \varepsilon$ on ait $f_{N_\varepsilon}(x) > f(\bar{x})$.*

Fixons $\varepsilon \in]0, \varepsilon_0[$. On sait alors que f_{N_ε} admet un minimum local sur l'ouvert $\{x \in \mathbb{R}^n; \|x - \bar{x}\| < \varepsilon\}$ en un point noté x_ε . En ce point la condition nécessaire d'optimalité s'applique et donc

$$\nabla f_{N_\varepsilon}(x_\varepsilon) = 0$$

D'où :

$$\nabla f(x_\varepsilon) + 2(x_\varepsilon - \bar{x}) + N \left[\sum_{i \in I} h_i(x_\varepsilon) \nabla h_i(x_\varepsilon) + \sum_{j \in J} \max(0, g_j(x_\varepsilon)) \nabla g_j(x_\varepsilon) \right] = 0.$$

On pose

$$p_\varepsilon := \left(1 + N^2 \sum_{i \in I} h_i(x_\varepsilon) + N^2 \sum_{j \in J} \max(0, g_j(x_\varepsilon))^2 \right)^{\frac{1}{2}}, \quad \mu_0^\varepsilon := \frac{1}{p_\varepsilon}, \quad \lambda_i^\varepsilon := N \mu_0^\varepsilon h_i(x_\varepsilon), \quad \mu_j^\varepsilon := N \mu_0^\varepsilon \max(0, g_j(x_\varepsilon)).$$

Lorsque ε tend vers 0, x_ε tend vers $\bar{x}$ et $(\mu_0^\varepsilon, \mu^\varepsilon, \lambda^\varepsilon)$ qui est de norme 1 dans $\mathbb{R}^{1+m+p}$ (donc non-nul) tend à une sous-suite prés vers un vecteur de norme 1 noté (μ_0, μ, λ) .

En divisant la condition nécessaire obtenue par p_ε on obtient en faisant $\varepsilon \rightarrow 0$

$$\mu_0 \nabla f(\bar{x}) + \left[\sum_{i \in I} \lambda_i \nabla h_i(\bar{x}) + \sum_{j \in J} \mu_j \nabla g_j(\bar{x}) \right] = 0$$

De plus, la condition de complémentarité est satisfaite. En effet, si $g_j(\bar{x}) < 0$ pour un certain j , alors on a $g_j(\bar{x}) < 0$ pour $\varepsilon > 0$ suffisamment petit. Donc $\mu_j^\varepsilon = 0$ par définition de μ_j^λ . D'où à la limite $\mu_j = 0$.

Théorème 1.5.2 (Karush-Kuhn-Tucker)⁹ (Cas de contraintes d'égalité) ou encore appelé (Le théorème des multiplicateurs de

9. Les conditions de Karush-Kuhn-Tucker ou anciennement connues sous le nom des conditions de Kuhn-Tucker qui sont une généralisation des multiplicateurs de Lagrange, (parfois appelées les conditions nécessaires de 1^{er} ordre). Elles sont utilisées pour la résolutions des problèmes de la programmation non linéaire (avec contraintes d'égalités ou d'inégalités) à fin d'avoir une solution optimale.

Lagrange). Si $\bar{x} \in \mathcal{O} \cap S$ est un minimum local de f sur S , et si $(QC)_{\bar{x}}$ a lieu, il existe alors

$\bar{\lambda} = (\bar{\lambda}_1, \dots, \bar{\lambda}_m) \in \mathbb{R}^m$ et $\bar{\mu} = (\bar{\mu}_1, \dots, \bar{\mu}_p) \in \mathbb{R}^p$ tel que :

$$(i) \quad \nabla f(\bar{x}) + \sum_{i=1}^m \bar{\lambda}_i \nabla h_i(\bar{x}) + \sum_{j=1}^p \bar{\mu}_j \nabla g_j(\bar{x}) = 0 ;$$

(ii) $\bar{\mu}_j \geq 0$ pour tout $j = 1, \dots, p$;

(iii) $\bar{\mu}_j = 0$ si $g_j(\bar{x}) < 0$.

Les $\bar{\lambda}_i$ et $\bar{\mu}_j$ mis en évidence dans ce théorème de Karush-Khun-Tucker (KKT en abrégé) sont appelés **multiplicateurs de Lagrange** (ou de **Lagrange KKT**).¹⁰

Une hypothèse de qualification de contraintes en $\bar{x}$, plus forte que $(QC)_{\bar{x}}$, est la suivante :

$(QC)'_{\bar{x}}$ Les vecteur $\nabla h_1(\bar{x}), \dots, \nabla h_m(\bar{x}), \nabla g_j(\bar{x}), j \in J(\bar{x})$, sont linéairement indépendants (L.I), où $J(\bar{x}) := \{j | g_j(\bar{x}) = 0\}$ est l'ensemble des indices des contraintes (de type inégalité) actives (ou saturées) en $\bar{x}$.

Si les fonctions h_i et g_j sont affines, on peut se dispenser de toute hypothèse de qualification des contraintes et déduire quand même les conditions nécessaires de minimalité de KKT.

Les conditions (i) – (ii) – (iii) de théorème de KKT (appelées les conditions KKT) deviennent suffisantes en présence de convexité dans les données. Considérons à cet effet le contexte suivant :

$$(C) \quad \begin{cases} f \text{ est convexe sur l'ouvert convexe } \mathcal{O}; \\ \text{les fonctions } h_i : \mathbb{R}^n \rightarrow \mathbb{R} \text{ sont affines ;} \\ \text{les fonctions } g_j : \mathbb{R}^n \rightarrow \mathbb{R} \text{ sont convexes.} \end{cases}$$

On a d'autre versions pour le théorème ci-dessous : Celle pour le cas des contraintes d'inégalités (Théorème de Karush-Kuhn-Tucker), celle des contraintes mixtes (Contraintes d'égalités et d'inégalités mixtes). Pour plus de détails et d'explication sur le théorème référez à l'ouvrage [9].

Démonstration 1.5.2 La démonstration se fera en utilisant le théorème de Fritz-John (Voir [9, 13]).

10. Les multiplicateurs de Lagrange est une méthode permettant de trouvé des points stationnaires (maximum, minimum...) d'une fonction dérivable d'une ou plusieurs variables.

Théorème 1.5.3 Dans la situation décrite par (C) ci-dessus, les conditions de KKT sont suffisantes pour que $\bar{x}$ soit un minimum de f sur $\mathcal{O} \cap S$.

Dans le contexte décrit par (C), il y a une condition d'énoncé simple assurant $(QC)_{\bar{x}}$ en tout point de S ; c'est la condition dite de Slater que voici : en notant $h_i : x \mapsto h_i(x) = \langle a_i, x \rangle - b_i$ les fonctions-contraintes du type égalité, et $Ax = b$ la conjonction des m contraintes du type égalité $h_i(x) = 0, i = 1, \dots, m$,

$$(S) \begin{cases} \text{Les vecteurs } a_i \text{ sont linéairement indépendants} \\ \text{(i.e. } A \text{ est surjective);} \\ \text{Il existe } x_0 \text{ tel que } Ax_0 = b \text{ et } g_j(x_0) < 0 \text{ pour tout } j = 1, \dots, p. \end{cases}$$

Les deux conditions réunies données dans le théorème 1.5.3 portent le nom de **Mangasarian-Fromowitz**.

Démonstration 1.5.3 La preuve est détaillée en [13].

1.5.2 Condition d'optimalité du 2nd ordre [2],[13]

1.5.2.1 Cas d'optimisation sans contraintes

Soit $\mathcal{O}$ un ouvert de $\mathbb{R}^n$ et $f : \mathcal{O} \rightarrow \mathbb{R}$.

- Si $\bar{x} \in \mathcal{O}$ est minimum local de f et si f est deux fois différentiable en $\bar{x}$, alors :

$\nabla f(\bar{x}) = 0$ et $\nabla^2 f(\bar{x})$ est semi-définie positive.

- Si $\bar{x} \in \mathcal{O}$ est un point où f est deux fois différentiable et si :

$\nabla f(\bar{x}) = 0, \nabla^2 f(\bar{x})$ est définie positive,

alors $\bar{x}$ est un minimum strict de f .

1.5.2.2 Cas d'optimisation avec contraintes

Soit $f : \mathbb{R}^n \rightarrow \mathbb{R}, h_i : \mathbb{R}^n \rightarrow \mathbb{R}, 1 \leq i \leq m$ et $g_j : \mathbb{R}^n \rightarrow \mathbb{R}, 1 \leq j \leq p$ des fonctions de classe C^2 . Soit K la contrainte fermée définie par :

$$K := \{x \in \mathbb{R}^n; h_i(x) = 0; i \in I; g_j(x) \leq 0; j \in J\},$$

avec $I = \{1, \dots, m\}$ et $J = \{1, \dots, p\}$ et soit le problème de minimisation :

$$\min_{x \in K} f(x). \quad (P)$$

Soit $\mathcal{L} : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}_+^p \rightarrow \mathbb{R}$ le lagrangien associé à (P) défini par

$$\mathcal{L}(x, \lambda, \mu) := f(x) + \sum_{i \in I} \lambda_i h_i(x) + \sum_{j \in J} \mu_j g_j(x).$$

Pour $x \in K$, on pose $J(x) := \{j \in J; g_j(x) = 0\}$ et :

$$\Lambda(x) := \{(\lambda, \mu) \in \mathbb{R}^m \times \mathbb{R}_+^p; \nabla_x \mathcal{L}(x, \lambda, \mu) = 0 \text{ et } \lambda_i = 0 \ \forall i \notin I(x)\}.$$

On définit le cône critique par

$$C(x) := \{v \in T_K(x); \langle \nabla f(x), v \rangle = 0\}.$$

Théorème 1.5.4 *Soit x un minimum local de du problème et (λ, μ) les multiplicateurs associés. Si les vecteurs :*

$$\nabla h_i(x), i \in I(x), \nabla g_j(x), j \in J(x)$$

sont linéairement indépendants, alors $\nabla_x^2 \mathcal{L}(x, \lambda, \mu)$ est positif dans le sous ensemble :

$$M := \text{Vect}(\nabla h_i(x), i \in I(x), \nabla g_j(x), j \in J(x))^\perp,$$

i.e. si d vérifie $\langle d, \nabla h_i(x) \rangle, \langle d, \nabla g_j(x) \rangle = 0$, alors

$$\langle \nabla_x^2 \mathcal{L}(x, \lambda, \mu) d, d \rangle \geq 0.$$

Pour faire la preuve, commençons par le lemme suivant (parfois appelé le lemme de Gordan).

Lemme 1.5.2 *Soit $(a_i)_{1 \leq i \leq m}$ et $(b_j)_{1 \leq j \leq p}$, $m + p$ vecteurs de $\mathbb{R}^n$. Considérons les deux assertions suivantes :*

- (i) $\exists y \in \mathbb{R}^n$, tel que $\forall 1 \leq i \leq m, \langle a_i, y \rangle < 0$ et $\forall 1 \leq j \leq p, \langle b_j, y \rangle = 0$.
- (ii) La famille $\{a_1, \dots, a_m, b_1, \dots, b_p\}$ est linéairement indépendante.

Démonstration 1.5.4 *Ce lemme est démontré en détails en [13]*

Démonstration 1.5.5 *On applique le lemme précédent avec la famille*

$\nabla h_i(x)$, $(1 \leq i \leq m)$, $\nabla g_j(x)$, $j \in J(x)$. On déduit qu'il existe un vecteur $y \in \mathbb{R}^n$ vérifiant la condition (i) ci-dessus donc que la contrainte est qualifiée au point x . Ainsi, on a :

$$T_k(x) = M$$

Donc il existe une suite de point réalisable (x_k) telle que $x_k \rightarrow x$ telle que $\frac{x_k - x}{\|x_k - x\|} \rightarrow d$ et telle que $h_i(x_k) = 0$ et $g_j(x_k) = 0$ pour $j \in J(x)$. D'où en posant

$$d_k := x_k - x :$$

$$0 \leq f(x_k) - f(x) = \mathcal{L}(x_k, \lambda, \mu) - \mathcal{L}(x, \lambda, \mu) = \frac{1}{2} \langle \nabla_x^2 \mathcal{L}(x, \lambda, \mu) d_k, d_k \rangle + o(\|d_k\|^2),$$

par un développement de Taylor. d'où le résultat en passant à la limite.

On énonce maintenant le théorème des conditions suffisantes au second ordre.

Théorème 1.5.5 Soit $\bar{x} \in K$ et $(\lambda, \mu) \in \Lambda(\bar{x})$. On suppose que pour tout $d \in \mathbb{R}^n$ non nul vérifiant

$$\langle d, \nabla h_i(\bar{x}) \rangle = 0 \quad \forall i \in I, \quad \langle d, \nabla g_j(\bar{x}) \rangle \leq 0 \quad \forall j \in J(\bar{x}); \quad \langle d, \nabla g_j(\bar{x}) \rangle = 0 \quad j \in J(\bar{x}),$$

On a

$$\langle \nabla_x^2 \mathcal{L}(\bar{x}, \lambda, \mu) d, d \rangle > 0.$$

Alors $\bar{x}$ est un minimum local stricte de (P) et il existe $c > 0$ et $\varepsilon > 0$ tel que :

$$\forall x \in \mathbb{R}^n, \quad \|x - \bar{x}\| \leq \varepsilon \Rightarrow f(x) \geq f(\bar{x}) + c\|x - \bar{x}\|^2.$$

Démonstration 1.5.6 On utilise le raisonnement par absurde pour plus de détails voir [13].

1.6 Conclusion

Dans ce chapitre, nous avons défini, étudié la convexité, la non convexité des ensembles, fonctions et programmes. En présence de la convexité, l'optimum local est aussi global, les conditions nécessaires d'optimalité sont

équivalentes aux conditions suffisantes d'optimalité. D'où l'intérêt de la convexité, et la facilité d'étude de ses problèmes.

En absence de la convexité, on utilise l'optimisation globale pour résoudre les problèmes non convexes, qui sera l'objet de deuxième chapitre.

Optimisation globale

2.1 Introduction

Dans nos jours, avec l'évolution de l'informatique beaucoup d'intérêt a été accordé à l'optimisation globale, qui cherche à localiser l'optimum global d'un problème non convexe, avec ou sans contraintes. L'objectif de l'optimisation globale, est de développer des algorithmes efficaces et rapides pour trouver l'extremum global.

Dans ce chapitre, nous allons définir l'optimisation globale, donner la différence de cette dernière, avec l'optimisation locale et citer les différents types d'approches (déterministes, stochastiques et hybrides). Pour guider le choix de la meilleure solution. Il est nécessaire de regarder : (i) la taille de problème, (ii) les propriétés de la fonction objectif et des contraintes (continuité, différentiabilité, linéarité et convexité ...), (iii) et ainsi que le temps disponible pour résoudre le problème.

2.2 Définition d'optimisation globale [3], [4], [6]

Définition 2.2.1 *L'optimisation globale : est une branche des mathématiques appliquées et de l'analyse numérique, qui s'intéresse à la résolution des problèmes d'optimisation non convexe. Elle consiste à chercher la meilleure solution possible, et trouver l'optimum global (ou les optimums globaux) sur $\mathbb{R}$ ou $\mathbb{R}^n$ (optimisation globale sans contraintes) ; ou sur un ensemble bien défini de contraintes (optimisation globale avec contraintes) ; en utilisant*

différentes approches : déterministes (exactes), stochastiques (approchées) et hybrides (qui n'est qu'une combinaison de ces deux premières méthodes) La théorie de l'optimisation globale est assez développée et possède beaucoup d'applications importantes. Récemment une série de travaux a été publiée [54, 57] ...

2.2.1 Optimum local-global [6]

Soit le problème (P) d'optimisation en dimension finie suivant :

$$(P) \begin{cases} \min f(x) \\ x \in S \subseteq \mathbb{R}^n \end{cases}$$

S est appelé l'ensemble faisable (réalisable) et $f : S \rightarrow \mathbb{R}$ la fonction objectif.

Définition 2.2.2 $\bar{x}$ est appelé un *minimum local*, s'il existe une boule ouverte $B(\bar{x}, r)$ de $\mathbb{R}^n$ telle que :

$$\forall x \in B(\bar{x}, r) \cap S : f(\bar{x}) \leq f(x).$$

L'existence de plusieurs optimums locaux, qui donnent des différentes valeurs à la fonction objectif dans de nombreux problèmes d'optimisation nous conduit à définir la notion d'optimum global.

Définition 2.2.3 On dira que $\bar{x}$ est un *minimum global* de f si :

$$\forall x \in S, f(\bar{x}) \leq f(x).$$

2.3 Méthodes d'optimisation globale

On distingue trois types de méthodes d'optimisation global¹ : méthodes déterministes, méthodes stochastiques et méthodes hybrides .

1. Les méthodes d'optimisation globale sont principalement classées en deux catégories : les approches déterministes (Horst and Tuy 1990 ; Grossmann 1996 ; Pinter 1996 ; Esposito and Floudas 2000) et les approches stochastiques (Guus et al. 1995 ; Ali et al. 1997 ; Törn et al. 1999), puis vient une troisième catégorie qui sont les méthodes hybrides qui ne sont qu'une combinaison des deux méthodes précédentes.[16]

2.3.1 Méthodes déterministes (exactes) [7], [19], [20]

Les méthodes déterministes sont des méthodes appliquées sur les problèmes de l'optimisation avec contraintes et ayant une structure particulière. Elles sont subdivisées en deux sous groupes : les méthodes de la programmation mathématique de gradient et de sous gradient. Ces approches s'intéressent beaucoup plus à la qualité de la solution, car il est important de savoir si l'extremum trouvé est local ou global. De plus, elles garantissent d'arriver au minimum global ; puisque la valeur de la fonction objectif trouvée pour l'optimum local est très proche de l'optimum global.

Inconvénients des méthodes exactes :

Pour les problèmes de grandes tailles, le temps de résolution est parfois très grand, d'où la nécessité d'amélioration de certaines d'elles comme : la méthode de Hit and Run (Törn an al 2009).

Parmi les méthodes déterministes qui existent on peut citer :

2.3.1.1 Méthode de Tunnel (Tunneling) [7], [14], [15] [17], [18]

La méthode de Tunnel encore appelée méthode de Tunneling est une méthode de pénalisation², introduite pour la première fois par Levy et Montalvo³ [14].

Cette approche consiste à appliquer la méthode de descente⁴ à une fonction auxiliaire f (appelée la fonction de pénalité ou la fonction de Filled⁵) à partir d'un point quelconque x_0 , on trouve x^* , ensuite on résout l'équation $f(x_0^*) = 0$. On aura donc deux cas :

- 1) Soit l'équation admet des solutions ; et à partir d'une des solution x_1^* , on résout l'équation $f(x_1^*) = 0$, et on répète la procédure.
- 2) Soit l'équation n'admet pas de solution, alors la solution trouvée x_0^* est

2. La méthode de pénalisation : est une méthode d'optimisation avec contraintes, qui consiste à transformer la contrainte en un terme de pénalisation (on dit qu'on relâche la contrainte) ; de plus, elle permet de se ramener au cas d'optimisation sans contraintes.

On distingue deux types de pénalisation : pénalisation extérieure pour les contraintes d'égalité, pénalisation intérieure pour les contraintes d'inégalités.

3. Paul Levy : mathématicien français, l'un des fondateurs de la théorie modernes des probabilités.

4. La méthode de descente : est une méthode itérative, qui consiste à résoudre les problèmes d'optimisation avec et sans contraintes ; tel que dans chaque itération, on résout un problème de minimisation\maximisation à une variable.

5. La fonction de Filled est une fonction utilisée et développée dans la méthode de fonction de Filled ; qui est une méthode d'optimisation globale développée par Y.M.Liang, L.S.Zhang, M.M.Li et B.S.Han.

optimale globale.

Cette méthode est étendue par Yoa [15] pour solutionner les problèmes d'optimisation avec contraintes; et aussi pour résoudre les problèmes ayant des équations non linéaires. (cf. Figure 2.1)

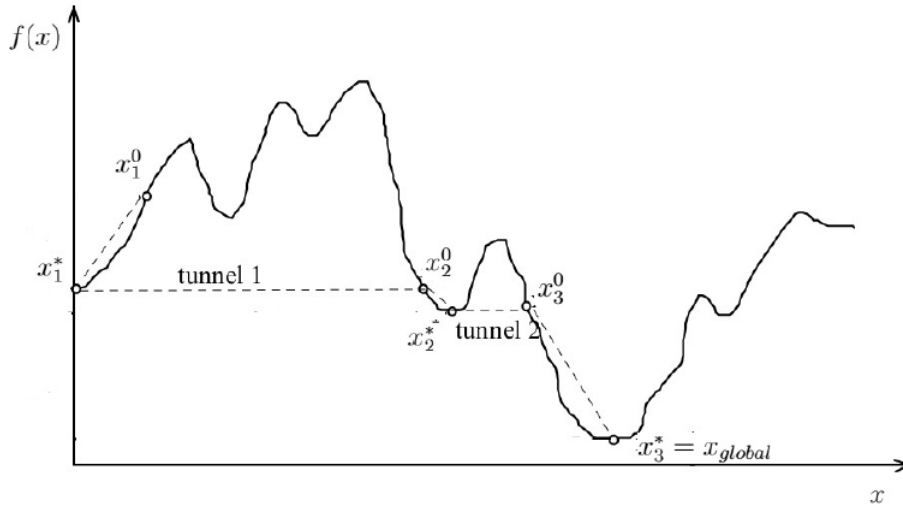


FIGURE 2.1 – Procédure de Tunneling.

2.3.1.2 La méthode de Branch and Bound [4],[5],[6]

La non-existence de critère d'optimalité en optimisation globale, nous ramène à appliquer la méthode de Branch and Bound afin de déterminer le minimum global

Définition 2.3.1 *La méthode de Branch and Bound (en littérature anglo-saxonne) (BB) ou encore la méthode de séparation et d'évaluation : est une approche déterministe utilisée pour la résolution des problèmes non linéaires, non convexes d'optimisation globale, en particulier. Et généralement, pour la résolution des problèmes d'optimisation combinatoire. Cette méthode utilise la stratégie : "le meilleur d'abord".*

Principe de la méthode BB:

► Principe de la méthode :

La procédure de base est composée de deux phases : phase de séparation et phase d'évaluation.

Soit le problème d'optimisation globale suivant :

$$(PG) \begin{cases} \min f(x) \\ x \in S \end{cases}$$

f continue non convexe, S convexe ou pas, S est subdivisé en S_i .

Phase de séparation : consiste à résoudre le problème initial, en subdivisant l'ensemble S en des sous ensembles S_i , et prendre la meilleure solution trouvée sur chaque partition S_i .

Phase d'évaluation : a pour but de déterminer l'optimum, lorsque le problème est facile à résoudre. Néanmoins, dans la pratique on rencontre des problèmes difficiles à résoudre (problèmes avec des sous ensembles non convexes). La procédure la plus utilisée engendre deux suites convergentes qui sont respectivement la borne inférieure LB_K (LB : Lower bound) et la borne supérieure UB_K (UB : Upper Bound) ; de valeur minimale de la fonction objectif. À l'itération K les bornes UB et LB seront :

$$\begin{cases} UB_K = \min UB_{K_i} \\ LB_K = \min LB_{K_i} \end{cases}$$

(Avec S subdivisé en S_{K_i}).

Dans la majorité des cas, on aura besoin d'une fonction auxiliaire F_i , minorante de la fonction objectif ; pour chaque sous ensemble S_i . Et on résout le problème de minimisation suivant :

$$\min_{x \in S_i} F_i(x)$$

.

Si on arrive à trouver que la borne inférieure de deuxième sous problème, est supérieure à la borne supérieure de premier sous problème. Donc, on est sûr que le minimum se trouve dans le premier. Ainsi, on élimine le deuxième, puisque un tel sous ensemble ne peut pas contenir l'optimum. Et on sélectionne un autre sous ensemble parmi les restant, pour poursuivre la phase de séparation.

Notons $\bar{x}$ la solution trouvée pour ce qui suit.

► Algorithme de la méthode :

- 1- Construire l'ensemble R tel que : $S \subset R$.
- 2- Posons : $K = 1$, $I_K = R$, fixer ε (la précision désirée).
- 3- Construire les problèmes de bornes inférieure et supérieure de $\min f(x)$ sur R , Soient LB_K et UB_K des solutions obtenues respectivement.
- 4- Si $UB_K - LB_K \leq \varepsilon$ donc on s'arrête et on pose :
 $\min f(x) = UB_K$ et $\bar{x} = x^K \in \{x : f(x) = UB_K, x \in S \cap R\}$.
- 5- Sinon, subdiviser I_K en deux sous ensembles (ou en un nombre fini de sous ensembles) R_{K_1}, R_{K_2} , tel que :

$$\bigcup_{i=1}^{i=2} \mathring{R}_{K_i} = \mathring{R} \text{ et } \mathring{R}_{K_1} \cap \mathring{R}_{K_2} = \phi$$

où $\mathring{R}$ est l'intérieur de R .

- 6- Construire les problèmes de bornes inférieure et supérieure de $\min f(x)$ sur $S \cap R_{K_i}$
 $i = 1, 2$ Soient :

$$LB_{K_1}, UB_{K_1} \text{ et } LB_{K_2}, UB_{K_2}$$

les solutions obtenues.

- 7- Posons :

$$\begin{cases} UB_{K+1} = \min\{UB_{K_1}, UB_{K_2}, UB_K\} \\ LB_{K+1} = \min\{LB_{K_1}, LB_{K_2}\} = LB_{\bar{K}} \end{cases}$$

- 8- Posons $I_K = \{R_{K_1}, R_{K_2}\}$.
- 9- Éliminer de I_K tout sous ensembles $R_{K_j}, j = 1, 2$ tel que
 $LB_{K_j} > UB_{K+1}$ où $S \cap R_{K_j} = \phi$ et posons : $I_{K+1} = R_{\bar{K}}$.
- 10- Posons : $K = K + 1$ et revenant à 4 - 10.

Notation : R_K : le sous ensemble actuel, LB_K : la borne inférieure (à la $K^{\text{ème}}$ itération), UB_K : la borne supérieure (à la $K^{\text{ème}}$ itération), x_K : la solution trouvée (à la $K^{\text{ème}}$ itération).

► Convergence de l'algorithme :

Si l'algorithme est fini. Donc, il se termine après un nombre fini d'itérations ; et UB_j est la valeur optimale de la fonction objectif, et x_j est la solution optimale de problème donné. Mais, dans le cas où le problème est infini, on doit montrer la convergence de l'algorithme de Branch and Bound.

Cet algorithme engendre une suite de subdivisions successives de sous ensembles tel que : $R_{K+1} \subset R_K$, $K \in \mathbb{N}$.

Le théorème suivant montre la convergence de l'algorithme de BB en montrons que chaque point d'accumulation de la suite des solutions est une solution optimale.

Théorème 2.3.1 [6] *Si pour chaque suite infinie $\{R_K\}$, $R_{k+1} \subset R_K$, $K \in \mathbb{N}$ des ensembles des partition successives, les bornes inférieures et supérieures vérifient :*

$$\lim_{K \rightarrow \infty} (UB_K - LB_K) = \lim_{K \rightarrow \infty} (UB_K - LB(R_K)) = 0 \quad (1)$$

Alors :

$$UB = \lim_{K \rightarrow \infty} UB_K = \lim_{K \rightarrow \infty} f(x_k) = \lim_{K \rightarrow \infty} LB_K = LB \quad (2).$$

Et chaque point d'accumulation $\bar{X}$ de la suite $\{X_K\}$ est une solution optimale de $\min f(X)$, $x \in S$.

Démonstration 2.3.1 *À l'itération K le sous-ensemble R_K sera choisit suivant la règle suivante : $LB_{K+1} = \min\{LB_{K_1}, LB_{K_2}\} = LB_{K_j}$, à la fin de l'itération $(K-1)$ et donc :*

$$LB_K = LB(R_K).$$

Soit $\{X_K\}$ la suite des solutions optimales engendrées par l'algorithme, comme S est compact, alors $\{X_K\}$ a des point d'accumulations.

Soit $\bar{X}$ un point d'accumulation de la suite $\{X_K\}$, donc il existe une sous suite infinie de $\{X_K\}$ qui converge vers $\bar{X}$, et comme f est continue alors

$$\lim_{k \rightarrow \infty} f(X_K) = f(\bar{X}).$$

Posons : $\bar{f} = \min\{f(X), X \in S\}$, la suite $\{LB_K\}$ des bornes inférieures est croissante monotone, majorée par $\bar{f}$, donc la limite $LB = \lim_{K \rightarrow \infty} LB_K$

existe.

D'autre part, la suite $\{UB_K\}$ des bornes supérieures et inférieures est décroissante minorée par $\bar{f}$, et donc sa limite : $UB = \lim_{k \rightarrow \infty} UB_K$ existe, et on a : $UB_K = f(X_K) \geq \bar{f}$, et ça implique que :

$$LB \leq \bar{f} \leq \lim_{k \rightarrow \infty} f(X_K) = f(\bar{X}) = UB.$$

Et de fait de (1) on peut déduire directement (2).

D'autres améliorations de la méthode de Branch and Bound ont été proposés parmi elles on peut citer :

2.3.1.3 La méthode d'IBB [39, 46]

Définition 2.3.2 *La méthode de Branch and Bound par intervalles : est une méthode déterministe, s'appuyant sur l'analyse d'intervalles utilisée pour résoudre les problèmes d'optimisation globale. Les premiers algorithmes (IBBA pour Interval Branch and Bound Algorithm) sont nés aux années 70, le premier algorithme fut développé par Moore-Skelboe en 1974 [40], puis l'ajout de test au point milieu en 1979 par Ichida et Fujii [41], Hansen [42], Kearfott [43], Ratz [44] et beaucoup d'autres continuant à développer l'algorithme en incluant des tests de monotonies, de concavité, etc. Elle est l'une des méthodes rares dans ce domaine permettant d'avoir une preuve mathématique que le résultat trouvé est bien le minimum global recherché à la précision près [45, 46].*

Principe de la méthode :

L'algorithme de la méthode Branch and Bound par intervalles a beaucoup évolué, mais il garde toujours le même principe de la méthode de Branch and Bound classique. Le but, c'est de trouver un encadrement de largeur inférieure à la précision demandée.

Prenant :

$$(P) \begin{cases} \min f(x) \\ x \in [X] \end{cases}$$

On découpe l'intervalle $[X]$ en deux intervalles, puis on calcule la borne inférieure et supérieure de chacun des deux intervalles. Si la borne inférieure de l'un des deux intervalle est supérieure à la borne supérieure de deuxième. Alors, on élimine le deuxième (puisque un tel sous ensemble ne peut pas contenir le minimum global), et on garde le premier. Si ce n'est pas le cas, on garde les deux intervalles dans la listes des intervalles considérés. On réitère le procédés, jusqu'à obtenir le minimum global avec la précision voulue. (cf. Figure 2.2)

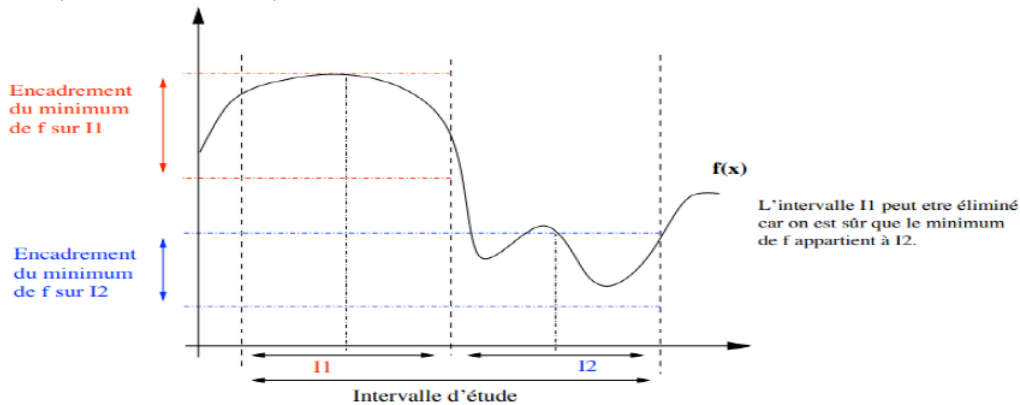


FIGURE 2.2 – Principe de l'algorithme IBBA.

Algorithme de la méthode IBB (IBBA : Interval Branch and Bound Algorithm) :

1. Initialiser $[x] :=$ domaine initial contenant le minimum global.
 2. Initialiser $\tilde{f} := +\infty$: valeur courante de la borne supérieure du minimum.
 3. Initialiser $\tilde{y} := [x]$: valeur courante du minimum de f .
 4. Initialiser la liste $L := (-\infty, [x])$: structure qui va contenir tous les éléments qui restent encore à traiter.
- $\forall (z, Z) \in L$, z représente une borne inférieure de f sur Z .
5. Extraire de L l'élément ayant la plus petite borne inférieure $\min_{(z, Z) \in L} Z$.
 6. Découper l'intervalle Z associé en deux intervalles, donnant V_1 et V_2 .
 7. Pour $j := 1$ à 2 faire
 - (a) Calculer $v_j := \text{lower bound}(f, v_j)$ (calcul d'une borne inférieure de f sur V_j).
 - (b) Si $\tilde{f} \geq v_j$ et si aucune contrainte n'est insatisfaite sur V_j alors :
 - Insérer v_j, V_j dans L .
 - Soit m milieu de V_j , si m satisfait toutes les contraintes alors
 - Si $\tilde{f}$ a changé alors

Supprimer de L tous les éléments (z, Z) tel que $z > \tilde{f}$
 et faire $\tilde{y} := m$
 8. Si $\tilde{f} - \max_{(z, Z) \in L} Z < \varepsilon$ alors stop.
 Sinon retourner à l'étape 5.

2.3.1.4 La méthode α BB [48, 52] :

Définition 2.3.3 *La méthode α BB : est une approche déterministe utilisée pour la résolution des problèmes d'optimisation continus (uniquement avec des variables continues), impliquant la non convexité dans la fonction objectif et/ou les contraintes. Les non convexités sont classées comme étant des structures spéciales et génériques. Une relaxation convexe des problèmes non convexes est atteint en remplaçant les termes généraux non convexes d'une structure spéciale (bilinéaire, fractionnaire,), par des fonctions borne inférieure serrées, en utilisant un paramètre α défini en [48] , et calculer une borne inférieure des termes de la structure générique.*

Principe de la méthode :

Soit le problème d'optimisation non convexe continu suivant :

$$(P_0) \begin{cases} \min_x f(x) \\ h_j(x) = 0, j = 1, \dots, M \\ g_k(x) \leq 0, k = 1, \dots, K \\ Ax \leq c \\ x^L \leq x \leq x^U \end{cases}$$

où X : est le vecteur des variables,

$f(x)$: est une fonction objectif non linéaire de classe C^2 ,

$h_j(x)$: est un ensemble de contraintes d'égalités non linéaires,

et $g_k(x)$, $\forall k = 1, \dots, K$ est un ensemble de contraintes d'inégalités non linéaires.

Notre approche est basée sur la **relaxation convexe** des termes non convexes de (P_0) . Cela nécessite la borne inférieure de tous les termes non convexes apparaissant dans P_0 .

Ces termes peuvent être divisés en trois classes :

- convexes,
- non convexe avec un structure spéciale,
- non convexe avec une structure générique.

Il est claire qu'aucune action de limitation n'est requise pour les fonctions convexes. Pour les fonctions non convexes (de structure spéciale ou générique), on applique l'action de limitation. (Voir plus de détails en [49])

Algorithme de la méthode :

Étant donné une séparation maximale entre les termes non convexes et leurs fonctions de borne inférieure convexes respectives sont des fonctions bornées continues de taille de domaine rectangulaire, des tolérance de faisabilité ε_f et de convergence ε_c arbitrairement petites sont atteintes pour un éléments de partition de taille finie.

Les étapes de base de l'algorithme d'optimisation globale proposé sont les suivantes :

Étape 1 : initialisation :

Une tolérance de convergence ε_c , et une tolérance de faisabilité ε_f , sont sélectionnées et le compteur d'itérations $Iter$ est mis à un. Les bornes des variables $x^{L,Iter}$, $X^{U,Iter}$ pour la première itération courante sont définies pour être égales X^{LBD} , X^{UBD} . Bornes inférieure et supérieure LBD, UBD sont initialisées au minimum global de (P) et un point courant initial $X^{c,Iter}$ est sélectionné.

Étape 2 :

Solution locale de PNL non convexe et mise à jour de la borne supérieure G_0^{UBD} comme suit : $UBD = \min(UBD, f_{local}^{Iter})$.

Étape 3 :

Partitionnement de rectangle actuel $[X^{L,Iter}, X^{Iter}]$ en deux rectangles ($r = 1, 2$) comme suit :

$$\begin{bmatrix} x_1^{L,Iter} & x_1^{U,Iter} \\ \vdots & \vdots \\ x_{l^{Iter}}^{L,Iter} & \frac{(x_{l^{Iter}}^{L,Iter} + x_{l^{Iter}}^{U,Iter})}{2} \\ \vdots & \vdots \\ x_N^{L,Iter} & x_N^{U,Iter} \end{bmatrix}, \begin{bmatrix} x_1^{L,Iter} & x_1^{U,Iter} \\ \vdots & \vdots \\ \frac{(x_{l^{Iter}}^{L,Iter} + x_{l^{Iter}}^{U,Iter})}{2} & x_{l^{Iter}}^{U,Iter} \\ \vdots & \vdots \\ x_N^{L,Iter} & x_N^{U,Iter} \end{bmatrix}$$

où l^{Iter} correspond à la variable de plus grand côté dans le rectangle initial,

$$l^{Iter} = \arg \max_i (x_i^{U,Iter}, -x_i^{L,Iter})$$

Étape 4 :

Mettre à jour de $\alpha_{i,k}^j(x^{U,Iter}, x^{L,Iter})$ est à l'intérieure des deux rectangles $r = 1, 2$)

Étape 5 :

La solution de R à l'intérieure de deux rectangles ($r = 1, 2$) en utilisant n'importe quel solveur non linéaire convexe (par exemple MINOS5.4 [50]).

Étape 6 :

Mettre à jour le compteur d'itération $Iter$ et la limite inférieure LBD tel que :

$$Iter \leftarrow Iter + 1$$

$$LBD = l_{sol}^{r', Iter'}$$

$$\text{où } l_{sol}^{r', Iter'} = \min_{r, I} l_{sol}^{r, I}, \quad r = 1, 2, \quad I = 1, \dots, Iter - 1.$$

Étape 7 :

Mettre à jour le point courant $x^{c, Iter}$ et les bornes actuelles $x^{L, Iter}, x^{U, iter}$ en x pour être la solution de minimum trouvé précédemment solution à l'étape 6,

$x^{c, Iter} = x_{sol}^{r', Iter'}$, et le rectangle courant devient le sous rectangle contenant la solution trouvée précédemment,

$$\begin{aligned}
 [x^{L,Iter}, x^{U,Iter}] &= \begin{bmatrix} x_1^{L,Iter'} & x_1^{U,Iter'} \\ \vdots & \vdots \\ x_{l^{Iter'}}^{L,Iter'} & \frac{(x_{l^{Iter'}}^{L,Iter'} + x_{l^{Iter'}}^{U,Iter'})}{2} \\ \vdots & \vdots \\ x_N^{L,Iter'} & x_N^{U,Iter'} \end{bmatrix}, \text{ si } r' = 1 \\
 [x_i^{L,Iter}, x_i^{U,Iter}] &= \begin{bmatrix} x_1^{L,Iter} & x_1^{U,Iter'} \\ \vdots & \vdots \\ \frac{(x_{l^{Iter'}}^{L,Iter'} + x_{l^{Iter'}}^{U,Iter'})}{2} & x_{l^{Iter'}}^{U,Iter'} \\ \vdots & \vdots \\ x_N^{L,Iter'} & x_N^{U,Iter'} \end{bmatrix}, \text{ si } r' = 2
 \end{aligned}$$

Étape 8 : Chercher la convergence.

Si $(UBD - LBD) > \varepsilon_c$, donc retourner à l'étape 2. Sinon, la convergence ε_c est atteinte et la solution minimale globale et le point de solution sont :

$$\begin{aligned}
 \bar{f} &\leftarrow f^{c,Iter''} \\
 \bar{x} &\leftarrow x^{c,Iter''}
 \end{aligned}$$

Où $Iter'' = \arg\{f^{c,I} = UBD, I = 1, \dots, Iter\}$.

Convergence de l'algorithme :

Démonstration 2.3.2 *La preuve de convergence de notre algorithme vers le minimum global est basée sur l'analyse des algorithmes de l'optimisation globale déterministes standards présentés dans [51] comme indiqué dans [48] et [52].*

2.3.1.5 La méthode LGO (Lipschitzien global optimization method) [27], [53] :

Soit (X, d) un espace métrique et f une fonction lipschitzienne de constante L sur X . La propriété suivante est donc vérifiée :

$$\forall x, y \in X, \quad \|f(x) - f(y)\| \leq Ld(x, y).$$

Si nous arrivons à choisir les points $x_1, x_2, \dots, x_k$. On pose $\bar{x}_k^* =$

$\min\{f(x_1), f(x_2), \dots, f(x_k)\}$, et on désigne par $(B(x_i, \rho_{i,k}))_{0 \leq i \leq k}$ la famille des boules de centres x_i et de rayons $\rho_{i,k} = \frac{f(x_i) - \bar{x}_k^* + \varepsilon}{L}$ pour $0 \leq i \leq k$ avec ε un nombre positif donné. Alors pour tout $i \in \{1, 2, \dots, k\}$ et pour tout $x \in B(x_i, \rho_{i,k})$ on a :

$$d(x, x_i) \leq \frac{f(x_i) - \bar{x}_k^* + \varepsilon}{L},$$

alors

$$Ld(x, x_i) \leq (f(x_i) - \bar{x}_k^* + \varepsilon),$$

donc

$$f(x_i) - f(x) \leq (f(x_i) - \bar{x}_k^* + \varepsilon),$$

ce qui est équivalent à

$$\bar{x}_k^* - \varepsilon \leq f(x)$$

Il en résulte que si $X \subset \bigcup_{i=1}^k B(x_i, \rho_{i,k})$, $\bar{x}_k^*$ est un minimum global de f avec la précision ε .

Théorème 2.3.2 *Soit f est une fonction lipschitzienne de constante L , définie sur un compact X de l'espace $(\mathbb{R}^n, d)$. supposons que la fonction f est évaluée aux points $x_1, x_2, \dots, x_k \in X$ et on pose $\bar{x}_k^* = \min\{f(x_1), f(x_2), \dots, f(x_k)\}$. Si on a $X \subset \bigcup_{i=1}^k B(x_i, \rho_{i,k})$ alors $\bar{x}_k^*$ est un minimum global de f avec une précision ε .*

Démonstration 2.3.3 *Pour la démonstration de ce théorème voir [27].*

Il existe plusieurs méthodes lipschitziennes en optimisation globale; notamment : la méthode d'Evtushenko et la méthode de Piyavskii-Shubert ...

Dans ce mémoire, on traitera la méthode d'Evtushenko.

Définition 2.3.4 *La méthode d'Evtushenko est une méthode déterministe. C'est l'une des méthodes de recouvrement la plus connue, la plus commentée en littérature; développée par Evtushenko (1971-1985).*

Elle utilise comme recouvrement, les cubes définis par :

$$S_{jk} = \{x \in \mathbb{R}^n / d_{\max}(x_j, x) = |x - x_j|_{\infty} \leq r_{j,k}\}$$

avec

$$r_{j,k} = \frac{f(x_j) - R_k + \varepsilon}{L\sqrt{n}}$$

où R_k est le record d'ordre k définie par : $R_k = \min\{f(x_1), f(x_2), \dots, f(x_k)\}$. Dans le cas unidimensionnel ($X = [a, b]$) et pour une fonction lipschitzienne f de constante L , et d'après **le théorème 2.3.2** il faut couvrir avec des intervalles de rayon $\rho_{i,k} = \frac{f(x_i) - R_k + \varepsilon}{L}$ qui sont les centres des intervalles $\{I_i\}_{1 \leq i \leq k}$ tel que $X \subset I_i$. Définir x_{i+1} ainsi : $x_{i+1} = \frac{f(x_i) - \bar{x}_k^* + \varepsilon}{L} + \frac{\varepsilon}{L}$ (mieux détaillé en [53]).

Principe de la méthode :

Cette méthode se base sur le principe de recouvrement. Dans le cas unidimensionnel, dans le cas où D est un intervalle $[a, b]$ de $\mathbb{R}$, d'après **le théorème 2.3.2**, on peut couvrir l'intervalle $[a, b]$ par des sous intervalles $\{I_i\}_{1 \leq i \leq k}$ de centres $\{x_i\}_{1 \leq i \leq k}$ et de rayons $r_{ik} = (f(x_i) - R_k + \varepsilon)/L$ tel que $[a, b] \subset \bigcup_{i=1}^k I_i$.

Puisque le plus petit des rayons $r_{i,k}$ est $\frac{\varepsilon}{L}$, on prend $x_1 = a + \frac{\varepsilon}{L}$ qui est le centre de l'intervalle $I_1 = \left[a, a + \frac{2\varepsilon}{L}\right]$ et de rayon $r_{11} = \frac{\varepsilon}{L}$, parce que à l'initialisation $f(x_1) = \bar{x}_1$. Avec cette valeur de x_1 , on gagne le temps de ne pas ignorer le minimum global au voisinage de ce point. En effet si $\bar{x} = \arg \min_{x \in [a, b]} f(x) \in \left[a, a + \frac{\varepsilon}{L}\right]$ alors :

$$\|f(x_1) - f(\bar{x})\| \leq Ld(x_1, \bar{x}) \leq L\frac{\varepsilon}{L}.$$

Pour le point x_2 qui est le centre de l'intervalle I_2 de rayon $r_{22} = (f(x_2) - \bar{x}_2 + \varepsilon)/L$, on devait prendre $x_2 = x_1 + r_{11} + r_{22}$. Or r_{22} est inconnu mais on sait que $r_{22} > \frac{\varepsilon}{L}$, on prend alors $x_2 = x_1 + r_{11} + \frac{\varepsilon}{L} = x_1 + \frac{2\varepsilon}{L}$.

En générale, pour $i \geq 1$, la suite $\{x_{i+1}\}$ est définie par :

$$x_{i+1} = x_i + (f(x_i) - \bar{x}_k + \varepsilon)/L + \frac{\varepsilon}{L} = x_i + (f(x_i) - \bar{x}_k + 2\varepsilon)/L.$$

Ce choix nous permet de ne rater le minimum global de f , car les intervalles I_i se rencontrent. On arrête quand k vérifie $D \subset \bigcup_{i=1}^k I_i$.

Si $x_k < b$ et $x_k + r_{kk} \geq b$, alors le dernier point de la suite est x_k .

Si $x_k + r_{kk} < b$ et $x_k + r_{kk} + \frac{\varepsilon}{L} \geq b$ le dernier point de la suite sera $x_{k+1} = b$.

Algorithme de la méthode :

L'algorithme de la méthode d'Evtushenko dans le cas unidimensionnel est le suivant :

1) Initialisation :

Poser $x_1 = a + \frac{\varepsilon}{L}$, $x_\varepsilon = x_1$, $f_\varepsilon = f(x_\varepsilon)$

2) Étape $k = 2, 3, \dots$

Poser $x_{k+1} = x_k + \frac{2\varepsilon + f(x_k) - f_\varepsilon}{L}$

Si $x_k > b$ alors, arrêter

Sinon calculer $f(x_{k+1})$

Si $f(x_{k+1}) < f_\varepsilon$ alors,

Poser $f_\varepsilon = f(x_{k+1})$ et $x_\varepsilon = x_{k+1}$

Poser $k = k + 1$

Aller à l'étape 2.

Méthode de Hölder mixée (combinée) (BB) Soit le problème d'optimisation globale suivant :

$$(P) \begin{cases} \min f(x) \\ x \in I \end{cases}$$

$I = [a, b] = I_0$: intervalle de $\mathbb{R}$, f est continue et vérifie la propriété de Hölder suivante : $\forall x \in I, \forall y \in I, |f(x) - f(y)| \leq L|x - y|^{\frac{1}{\alpha}}$;

où L : est la constante (> 0) de Lipschitz, $\alpha \geq 1$.

► L'algorithme de Hölder dans $\mathbb{R}$:

L'algorithme de base de Hölder s'appuie sur la construction de fonctions bornes inférieures, combinées à la méthode de séparation et d'évaluation.

Pour plus de détails sur l'algorithme voir [5]

Remarque 2.3.1 *Il existe une autre méthode lipschitzienne suivant le même principe de recouvrement (Piyavskii-Shubert [53]).*

Les problèmes généraux de l'optimisation non linéaire s'avèrent difficiles

à résoudre avec les approches déterministes. Ce qui explique le passage aux méthodes stochastiques.

2.3.2 Méthodes stochastiques [7], [17], [27]

Les méthodes stochastiques également appelées les méthodes approchées ou non déterministes : sont des approches appliquées sur les problèmes d'optimisation avec ou sans contraintes, et sans structure particulière. Elles s'applique également dans le cas où la fonction objectif est donnée sous forme "d'une boîte noire" , donc le problème de l'optimisation globale est particulièrement difficile à résoudre avec les approches exactes. Ces approches sont rapides par rapport aux méthodes déterministes, et leurs résultats est garanti avec une probabilité proche de 1.

2.3.2.1 Méthode de Törn (Multistart) [26], [27]

La méthode de Multistart également appelée la méthode de Törn : est une approche basée sur les méthodes d'optimisation locale. Elle consiste à faire des recherches multiples des minima locaux, à partir des point initiaux choisis parmi les éléments d'un maillage uniforme⁶.

Si les minima locaux sont éloigner alors, on fait les modifications suivantes à la méthode de Törn :

1)- Localiser les minimiseurs locaux par des voisinages.

Ensuite, appliquer la méthode de descente.

Si l'un des points de recherche atteint ce voisinage, donc l'algorithme converge vers ce minimum local.

2)- Application de la méthode des points candidats, qui consiste à faire des descentes simultanées à partir des point initiaux, en remplaçant les point voisins par celui qui a la plus petite valeur de la fonction objectif.

► Inconvénients de la méthode :

Pour être sûr de trouver le minimum global avec la précision voulue, le nombre de points initiaux à prendre dans cette approche, doit être plus grand que le nombre des minimum locaux (ce dernier n'est pas toujours

6. Un maillage uniforme : est une déscription spatiale d'un milieu continu, ou aussi une modélisation géométrique des éléments partitionnés finis et bien définis.

connu).

La difficulté de choisir les voisinages qui sont des domaines d'attractions des minimiseurs locaux.

2.3.2.2 Méthode de recuit simulé [28], [29], [30], [33], [34], [35], [36]

Définition 2.3.5 *La méthode de recuit simulé est conçue pour la résolution des problèmes d'optimisation combinatoire, et essentiellement pour solutionner les problèmes de la recherche opérationnelle i.e. les problèmes de maximisation ou de minimisation d'une quantité. Apparue pour la première fois dans la revue de science de Kirpatrick (Kirkpatrick & al 93), développée simultanément par trois chercheurs de la société IBM, S.Kirkpatrick, C.D.Gellatt et M.P.Vecchi en 1983 [33], et Cerny en 1985 [34]. Elle est obtenue par analogie au phénomène thermodynamique de recuit des métaux (Salomon.M 2001) (Bosi.H.C 96), c'est à dire pour atteindre les états de basse énergie d'un solide, on chauffe celui-ci à une température très élevée puis on le laisse refroidir lentement.*

- Principe de la méthode :

- On considère un système à N éléments.
- À chaque configuration on associe une fonction à optimiser (suivant le critère de Metropolis [30]).
- Si l'on considère une configuration qui minimise la fonction, celle-ci joue le rôle de l'énergie.
- Notre point de départ, est une configuration choisie astucieusement en fonction du problème.
- À l'itération initiale, on fixe un paramètre de température en relation avec la gamme des énergies accessibles de système.
- On itère le processus suivant ; dit de Metropolis ou de Monte Carlo (dérivée de probabilité de Boltzmann [30]) :
 - On tire au sort une modification de la configuration actuelle qui est effectuée par des changements discrets de l'état des composants du système, qui change l'énergie de système.

- Si l'énergie diminue, on effectue le changement.
- L'itération se poursuit tant que l'énergie du système diminue. Lorsque l'énergie reste stationnaire, on diminue un peu la température et l'on reprend le processus de Monte-Carlo de décroissance de l'énergie.
- On arrête lorsque les diminutions de température restent inefficaces.

• Algorithme de la méthode :

La méthode repose sur l'algorithme de Metropolis [30]. Son point crucial est la loi de décroissance de température. De nombreuses lois de décroissances ont été proposées[36].

L'algorithme est implémenté comme suit :

- 1)-(Déplacement) : partir de la solution courante vers une nouvelle solution.
- 2)-(Évaluation) : Évaluer le coût de la nouvelle solution.
- 3)-(Acceptation) : décider d'accepter ou de rejeter une nouvelle solution.
- 3)-(Test d'arrêt) : si le test d'arrêt est vérifié arrêter , sinon, revenir à 1).

La fonction de d'acceptation peut être implémentée comme s'est détaillé en [28].

• Convergence de l'algorithme :

Soit $J(p_1, p_2, \dots, p_n)$ la fonction coût. Soit $G(V, E)$ son graphe, on peut construire un graphe direct $G(V, E)$ correspondant au problème d'optimisation donné. Soit T la température minimale que le recuit simulé a accédé. Et soit

$$\delta = \max_{i \in V, j \in N(i)} \{J(i) - J(j)\}$$

On note respectivement le degré et le diamètre de $G(V, E)$ par d , D .

Lemme 2.3.1 *Si X est un état de V , donc l'espérance du nombre d'itération avant que l'optimum global soit visité est $\leq (\frac{1}{d} \exp(-\delta/T))^{-D}$.*

Théorème 2.3.3 *Le recuit simulé converge dans un temps $\leq 2k [d \exp(-\delta/T)]^D$, avec une probabilité $\geq (1 - 2^{-k})$.*

Remarque 2.3.2 *Pour les démonstrations du lemme, du théorème et pour plus de détails et de preuves de résultats, nous vous invitons à voir [35].*

2.3.2.3 Les algorithmes génétiques

- Cas des problèmes sans contraintes [28] [29], [30], [31], [32] :

Définition 2.3.6 *L'algorithme génétique : est dérivé de la génétique et l'évolution naturelle de l'être vivant. C'est un algorithme stochastique, inventé et développé par John Holland dans son livre [31], et utilisé la première fois par Goldberg [32]. Il consiste à chercher le ou les extrema d'une fonction définie sur un espace de données. D'une autre manière, chercher les meilleur individus "parents", à produire les meilleurs descendants "enfants", pour tirer une solution de problème traité. Il s'appuie sur cinq principes (Holland .J.H.75), (Salomon.M2011), (Barbay.J.98).*

- Le principe des algorithmes génétiques :
 - Un principe de codage de l'élément de la population : associer à chaque point de l'espace des états une structure de données.
 - Un mécanisme de génération de la population initiale : Capacité de produire une génération initiale capable de produire des générations futures. Ce choix est important car il influence sur la convergence vers le minimum global
 - Une fonction à optimiser : appelée fitness ou fonction d'évaluation de l'individu.
 - Opérateurs pour diversifier la population et explorer l'espace des états : recomposer les gènes par les opérateurs de croisement.
 - L'exploration de l'espace des états par l'opérateur de mutation.

- Paramètres de dimensionnement : taille de population, nombre total de générations, critère d'arrêt...
- Algorithme génétique : Générer une population initiale aléatoirement de N individus P_0 , en considérant que le test d'arrêt est un nombre fini d'itérations .

Le processus est le suivant :

- 1)- Codage de chaque individu de la population P_k .
- 2)- Croisement de tout les individus de la population deux par deux avec une probabilité P_m on aura N enfants notés C_k .
- 3)- Mutation de tout les individus de la population, on aura N éléments notés M_k .
- 4)- Choisir les N meilleurs éléments de l'ensemble $P_k \cup C_k \cup M_k$ c.à.d ceux qui ont une valeur inférieure de la fonction objectif, et de les mettre dans la population P_{k+1} .
- 5)- Si le test d'arrêt est vérifié arrêter, sinon revenir à l'étape 1.

Remarque 2.3.3 *On a un autre critère d'arrêt des algorithmes génétiques, c'est quand la population n'évolue plus suffisamment.*

- Cas des problèmes avec contraintes [28] :

Les problèmes de la vie réelle sont soumis à une infinité ou certains nombre de contraintes.

Les contraintes : sont des relations exprimés soient par des égalités ou des inégalités, que les variables de la fonction à minimiser doivent respectées.

Dans le cas sans contraintes, les phénotypes⁷ engendrés sont aveugles, et peuvent ne pas être réalisables, d'où l'inconvénient des algorithmes génétiques sans contraintes.

Pour respecter ces contraintes des approches sont appliquées sur les opérateurs :

7. En génétique un phénotype est l'ensemble des caractéristiques ou traits observables d'un organisme.

- Application de la méthode de la peine de mort (death penalty method) qui consiste à calculer la fonction d'adaptation dans l'espace réalisable, et associer une adaptation nulle à ceux qui sont dans l'espace irréalisable. Cette méthode peut être assimilée à la méthode de pénalité avec pénalité infinie.
- Pour avoir une solution plus élaborée et plus efficace, on doit calculer la fonction d'adaptation dans les deux espaces réalisable et irréalisable. Ceci se fait en rajoutant une pénalité faible qui donne des informations sur le nombre et l'importance des contraintes violées, et la pénalité forte qui risque de faire la convergence prématurée de notre algorithme.
- Utilisation d'un algorithme de réparation à fin de rendre tous les phénotypes irréalisables réalisables et ceci en remplaçant les opérateurs de recherche standard par des opérateurs dépendant de problème; à fin que le problème ne produise que les génotypes descendant de phénotype réalisables.
- Ramener un problème d'optimisation avec contraintes à un problème d'optimisation sans contraintes. En considérant les contraintes comme des fonctions objectifs prioritaires que le critère à optimiser.

Dans notre cas, on utilise la méthode de correction [28].

2.3.2.4 La recherche Tabou [29], [30], [47]

Définition 2.3.7 *La méthode de la recherche Tabou : est une métaheuristique développée par Glover en 1986 [47]. C'est une variante de la recherche locale, utilisée spécifiquement pour la résolution des problèmes d'optimisation combinatoire. Elle permet de trouver un compromis entre la qualité de la solution et le temps de calcul. Le principe c'est d'ajouter au processus de recherche une mémoire flexible, qui permet de mener une "recherche plus intelligente" dans l'espace des solutions.*

Le principe de la méthode :

Déplacer d'une solution X vers une solution Y située dans V voisinage de X par l'exploration de l'espace de recherche. Au début, le vecteur pas couvre tout l'espace de recherche puis il diminue à chaque itération. Parmi tout les points générer de voisinage V de X , on prend le meilleur, qui est celui qui améliore le plus la fonction objectif ou celui qui la dégrade le moins. Pour éviter l'exploration des solutions déjà visitées, l'algorithme possède une mémoire, qui garde la trace des solution déjà visitées (liste taboue), puisque cette dernière peut être longue d'où la difficulté de la recherche. Une nouvelle solution est acceptée si elle n'appartient pas à la liste taboue. Pour plus de détails sur le principe, et pour plus d'explication sur le principe de la méthode tabou pour les variables continues référer à [30].

L'algorithme de la méthode Tabou :

Les caractéristiques essentiels d'une recherche Tabou sont résumés dans l'algorithme suivant :

Calculer un sous voisinage v^* de V . La meilleure solution y^* dans v^* est la nouvelle solution courante. Lorsque aucun mouvement n'améliore la solution actuelle, on visite la solution précédente, donc il est interdit d'exploiter des solutions récemment visitées, et ceci se fait en éliminant les solutions déjà visitées, en utilisant une ou plusieurs listes Tabou.

2.3.3 La méthode Branch and Bound avec une fonction borne inférieure quadratique [21]

Les auteurs dans leur article [21] ont proposé un algorithme efficace pour la résolution des problèmes d'optimisation globale univariés non convexes avec une fonction et une contrainte deux fois différentiables (non convexes), et leurs secondes dérivées sont bornées i.e. ils existent des nombres positif K et K_g tel que : $|f''(s)| \leq K$, $|g''(s)| \leq K_g$, $\forall s \in S$.

Cette contrainte dans le cas simple est sous la forme d'un intervalle. Et en générale, c'est une contrainte supplémentaire non convexe et deux fois différentiable. Ils utilisent une fonction borne inférieure quadratique pour

la procédure d'évaluation ; et la w-subdivision pour la procédure de branchement.

Cet algorithme est basé sur la méthode de Branch and Bound qu'on va adapter dans tout ce qui s'en suit.

Dans cette synthèse, on commencera par l'application de la méthode BB sur les problèmes de type (P) , puis ceux de type (PC) .

- Un problème d'optimisation globale univarié avec une contrainte simple, s'écrit comme suit :

$$(P) \begin{cases} \alpha := \min f(s) \\ s \in S, \end{cases}$$

avec $S := [a, b]$ est intervalle fermé borné, et f est continue sur S .

On ne considère que le cas où la fonction f n'est pas concave ; car sinon le problème est trivial. Ainsi, $\min \{f(s) : s \in [a, b]\} = \min \{f(a), f(b)\}$.

- Un problème d'optimisation globale univarié avec une contrainte supplémentaire non convexe s'écrit comme suit :

$$(PC) \begin{cases} \alpha_g := \min f(s) \\ s \in S \\ g(s) \leq 0 \end{cases}$$

où g est une fonction non convexe sur S .

► Cas des problèmes avec une contrainte simple (P) :

Dans cette partie, nous appliquons la méthode BB sur les problèmes simples (P) , avec une contrainte d'intervalle.

- Procédure d'évaluation

Les auteurs utilisent pour la procédure d'évaluation, la fonction borne inférieure quadratique convexe.

Définition 2.3.8 *La fonction borne inférieure quadratique convexe : est une minorisation convexe de la fonction f sur $[a_k, b_k]$ qui est meilleure que la fonction borne inférieure affine l_k . (voir [21])*

Théorème 2.3.4 On a :

$$L_h f(s) - \frac{1}{8} K h^2 \leq q_k(s) := L_h f(s) - \frac{1}{2} K (s - a_k)(b_k - s) \leq f(s), \quad \forall s \in [a_k, b_k]$$

Démonstration 2.3.4 On renvoie le lecteur à consulter l'article [21]

Remarque 2.3.4 Par facilité de calcul de K_k et puisque il est plus petit par rapport à K . Donc, la procédure d'évaluation q_s ; génère une borne inférieure meilleure. Ainsi, il est préférable de remplacer K par K_k pour chaque intervalle $[a_k, b_k]$ avec $\|f''(s)\| \leq K_k$, $K_k \geq 0$.

Notons $LB(T^k)$ la meilleure borne inférieure de f sur T_k , calculée en résolvant le programme convexe suivant :

$$LB(T^k) := \min \{q_k(s) : a_k \leq s \leq b_k\}. \quad (1)$$

Et q_k est une fonction quadratique de la forme :

$$q_k(s) := \frac{K}{2} s^2 + \left[\frac{f(b_k) - f(a_k)}{b_k - a_k} - \frac{K}{2} (b_k + a_k) \right] s + \frac{K}{2} a_k b_k + \frac{f(a_k)b_k - f(b_k)a_k}{b_k - a_k} \quad (2),$$

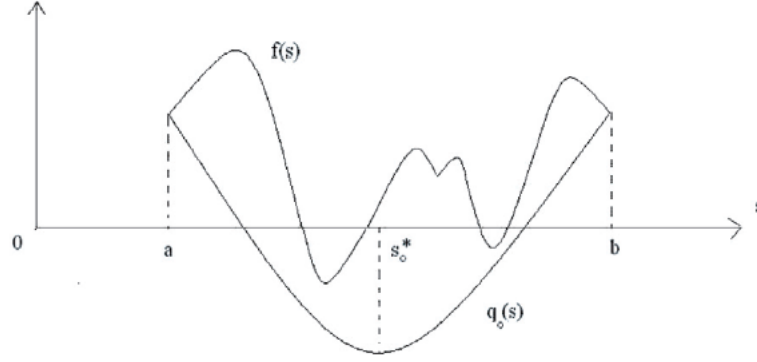
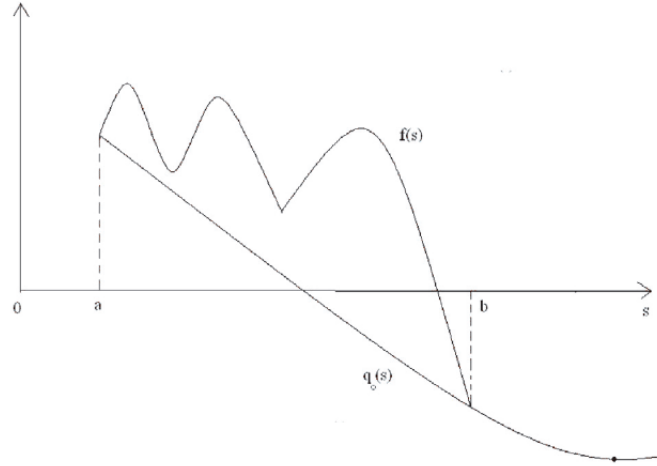
la solution optimale s_k^* est défini comme suit :

$$s_k^* = \begin{cases} \mu := \frac{1}{2}(a_k + b_k) - \frac{1}{K_h}(f(b_k) - f(a_k)) & \text{si } \mu \in [a_k, b_k], \\ a_k & \text{si } \mu < a_k, \\ b_k & \text{si } \mu > b_k. \end{cases} \quad (3)$$

Remarque 2.3.5 Si $s_k^* \notin]a_k, b_k[$, alors :

$$\min \{q_k(s) : a_k \leq s \leq b_k\} = \min \{f(s) : a_k \leq b_k\} = \min \{f(a_k), f(b_k)\}.$$

Dans un tel cas on a une évaluation exacte sur l'intervalle T^k (voir **la figure 2.4**). Par conséquent, T^k est supprimé de la liste des intervalles qui seront considérés dans les prochaines étapes de l'algorithme de branchement et de liaison.


 FIGURE 2.3 – La fonction borne inférieure de f , cas 1.

 FIGURE 2.4 – La fonction borne inférieure de f , cas 2.

• Procédure de branchement

Pour la procédure de branchement, ils utilisent la w -subdivision, qui est la meilleure procédure de subdivision rectangulaire bien connu par rapport à la subdivision normale (subdivision utilisant le point médian). De plus, elle est très efficace, car on obtient une évaluation exacte lors de calcul de la borne inférieure.

Définition 2.3.9 w -subdivision est une procédure de bisection, introduite par Falk-Soland [24] pour les problèmes de la programmation séparable non convexe. Elle consiste à choisir un indice i_k tel que :

$$i_k \in \arg \max_i \{f_i(x_i^*)q_i(x_i^*)\}$$

où $q(x) := \sum_{i=1}^n q_i(x_i)$: Fonction de borne inférieure séparable,

$\sum_{i=1}^n f_i(x_i)$: Fonction objectif séparable.

Cette procédure consiste à diviser le rectangle T_k à l'itération k en deux sous-rectangles tel que :

$$T_1^k = \{x \in T^k : x_{ik} \leq x_i^*\}, T_2^k = \{x \in T^k : x_{ik} \geq x_i^*\},$$

où x^* est un minimum de q sur T^k .

Ainsi, le rectangle T^k est divisé par s_k^* c'est-à-dire $T_1^k = T_1^k \cap T_2^k$, où $T_1^k = [a_k, s_k^*]$, $T_2^k = [s_k^*, b_k]$, avec s_k^* est un point donné en (3).

• Algorithme de la méthode de Branch and Bound :

Notons par LB_k , UB_k et s^k , la meilleure borne inférieure, la meilleure borne supérieure et la meilleure solution trouvée de (P), à l'itération K .

1. *Initialisation :*

a) Soit ε un nombre donné suffisamment petit, et soit K est une constante vérifiant $K \geq |f''(s)|$ pour tout s dans $[a, b]$.

b) Poser $k := 0$, $T^0 = [a, b]$, $M := \{T^0\}$, $h_0 = a - b$, $s_0^* = \frac{1}{2}(a + b) - \frac{1}{K(b-a)}(f(b) - f(a))$.

c) Poser $UB_0 = \min \{f(a), f(b), f(s_0^*)\}$.

d) Poser $LB_0 := q_0(s_0^*)c = \frac{K}{2}s_0^* + \left[\frac{f(b) - f(a)}{b - a} - \frac{K}{2}(b + a) \right] s_0^* + \frac{K}{2}ab + \frac{f(a)b - f(b)a}{b - a}$.

2. Tant que $UB_k - LB_k > \varepsilon$ faire

3. Soit $T^k = [a_k, b_k] \in M$ un intervalle tel que $LB_k = LB(T^k)$.

Soit K_k vérifiant $K_k \geq |f''(s)|$ pour tout s dans $[a_k, b_k]$.

4. Coupez T^k en deux intervalles par la procédure de w-subdivision :

$$T_1^k = [a_k, s_k^*] := [a_k^1, b_k^1], T_2^k = [s_k^*, b_k] := [a_k^2, b_k^2].$$

5. Pour $i = 1, 2$ faire

5.1. Poser $s_{k,i}^* = \frac{1}{2}(a_k^i + b_k^i) - \frac{f(b_k^i) - f(a_k^i)}{K_k(b_k^i - a_k^i)}$

5.2. Si $s_{k,i}^* \notin]a_k^i, b_k^i[$, alors mettre à jour $LB(T_i^k)$ tel que

$$\begin{aligned} LB(T_i^k) &= UB(T_i^k) \\ &= \min \{f(a_k^i), f(b_k^i)\} \end{aligned}$$

sinon définir

$$LB(T_i^k) := \frac{K_k}{2} s_{k,i}^{*2} + \left[\frac{f(b_k^i) - f(a_k^i)}{b_k^i - a_k^i} - \frac{K}{2} (b_k^i + a_k^i) \right] s_{k,i}^* + \frac{K_k}{2} a_k^i b_k^i + \frac{f(a_k^i) b_k^i - f(b_k^i) a_k^i}{b_k^i - a_k^i}.$$

5.3. Pour rentrer dans M les intervalles $T_i^k : M \leftarrow M \cup \{T_i^k : UB^k - LB(T_i^k) \geq \varepsilon, i = 1, 2\} \setminus \{T^k\}$.

5.4. Mettre à jour $UB_k = \min \{UB_k, f(a_k^i), f(b_k^i), f(s_{k,i}^*)\}$.

6. Mettre à jour $LB_k = \min \{LB(T) : T \in M\}$.

7. Supprimer de M tout intervalle T tel que $LB(T) > UB_k - \varepsilon$.

8. Fixer $k \leftarrow k + 1$.

9. Fin tant que

10. STOP : s^k est une solution ε -optimale de (P).

Théorème 2.3.5 (Convergence de l'algorithme)

(i) Pour un $\varepsilon > 0$ donné, l'algorithme se termine après un nombre fini d'itérations et fournit une solution ε -optimale au problème (P).

(ii) Avec $\varepsilon = 0$ soit l'algorithme est fini, soit il génère une suite bornée $\{s_k\}$ dont tout point d'accumulation est une solution globale optimale de (P), et

$$UB_k \searrow \alpha, \quad LB_k \nearrow \alpha.$$

Démonstration 2.3.5 Pour faire la preuve on a besoin de théorème suivant sur la fonction borne inférieure affine :

Théorème 2.3.6 On a : $|L_h f(s) - f(s)| \leq \frac{1}{8} K h^2, \forall s \in S$.

Pour la preuve de ce théorème : (voir [25])

Remarque 2.3.6 On remarque d'après ce théorème qu'avec une discrétisation uniforme comme ci-dessus nous avons besoin de $m = (b-a)\sqrt{\frac{K}{8\varepsilon}} + 1$ évaluations de fonctions à m points de discrétisation ($[x]$ désigne la partie entière de x) pour déterminer une fonction borne inférieure linéaire ε par morceaux de f .

(i) D'après la (**Remarque 2.3.3**), on voit que le nombre maximal k d'itérations auquel $UB_k - LB_k \leq \varepsilon$ est $m = (b-a)\sqrt{\frac{k}{8\varepsilon}} + 1$, ainsi la partie (i) est vérifiée.

(ii) Supposons maintenant que l'algorithme est infini, alors il doit générer une suite infinie $\{T_k\}$ d'intervalles dont la longueur h diminue jusqu'à zéro, puis la suite entière $\{T_k\}$ se réduit à un singleton.

D'abord, à partir de la description de l'algorithme, on voit qu'à chaque itération k ,

$$LB(T_i^k) \geq LB(T^k).$$

En effet, si $s_{k,i}^* \notin]a_k^*, b_k^*[$, alors, $LB(T_i^k) = UB(T_i^k) = \min\{f(a_k^i), f(b_k^i)\} \geq LB(T^k) := \min\{f(s) : s \in T^k\}$, et donc T^k est supprimé de l'ensemble M . (voir **Remarque 2.3.2**).

Si $s_{k,i}^* \in]a_k^i, b_k^i[$ ($i = 1, 2$), alors de (2); il résulte que

$$q_k(s) - q_{k,i}(s) = B^i s + C^i$$

avec B^i et C^i sont des constantes données par

$$B^i := \frac{f(b_k) - f(a_k)}{b_k - a_k} - \frac{K}{2}(b_k + a_k) - \frac{f(b_k^i) - f(a_k^i)}{b_k^i - a_k^i} + \frac{K}{2}(b_k^i + a_k^i),$$

$$C^i := \frac{K}{2}a_k b_k + \frac{f(a_k)b_k - f(b_k)a_k}{b_k - a_k} - \frac{K}{2}a_k^i b_k^i + \frac{f(a_k^i)b_k^i - f(b_k^i)a_k^i}{b_k^i - a_k^i}.$$

Ainsi

$$\begin{aligned} \max \{q_k(s) - q_{k,i}(s) : s \in [a_k^i, b_k^i]\} &= \max \{q_k(a_k^i) - q_{k,i}(a_k^i), q_k(b_k^i) - q_{k,i}(b_k^i)\} \\ &= 0 \end{aligned}$$

car

$$\begin{aligned} q_k(a_k^1) &= q_{k,1}(a_k^1), q_k(b_k^2) = q_{k,2}(b_k^2) = q_{k,2}(b_k^2), q_{k,1}(b_k^1) \\ &= f(b_k^1) < q_k(b_k^1), q_{k,2}(a_k^2) = f(a_k^2) < q_k(a_k^2). \end{aligned}$$

Cela signifie que $q_k(s) - q_{k,i}(s) \leq 0$, or $q_k(s) \leq q_{k,i}(s)$ pour tout $s \in [a_k^i, b_k^i]$. En conséquence, $LB(T_i^k) \geq LB(T^k)$ et la suite $\{LB_k\}$ est donc non décroissante et bornée par le haut par α . En combinant avec le fait que la suite UB_k est non croissante et bornée par le bas par α , et que toute suite $\{T_k\}$ se réduit à un singleton, on obtient

$$UB_k - LB_k \searrow 0, \quad UB_k \searrow \alpha, \quad LB_k \nearrow \alpha.$$

De plus, puisque $s^k \in T^0$ et $UB_k = f(s^k)$, tout point de Cluster de la séquence $\{s^k\}$ appartient à T^0 et a la valeur de la fonction α , c'est-à-dire résout le problème (P) . Le théorème est prouvé.

► Cas des problèmes avec une contrainte supplémentaire non convexe (PC) :

Dans cette partie, nous appliquons la méthode de BB sur les problèmes avec une contrainte supplémentaire non convexe (PC) , dont l'algorithme est basé sur celui des problèmes (P) , de plus, le domaine réalisable est noté par D_{pc} .

Ici, on considère que la contrainte non convexe est essentielle i.e. : $\exists s$ tel que $g(s) > 0$, car sinon le problème (P) et (PC) auront le même domaine réalisable, donc, ils seront identiques.

On calcule la borne inférieure de la fonction objectif, en relaxant à la fois l'ensemble faisable et la fonction objectif. On obtient donc un problème de type (1).

Plus précisément, sur chaque sous intervalle $[a_k, b_k]$, on considère le pro-

blème suivant :

$$(PC_h) \begin{cases} \beta_h := \min q_h(s) \\ L_h g(s) - \frac{1}{2} K_g(s - a_k)(b_k - s) \leq 0 \\ a_k \leq s \leq b_k \end{cases}$$

où l'ensemble faisable est noté par D_{pc}^h , avec $h_0 = b - a$,

on a $D_{pc}^{h_0} \supset D_{pc}$, $\forall k$ et h , $D_{pc}^h \supset D_{pc}^h := \{s \in [a_k, b_k] : g(s) \leq 0\}$. De plus :

$$\lim_{h \rightarrow 0} q_h^g(s) = g(s), \quad \lim_{h \rightarrow 0} q_h(s) = f(s)$$

qui assure la convergence de notre algorithme,

où, les extrémités a'_k et b'_k de l'intervalle $D_{pc}^h := \{s \in [a_k, b_k] : q_h^g(s) \leq 0\}$, peut être calculé facilement, comme suit :

$$a'_k := \max\{a_k, s_1^g\}, \quad b'_k := \{b_k, s_2^g\}$$

$$\text{avec } s_1^g < s_2^g \text{ solution de l'équation } q_h^g(s) = 0. \quad (4) \text{ On}$$

remarque que D_{pc}^k est une réunion d'intervalles, tandis que, D_{pc}^h est exactement un seul intervalle (s'il n'est pas vide), puisque

$$q_h^g(s) := L_h g(s) - \frac{1}{2} K_g(s - a_k)(b - s_k)$$

est une fonction convexe. Le vide de $D_{pc}^h \cap D_{pc}$ est peut être détecté en observant que si : le minimum de $q_h^g(s)$ sur $[a_k, b_k]$ noté $s_{kg}^* \notin [a_k, b_k]$, alors

$$\min \{g(s) : s \in [a_k, b_k]\} = \min \{q_h^g(s) : s \in [a_k, b_k]\} = \min \{g(a_k), g(b_k)\}.$$

Donc :

$$\text{si } s_{kg}^* \notin [a_k, b_k] \text{ et } \min \{g(a_k), g(b_k)\} > 0, \text{ donc } D_{pc}^h \cap D_{pc} = \emptyset. \quad (5)$$

L'algorithme de Branch and Bound pour les problèmes avec une contrainte supplémentaire non convexe, est noté par BB2, et diffère principalement de l'algorithme précédent (algorithme des problèmes simples avec un contrainte d'intervalle), de la procédure de branchement.

- Procédure d'évaluation:

La procédure d'évaluation d'un problème (PC) est la même avec le

problème (P) , c'est-à-dire qu'on fait toujours la résolution d'un problème quadratique convexe de la forme :

$$\min \{q_h(s) : a'_k \leq s \leq b'_k\}.$$

Par ailleurs, la borne supérieure, peut être améliorée à l'itération k de l'algorithme BB, à condition que l'ensemble D_{pc} soit non vide. De plus lorsque le point réalisable de (PC) est détecté en vérifiant la condition $g(s) \leq 0$, avec $s \in \{s_{kf}^*, s_{kg}^*, s_1^g, s_2^g\}$. Puisque $\lim_{h \rightarrow 0} q_h^g(s) = g(s)$.

- Procédure de branchement :

La procédure de branchement d'un problème (PC) est différente de celle des problèmes (P) :

- Subdiviser l'intervalle $T^k := [a_k, b_k]$ en $D_{pc}^{h_k^1}$ et $D_{pc}^{h_k^2}$ par s_{kf}^* comme suit :

$$D_{pc}^{h_k^1} := \{s \in [a_k, s_{kf}^*] : q_{h_k^1}^g(s) \leq 0\}, D_{pc}^{h_k^2} := \{s \in [s_{kf}^*, b_k] : q_{h_k^2}^g(s) \leq 0\}. \quad (6)$$

- Écarter le sous intervalle $D_{pc}^{h_k^i}$ ($i = 1, 2$), si $D_{pc}^{h_k^i} \cap D_{pc} = \emptyset$, utilisant (5).

Sinon, déterminer les extrémités par (4).

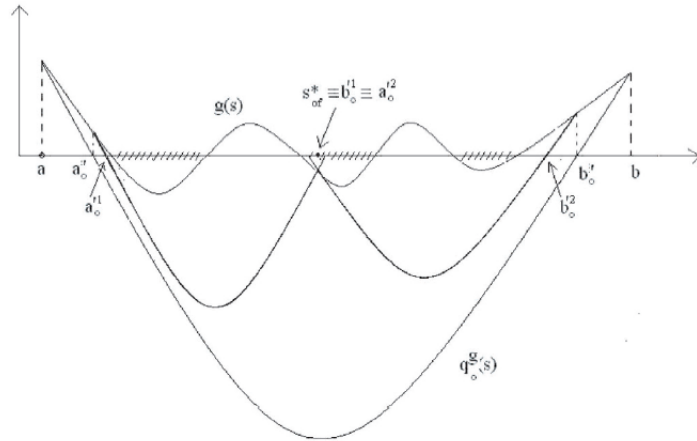


FIGURE 2.5 – L'ensemble faisable de problème (PC) après la 1^{ère} itération $[a_0^1, b_0^2]$

- Algorithme de la méthode Branch and Bound BB2 :

L'algorithme BB2 est un algorithme BB avec les modifications suivantes :

Ajouter à l'étape 1a) Soit K_g une constante vérifiant $K \geq |g''(s)|$, $\forall s \in [a, b]$. Modifier la partie concernant T^0 en étape 1b) : On pose $T^0 := \{s \in [a, b] : g(s) \leq 0\}$.

Chercher le vide de T^0 en utilisant (5). Si $T^0 = \emptyset$ alors STOP, le problème est irréalisable, sinon déterminer les extrémités a_0, b_0 de T^0 par (4).

Étape 1c) : Si $g(s_0^*) \leq 0$, alors on pose $UB_0 := \min\{f(a_0), f(b_0), f(s_0^*)\}$, sinon on pose $UB_0 = \min\{f(a_0), f(b_0)\}$.

Étape 4: subdiviser l'intervalle $T^k := [a_k, b_k]$ en $D_{pc}^{h_k^1}$ et $D_{pc}^{h_k^2}$ par s_{kf}^* suivant (6). Chercher le vide de $D_{pc}^{h_k^1}$ (resp. $D_{pc}^{h_k^2}$) en utilisant (5). Si $D_{pc}^{h_k^1} \neq \emptyset$, (resp. $D_{pc}^{h_k^2} \neq \emptyset$), alors déterminer ses extrémités a_k^1, b_k^1 (resp. a_k^2, b_k^2) suivant (4).

Étape 5.4 Mettre à jour $UB_k = \min\{UB_k, f(a_k^i), f(b_k^i), f(s_{k,i}^*)\}$. Pour $s \in \{s_{kf}^*, s_{kg}^*, s_1^g, s_2^g\}$, si $g(s) \leq 0$, alors faire la mise à jour de $UB_k = \min\{UB_k, f(s)\}$.

Convergence de l'algorithme BB2

La convergence de l'algorithme est assurée par :

$$\lim_{h \rightarrow 0} q_h^g(s) = g(s), \lim_{h \rightarrow 0} q_h(s) = f(s)$$

.

Remarque 2.3.7 Une preuve de convergence de cet algorithme à travers des simulation numériques est donnée en [21]

Remarque 2.3.8 L'exécution de l'algorithme sera explicité en **Chapitre 03**.

2.3.4 La méthode de Branch and Bound avec une nouvelle fonction borne inférieure pour les problèmes d'optimisation globale univariés [21], [25], [37], [38]

Les auteurs dans leur article [37], proposent l'algorithme de Branch and Bound avec une nouvelle fonction borne inférieure, qui n'est qu'une combinaison de deux autres fonctions borne inférieure quadratique [21], et classique de la méthode α BB [38].

Considérons le problème suivant :

$$(P) \begin{cases} \min b(s) \\ s \in [s^0, s^1] \subset \mathbb{R} \end{cases}$$

où b est non convexe et de classe C^2 , fonction continue sur l'intervalle fermé réel $[s^0, s^1]$.

Rappelons les différentes fonctions borne inférieure :

- La fonction quadratique borne inférieure :

Soit $K \in \mathbb{R}_+$ tel que $K \geq |b''(s)|, \forall s \in [s^0, s^1]$. La fonction quadratique borne inférieure sur l'intervalle $[s^0, s^1]$ est donnée par :

$$LB_q(s) = b(s^0) \frac{s^1 - s}{s^1 - s^0} + b(s^1) \frac{s - s^0}{s^1 - s^0} - \frac{K}{2} (s - s^0)(s^1 - s).$$

Elle vérifie les propriétés suivantes :

- 1- Elle est convexe et quadratique (c'est à dire $LB_q''(s) = K \geq 0$).
- 2- Elle coïncide avec la fonction b aux extrémités de l'intervalle $[s^0, s^1]$ (par construction de $LB_q(s)$).
- 3- Elle est une borne inférieure de $b(s)$ ($(b(s) - LB_q(s))'' = (b''(s) - K) \leq 0, \forall s \in [s^0, s^1] \Rightarrow b(s) - LB_q(s)$ est concave sur $[s^0, s^1]$, elle s'annule aux extrémités de l'intervalle $[s^0, s^1]$, alors $(b(s) - LB_q(s)) \geq 0, \forall s \in [s^0, s^1]$).

Pour plus de détails (voir [21]).

- La fonction borne inférieure dans la méthode α BB :

Soit $K_\alpha \in \mathbb{R}_+$ tel que $K_\alpha \geq \max\{0, -b''(s)\}, \forall s \in [s^0, s^1]$. La fonction borne inférieure dans la méthode α BB sur l'intervalle $[s^0, s^1]$ est donnée par :

$$LB_\alpha(s) = b(s) - \frac{K_\alpha}{2} (s - s^0)(s^1 - s).$$

Elle vérifie les propriétés suivantes :

- 1- Elle est convexe ($LB_\alpha''(s) = b''(s) + K_\alpha \geq 0$ puisque $K_\alpha \geq -b''(s), \forall s \in [s^0, s^1]$).
- 2- Elle coïncide avec la fonction b aux extrémités de l'intervalle $[s^0, s^1]$ (par construction de $LB_\alpha(s)$).

3- Elle est une borne inférieure de $b(s)$

$$(b(s) - LB_\alpha(s) = \frac{K_\alpha}{2}(s - s^0)(s^1 - s) \geq 0, \forall s \in [s^0, s^1]).$$

Voir plus de détails en [38].

• La nouvelle fonction borne inférieure :

Soit $K_q \in \mathbb{R}_+$ tel que $K_q \geq \max\{0, b''(s)\}$, $\forall s \in [s^0, s^1]$, et soit $L_h b(s)$ l'interpolation linéaire de b sur cet intervalle, donnée par

$$L_h b(s) = b(s^0) \frac{s^1 - s}{s^1 - s^0} + b(s^1) \frac{s - s^0}{s^1 - s^0} \text{ (voir [25])}.$$

Elle est donnée par

$$LB(s) = \frac{K_q b(s) + K_\alpha L_h b(s)}{K_\alpha + K_q} - \frac{K_\alpha K_q}{2(K_\alpha + K_q)}(s - s^0)(s^1 - s).$$

sur l'intervalle $[s^0, s^1]$.

La nouvelle fonction borne inférieure est plus serrée que celles données au dessus (fonction quadratique borne inférieure et fonction borne inférieure classique de la méthode α BB), c'est-à-dire qu'elle permet d'écarter plus de grandes régions qui ne contiennent pas le minimum global.

Proposition 2.3.1 *i) $LB(s)$ coïncide avec $b(s)$ aux extrémités de $[s^0, s^1]$.
ii) $LB(s)$ est convexe sur l'intervalle $[s^0, s^1]$.*

Démonstration 2.3.6 *i) C'est évident, par construction de $LB(s)$.*

ii) $\forall s \in [s^0, s^1]$, on a

$$LB''(s) = \frac{K_q b''(s)}{K_\alpha + K_q} + \frac{K_\alpha K_q}{K_\alpha + K_q} = \frac{K_q(K_\alpha + b''(s))}{K_\alpha + K_q} \geq 0.$$

En effet K_α, K_q sont des nombres non négatifs et $\forall s \in [s^0, s^1]$, on a $(K_\alpha + b''(s)) \geq 0$. Alors LB est convexe.

Dans le théorème qui suit, on montre que $LB(s)$ est une borne inférieure de $b(s)$.

Théorème 2.3.7 $LB(s) \leq b(s), \forall s \in [s^0, s^1]$.

Démonstration 2.3.7 En calculant la seconde dérivée de $(LB(s) - b(s))$ on a

$$(LB(s) - b(s))'' = \frac{K_q(K_\alpha + b''(s))}{K_\alpha + K_q} - b''(s) = \frac{K_\alpha(K_q - b''(s))}{K_\alpha + K_q} \geq 0.$$

Puisque K_α et K_q sont des nombres positifs et $K_q - b''(s) \geq 0$ sur $[s^0, s^1]$, donc $(LB(s) - b(s))$ est convexe. Elle s'annule également aux extrémités de cet intervalles. Alors $(LB(s) - b(s))$ est négatif ce qui implique que $LB(s) \leq b(s), \forall s \in [s^0, s^1]$.

Les deux théorèmes qui suivent montrent que la nouvelle fonction borne inférieure est plus serrée que celles développées en [21] et [38].

Théorème 2.3.8 $LB(s) \geq LB_\alpha(s), \forall s \in [s^0, s^1]$.

Démonstration 2.3.8 $\forall s \in [s^0, s^1]$, on a

$$(LB(s) - LB_\alpha(s))'' = \frac{K_\alpha(K_q - b''(s))}{K_\alpha + K_q} - K_\alpha = -\frac{K_\alpha(K_\alpha + b''(s))}{K_\alpha + K_q} \leq 0,$$

puisque K_α et K_q sont positifs et $(K_\alpha + b''(s))$ est non négatif sur l'intervalle $[s^0, s^1]$, alors $(LB(s) - LB_\alpha(s))$ est concave. Elle s'annule aux extrémités de cet intervalle, alors $LB(s) - LB_\alpha(s) \geq 0$ ce qui implique que $LB(s)$ est plus serrée que $LB_\alpha(s)$ sur l'intervalle $[s^0, s^1]$.

Théorème 2.3.9 $LB(s) \geq LB_q(s), \forall s \in [s^0, s^1]$.

Démonstration 2.3.9 En calculons la dérivée seconde de $(LB(s) - LB_q(s))$, on a

$$(LB(s) - LB_q(s))'' = \frac{K_q b''(s)}{K_\alpha + K_q} + \frac{K_\alpha K_q}{K_\alpha + K_q} - K.$$

On a la relation entre K , K_α et K_q qui est $K = \max\{K_\alpha, K_q\}$.

1^{er} cas : $K = K_\alpha \geq K_q$

$$\begin{aligned}
 (LB(s) - LB_q(s))'' &= \frac{K_q b''(s)}{K_\alpha + K_q} + \frac{K_\alpha K_q}{K_\alpha + K_q} - K_\alpha = \frac{K_q b''(s) - K_\alpha^2}{K_\alpha + K_q} \\
 &\leq \frac{K_q b''(s) - K_q^2}{K_\alpha + K_q} = (b''(s) - K_q) \frac{K_q}{K_\alpha + K_q}.
 \end{aligned}$$

Puisque K_q, K_α sont des nombres positifs et $(b''(s) - K_q) \leq 0, \forall s \in [s^0, s^1]$, alors $(LB(s) - LB_q(s))$ est concave sur $[s^0, s^1]$. Elle s'annule également sur les extrémités de cet intervalle, alors $LB(s) - LB_q(s) \geq 0, \forall s \in [s^0, s^1]$, ce qui implique que $LB(s)$ est plus serrée que $LB_q(s)$ sur l'intervalle $[s^0, s^1]$.

2^{ème} cas : $K = K_q \geq K_\alpha$

$$\begin{aligned}
 (LB(s) - LB_q(s))'' &= \frac{K_q b''(s)}{K_\alpha + K_q} + \frac{K_\alpha K_q}{K_\alpha + K_q} - K_q \\
 &= \frac{K_q b''(s) + K_\alpha K_q - K_q K_\alpha - K_q K_q}{K_\alpha + K_q} \\
 &= (b''(s) - K_q) \frac{K_q}{K_\alpha + K_q}.
 \end{aligned}$$

De la même manière que ci dessus, nous montrerons que $LB(s)$ est plus serrée que $LB_q(s)$.

Exemple 2.3.1 Soit $b(s) = \sin s, \forall s \in [0, 2\pi]$.

Pour cet exemple simple, on montre que $LB(s)$ est plus serrée que $LB_q(s)$ et $LB_\alpha(s)$.

On a $LB_\alpha(s) = \sin s - \frac{1}{2}s(2\pi - s)$, $LB_q(s) = -\frac{1}{2}s(2\pi - s)$ et $LB(s) = \frac{1}{2}LB_\alpha(s)$.

$\forall s \in [0, 2\pi]$, $LB''_\alpha(s) = -\sin s + 1$ est positive, alors elle est convexe sur cet intervalle, puisque LB_α s'annule en 0 et 2π , alors $LB_\alpha(s) \leq 0, \forall s \in [0, 2\pi]$.

$LB''_q(s) = 1$, alors $LB_q(s)$ est convexe, elle s'annule également en 0 et 2π , alors $LB_q(s) \leq 0, \forall s \in [0, 2\pi]$.

- On a $LB(s) = \frac{1}{2}LB_\alpha(s) \geq LB_\alpha(s)$. Puisque $LB_\alpha(s) \leq 0$ alors $LB(s)$ est plus serrée que $LB_\alpha(s)$ sur $[0, 2\pi]$.
- $(LB(s) - LB_q(s))'' = \frac{1}{2}(-\sin s + 1) - 1 = \frac{1}{2}(-\sin s - 1) \leq 0, \forall s \in [0, 2\pi]$, alors $(LB(s) - LB_q(s))$ est concave sur $[0, 2\pi]$, elle s'annule également en 0 et 2π , alors $(LB(s) - LB_q(s)) \geq 0, \forall s \in [0, 2\pi]$, alors $LB(s) \geq LB_q(s), \forall s \in [0, 2\pi] \Rightarrow LB(s)$ est plus serrée que $LB_q(s)$. (cf. Figure 2.7)

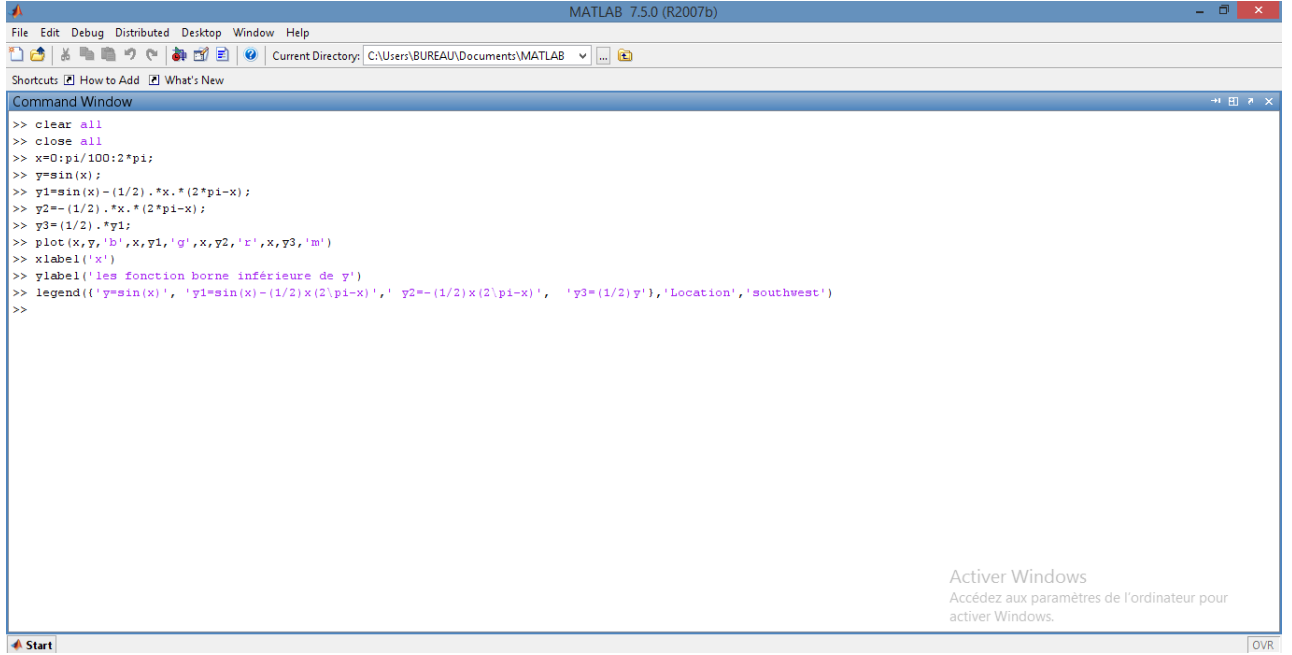


FIGURE 2.6 – Instruction Commande Window pour avoir la comparaison entre les différentes fonctions borne inférieure de la fonction $y = \sin(x)$.

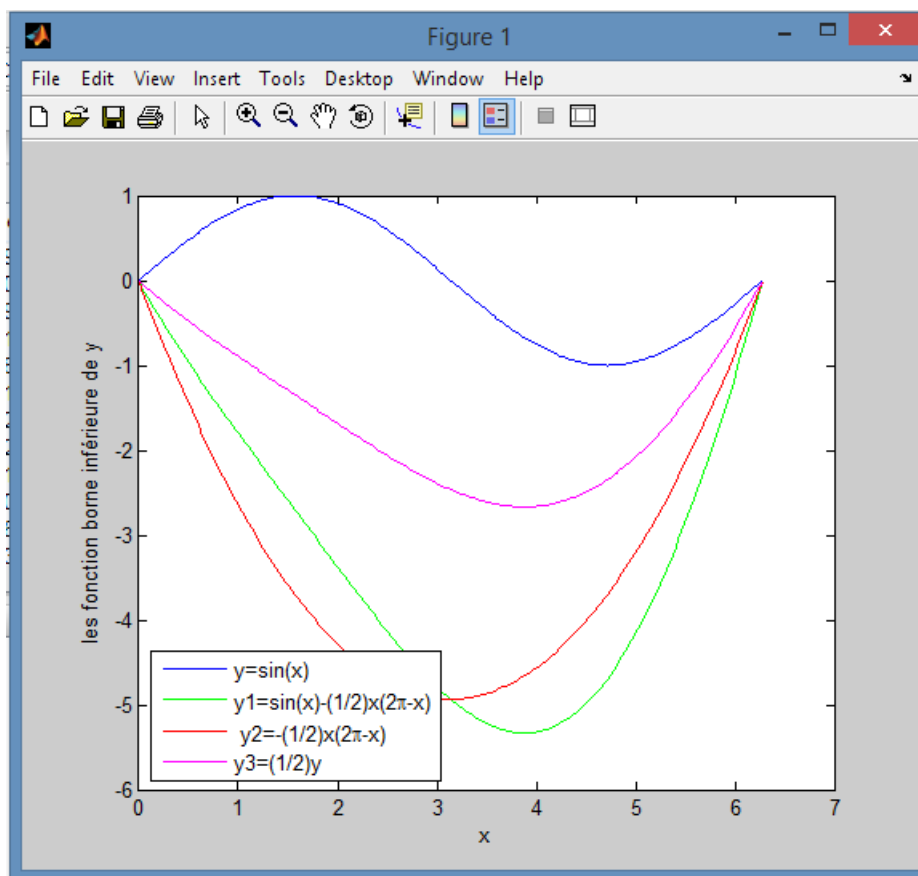


FIGURE 2.7 – Comparaison entre les différentes fonctions borne inférieure de la fonction $\sin(x)$.

Ici, $x = s$, y représente la fonction $\sin(s)$.

$$y_1 = LB_\alpha(s).$$

$$y_2 = LB_q(s).$$

$$y_3 = LB(s).$$

Remarque 2.3.9 Si le test de convexité/concavité est satisfait pour tous les sous intervalles considérer tout au long de l'algorithme, alors l'algorithme s'arrête, car chaque sous intervalle a un minimum exact, donc on trouve un minimum global, ou des minima globaux si la fonction objectif a plusieurs minima (multimodale).

Exemple 2.3.2 On considère à nouveau, le même exemple que ci dessus $b(s) = \sin s$, $s \in [0, \pi]$.

On a $K_\alpha = 1$, $K = 1$, et $K_q = 0$.

$K_q = 0$ donc b est concave sur $[0, \pi]$, alors son minimum global est atteint aux extrémités de cet intervalle (c'est-à-dire en 0 et π).

(cf. Figure 2.9)

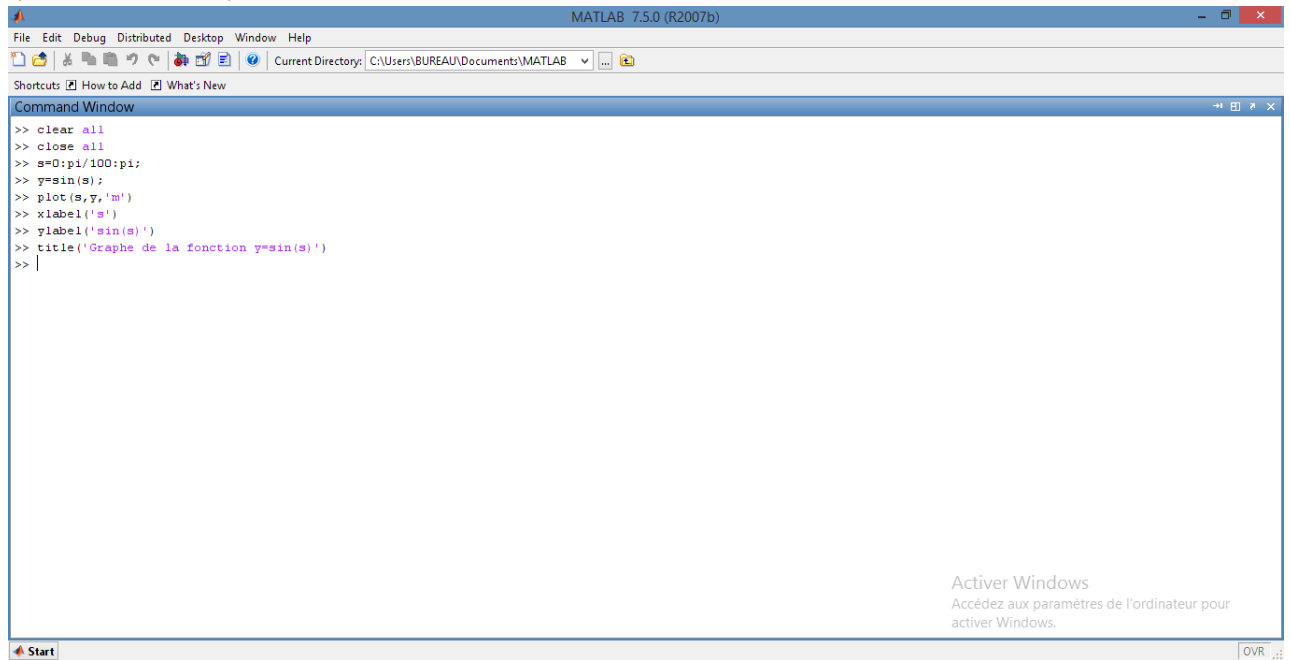
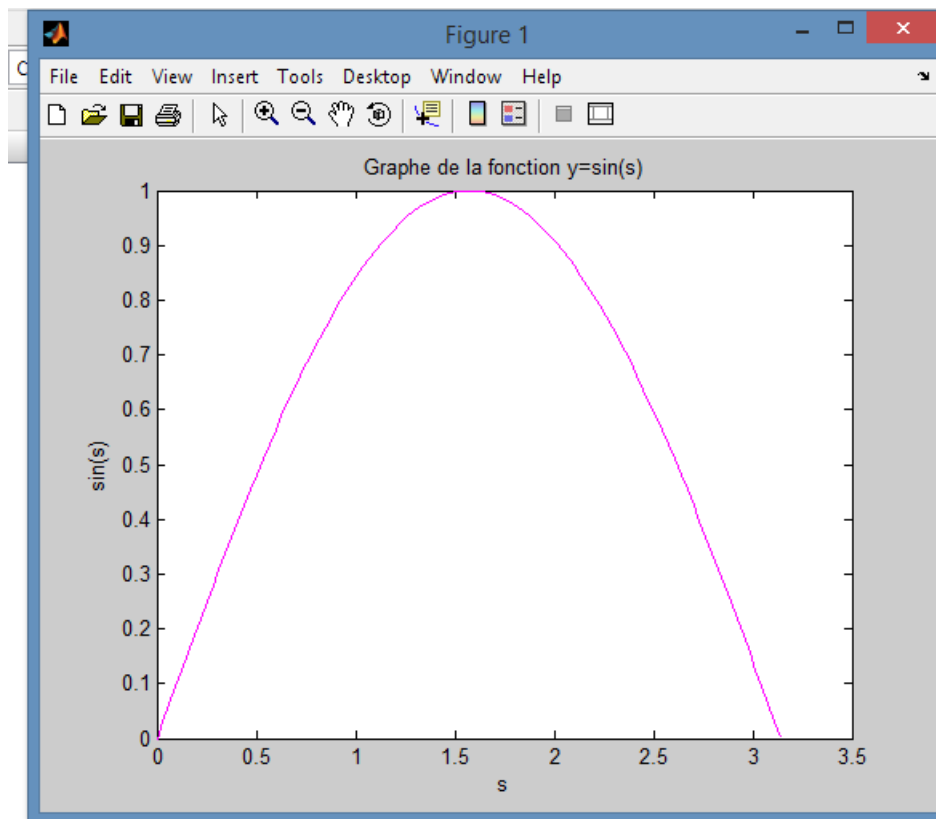


FIGURE 2.8 – Les instruction commande window pour avoir le graphe de la fonction sous Matlab

FIGURE 2.9 – La fonction $\sin(s)$ sous Matlab.

Remarque 2.3.10 Pour cet exemple, le test de convexité/concavité nous permet d'arrêter l'algorithme à la 1^{ère} itération et de trouver le minimum global, tandis que les deux méthode [21] et [38] ne peuvent

pas le faire à la première itération.

• **Algorithme de la méthode BB avec la nouvelle fonction borne inférieure :**

ii) Test de convexité/concavité

Si $K_\alpha^0 = 0$ stop b est convexe, tout recherche local donne une solution optimale.

Si $K_q^0 = 0$ stop b est concave, la solution optimale est atteinte aux extrémités de $[s^0, s^1]$.

iii) Soit $k := 0; T^0 = [s^0, s^1]; M := \{T^0\}$

iv) Résoudre le problème

$$\min\{LB_0(s) : s \in T^0\}$$

avec

$$LB_0(s) = \frac{K_q^0 b(s) + K_\alpha^0 (b(s^0) \frac{s^1 - s}{s^1 - s^0} + b(s^1) \frac{s - s^0}{s^1 - s^0} - \frac{K_\alpha^0 K_q^0}{2} (s - s^0)(s^1 - s))}{K_\alpha^0 + K_q^0}$$

pour obtenir une solution optimale s_0^* .

v) Poser $UB_0 := \min\{b(s^0), b(s^1), b(s_0^*)\}$

Poser $\bar{s}^0$ la meilleure solution courante c'est-à-dire $UB_0 = b(\bar{s}^0)$.

Poser $LB_0 := LB(T^0) = LB_0(s_0^*)$.

2. Pas d'itération Tant que $UB_k - LB_k > \varepsilon$, faire

2.1 Soit $T^k = [s^k, s^{k+1}] \in M$ un intervalle tel que $LB_k = LB(T^k)$ et soit s_k^* une solution de problème convexe T^k .

2.2 Couper T^k en deux intervalles $T_1^k = [s^k, s_k^*]$ et $T_2^k = [s_k^*, s^{k+1}]$

Poser $T_1^k := [s_1^k, s_1^{k+1}]$ et $T_2^k := [s_2^k, s_2^{k+1}]$.

2.3 Pour $i = 1, 2$ faire

a. Test de convexité/concavité

- Calculer $K_\alpha^{k_i}$ et $K_q^{k_i}$ sur T_i^k .

- Si $K_\alpha^{k_i} = 0$, b est convexe sur T_i^k , toute recherche locale donne une solution optimale $s_{k_i}^*$ sur T_i^k , puis mettre à jour $LB(T_i^k) = UB(T_i^k) = b(s_{k_i}^*)$ et aller à **c.**

- Si $K_q^{k_i} = 0$, b est concave sur T_i^k , puis mettre à jour $LB(T_i^k) = UB(T_i^k) = \min\{b(s_i^k), b(s_i^{k+1})\}$ et aller à **c.**

b. Posons $s_{k_i}^*$ la solution de problème convexe

$$\min\{LB^{k_i}(s) : s \in T_k^i\}$$

avec

$$LB^{k_i}(s) =$$

$$\frac{K_q^{k_i} b(s) + K_\alpha^{k_i} (b(s_i^k) \frac{s_i^{k+1} - s}{s_i^{k+1} - s_i^k} + b(s_i^{k+1}) \frac{s - s_i^k}{s_i^{k+1} - s_i^k}) - \frac{K_\alpha^{k_i} K_q^{k_i}}{2} (s - s_i^k)(s_i^{k+1} - s)}{K_\alpha^{k_i} + K_q^{k_i}}$$

et posons $LB(T_i^k) = LB^{k_i}(s_{k_i}^*)$.

c. Pour rentrer dans M les intervalles $T_i^k : M \leftarrow M \cup \{T_i^k : UB_k - LB(T_i^k) \geq \varepsilon\} \setminus \{T^k\}$.

d. Mettre à jour $UB_{k+1} := \min\{UB_k, b(s_i^k), b(s_i^{k+1}), b(s_{k_i}^*)\}$.

Poser $\bar{s}^{k+1}$ la meilleure solution c'est-à-dire $UB_{k+1} = b(\bar{s}^{k+1})$.

2.4 Mettre à jour $LB_{k+1} := \min\{LB(T) : T \in M\}$.

2.5 Supprimer de M tous les intervalles T tel que $LB(T) > UB_{k+1} - \varepsilon$.

2.6 Poser $k := k + 1$.

2.7 Fin tant que.

Sorties : $\bar{s}^k$ est une solution optimale globale de (P) .

Le théorème suivant montre la convergence de notre algorithme Branch and Bound.

Théorème 2.3.10 *La suite $\{\bar{s}^k\}$ générée par cet algorithme converge vers une solution optimale globale (P) .*

Démonstration 2.3.10 *Pour la démonstration de ce théorème référer à [37].*

Remarque 2.3.11 *Les différents calculs donné dans l'algorithme ci-dessus seront explicités et détaillés en **Chapitre 03**.*

2.3.5 Les méthodes hybrides [19], [39]

Définition 2.3.10 *La méthode hybride : se fait au moins entre deux méthodes distinctes (déterministes et non déterministes), afin d'avoir des algorithmes plus efficaces. Ses origines revient aux travaux de*

(Mühlenbein et al., 1988)(Grefenstette, 1987). Elle permet de déterminer le bassin d'attraction de minimum global, et la détermination de ce dernier. Elle consiste à exploiter les avantages respectifs de plusieurs méthodes, en combinant leurs algorithmes.

- Techniques d'hybridation : Il existe plusieurs techniques d'hybridation entre deux ou plusieurs méthodes exactes et approchées.

Dans le cadre de ce mémoire, on parlera d'une seule technique qui est :

1- La technique d'hybridation HIG : qui est un hybridation entre la méthode déterministe IBB (Interval Branch and Bound), et la méthode génétique.

- Principe de la technique HIG :

La technique d'hybridation (HIG) (Algorithme génétique d'intervalle hybride), se base sur le principe d'hybridation entre la méthode Branch and Bound par intervalles, qui prend des petites régions où la (ou les) solution(s) optimale(s) se trouve(nt), ainsi, initialiser une population de solutions possibles et les bornes initiales pour que le minimum global $\bar{f}$ soit obtenu.

L'algorithme génétique est appliqué de telle manière que, toute les informations ci-dessus soient exploitées. Il met à jour les bornes de chaque génération, et permet de définir le critère d'arrêt. Quand ce dernier est vérifié, l'algorithme converge vers le minimum global $\bar{f}$.

- Algorithme génétique d'intervalle hybride :

1. Appliquer un algorithme de subdivision ;
2. Initialiser la population ;
3. Evaluer chacun des individu de la population
4. Tant que (le critère d'arrêt non atteint) faire {
5. Mettre à jour des bornes
6. Choisir les individus pour la prochaine population ;

7. Appliquer les opérations génétiques pour produire de nouveaux individus ;
8. Evaluer les nouveaux individus ;}
9. Afficher $\overline{f^*}$ et $\overline{x^*}$.

Les bornes $\overline{F^*}$ et $\underline{F^*}$ qui sont obtenues par l'étape 1 de l'algorithme ci-dessus satisfaisant la relation suivantes :

$$\underline{F^*} \leq f^* \leq \overline{F^*}$$

2.4 Conclusion

Dans ce chapitre, nous avons défini l'optimisation globale, ensuite, donner la différence entre elle et l'optimisation locale, et enfin, citer les différents types des méthodes (déterministes, stochastiques et hybrides) ; en expliquant leur (définition, principe, algorithme, théorème de convergence et preuve de convergence).

L'optimisation globale a un intérêt primordial en pratique, car le minimum global est la seule solution qui reflète une réalité physique . Aussi, personne ne peut trouver une solution meilleure que le minima global. De plus, son importance pratique en industrie, ingénierie, biologie, médecine ...

Simulations numérique

3.1 Introduction

Dans ce chapitre on s'intéressera essentiellement à montrer l'efficacité de la méthode Branch and Bound avec une fonction borne inférieure quadratique [21], en explicitant les calculs sur trois itérations appliquées sur un exemple tiré de l'article [21]. De plus, on a exécuté trois itérations de l'algorithme présenté dans l'article [37], ensuite le comparer à celui donné dans [59]; prouver l'efficacité de l'algorithme Branch and Bound avec une nouvelle fonction borne inférieure, par l'application sur des exemples tirés de l'article [37].

3.2 Qu'est-ce-que Matlab ? [58]

Définition 3.2.1 *Matlab* : signifie *Matrix laboratory*. C'est un logiciel de calcul numérique, commercialisé par la société MathWorks¹; développé par Cleve Moler à la fin des années 70, afin de permettre à ses étudiant de travailler avec un langage de haut niveau et sans apprendre le Fortran ou le C.

C'est un langage de calculs scientifique, d'analyse des données, leur visualisation, développement des algorithmes. Son interface propose, d'une part, une fenêtre interactive type console pour l'exécution des commandes, et d'autre part, un environnement de développement intégré (IDE) pour la

1. <http://www.mathworks.com/>

programmation des applications.

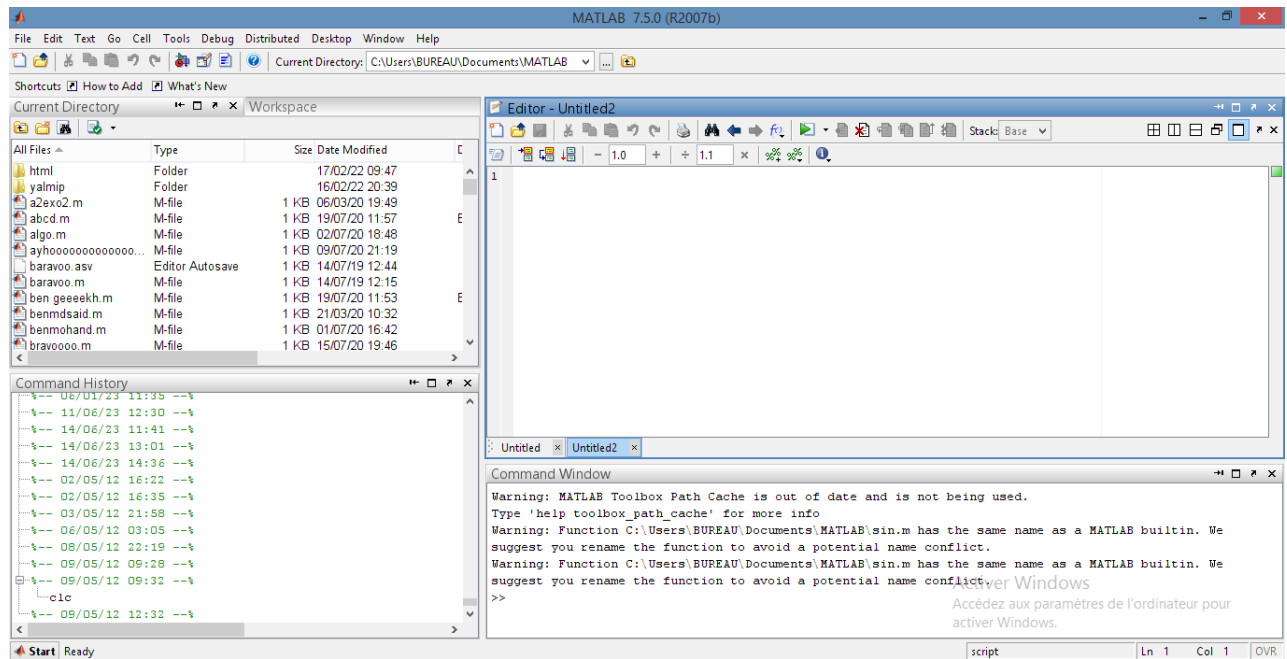


FIGURE 3.1 – Fenêtre principale de Matlab 2007.

3.3 Simulation numérique

3.3.1 Simulation numérique sur la méthode Branch and Bound avec une fonction borne inférieure quadratique [21]

Soit le problème suivant :

$$\min\{F_1(s) := \sin(s) + \sin\left(\frac{10s}{3}\right) + \ln(s) - 0.84s : 2.7 \leq s \leq 7.5\}.$$

1.Initialisation (Première itération $k = 0$) : Poser $K = 12.5$. Calculer

$$s_0^* := \frac{1}{2}(a + b) - \frac{1}{K(b - a)}(f(b) - f(a)) = 5.150736648187777 \simeq 5.150737.$$

Puisque $s_0^* \in [2.7, 7.5]$, nous traitons les procédures de Branch and Bound avec $T^0 = [2.7, 7.5]$, $LB_0 := q_0(s_0^*) = -37.973438103830809 \simeq -37.973438$ et $UB_0 := \min\{f_2(a), f_2(b), f_2(s_0^*)\} = f_2(s_0^*) = -4.586929097355390 \simeq -4.586929$.

On calcul la valeur de $UB_0 - LB_0 = -4.586929097355390 - (-37.973438103830809) = 33.386509006475421 \simeq 33.386509 > 0.002 = \varepsilon$ (*). Donc, on passe à **la deuxième itération** de l'algorithme :

2. Tant que $UB_k - LB_k > \varepsilon$ (D'après $(\star)$) faire

3. Soit $T^0 = [2.7, 7.5] \in M$ un intervalle, tel que $LB_0 = LB(T^0)$.

Soit K_k une constante vérifiant que $K_k \geq |f''(s)|$ pour tout $s \in [2.7, 7.5]$.

Prenons par exemple $K_k = 10$ (Puisque $10 > |f''(s)|$).

4. Couper T^0 en deux intervalles par la procédure de la w-subdivision :

$$T_1^0 = [a_0, s_0^*] := [a_0^1, b_0^1] = [2.7, 5.150737] \quad , \quad T_2^0 = [s_0^*, b_0] := [a_0^2, b_0^2] = [5.150737, 7.5]$$

5. Pour $i = 1, 2$ faire

$$5.1. \text{ Poser } s_{0,1}^* = \frac{1}{2}(a_0^1 + b_0^1) - \frac{f(b_0^1) - f(a_0^1)}{10(b_0^1 - a_0^1)} = 4.094773683862400 \\ \simeq 4.094774.$$

$$\text{Poser } s_{0,2}^* = \frac{1}{2}(a_0^2 + b_0^2) - \frac{f(b_0^2) - f(a_0^2)}{10(b_0^2 - a_0^2)} = 6.278226723569609 \\ \simeq 6.278227.$$

5.2. Puisque $s_{0,1}^* = 4.094774 \in]2.7, 5.150737[$, donc :

$$\text{On pose } LB(T_1^0) := \frac{10}{2}s_{0,1}^{*2} + \left[\frac{f(b_0^1) - f(a_0^1)}{b_0^1 - a_0^1} - \frac{12.5}{2}(b_0^1 + a_0^1) \right] s_{0,1}^* + \\ \frac{10}{2}a_0^1b_0^1 + \frac{f(a_0^1)b_0^1 - f(b_0^1)a_0^1}{b_0^1 - a_0^1} = -47.935762904906028 \simeq -47.935763.$$

Puisque on a $s_{0,2}^* = 6.278227 \in]5.150737, 7.5[$, donc :

$$\text{On pose } LB(T_2^0) := \frac{10}{2}s_{0,2}^{*2} + \left[\frac{f(b_0^2) - f(a_0^2)}{b_0^2 - a_0^2} - \frac{12.5}{2}(b_0^2 + a_0^2) \right] s_{0,2}^* + \\ \frac{10}{2}a_0^2b_0^2 + \frac{f(a_0^2)b_0^2 - f(b_0^2)a_0^2}{b_0^2 - a_0^2} = -110.2233395696455 \simeq -110.223396.$$

5.3. Pour rentrer dans M les intervalles $T_1^0 : M \rightarrow M \cup$

$$\{T_1^0 : UB^0 - LB(T_1^0) = 43.348833807550889 \simeq 43.348834 \geq \varepsilon = 0.002, \} \setminus \{T^0\}$$

Pour rentrer dans M les intervalles $T_2^0 : M \rightarrow M \cup$

$$\{T_2^0 : UB^0 - LB(T_2^0) = 105.6364104722901 \simeq 105.636410 \geq \varepsilon = 0.002, \} \setminus \{T^0\}$$

ainsi : $M = \{T^0\} = [2.7, 5.150737], [5.150737, 7.5]$.

$$\text{Mettre à jour } UB_0 = \min\{UB_0, f(a_0^1), f(a_0^2), f(b_0^1), f(b_0^2), f(s_{0,1}^*), f(s_{0,2}^*)\} \\ = -4.586929097355390 \simeq -4.586929$$

6. Mettre à jour $LB_0 = \min\{LB(T) : T \in M\} = \min\{LB(T_1^0), LB(T_2^0)\} = LB(T_2^0) = -110.2233395696455 \simeq -110.223396$.

7. On a $LB(T_1^0) = -47.935762904906028 \simeq -47.935763 \not\geq UB_0 - \varepsilon = -4.588929097355390 \simeq -4.588929$.

On a $LB(T_2^0) = -110.2233395696455 \simeq -110.223396 \not\geq UB_0 - \varepsilon = -4.588929097355390 \simeq -4.588929$.

Donc on n'en supprime aucun intervalle de T .

8. Poser $k \leftarrow 0 + 1 = 1$.

2. Puisque $UB_1 - LB_1 = 105.6364104722901 \simeq 105.636410 > \varepsilon = 0.002$, donc on passe à **la troisième itération** de l'algorithme alors, faire

3. Soit $T^1 = [a_1, b_1] \in M$ un intervalle tel que $LB_1 = LB(T^1) = LB(T_2^0) = -110.2233395696455 \simeq -110.223396$.

Soit K_k une constante vérifiant $K_k \geq |f''(s)|$, prenant par exemple $K_k = 8$ (puisque $8 \geq |f''(s)|$ pour tout $s \in [a_1, b_1] = [5.150737, 7.5]$).

4. Couper T^1 en deux intervalles par la procédure de la w-subdivision :

$$T_1^1 = [a_1, s_1^*] := [a_1, b_1^1] = [5.150737, 6.278227], \quad T_2^1 = [s_1^*, b_1] := [a_1^2, b_1^2] = [6.278227, 7.5].$$

5. Pour $i = 1, 2$ faire

$$5.1. \text{ Poser } s_{1,1}^* = \frac{1}{2}(a_1^1 + b_1^1) - \frac{f(b_1^1) - f(a_1^1)}{8(b_1^1 - a_1^1)} = 5.490586289657136 \\ \simeq 5.490586$$

$$\text{Poser } s_{1,2}^* = \frac{1}{2}(a_1^2 + b_1^2) - \frac{f(b_1^2) - f(a_1^2)}{8(b_1^2 - a_1^2)} = 6.982424286398203 \\ \simeq 6.982424.$$

5.2. Puisque $s_{1,1}^* = 5.490586 \in]5.150737, 6.278227[$, donc :

$$\text{On pose } LB(T_1^1) := \frac{8}{2}s_{1,1}^{*2} + \left[\frac{f(b_1^1) - f(a_1^1)}{b_1^1 - a_1^1} - \frac{10}{2}(b_1^1 + a_1^1) \right] s_{1,1}^* + \\ \frac{8}{2}a_1^1b_1^1 + \frac{f(a_1^1)b_1^1 - f(b_1^1)a_1^1}{b_1^1 - a_1^1} = -67.800629806547505 \simeq -67.800630$$

Puisque on a $s_{1,2}^* = 6.982424 \in]6.278227, 7.5[$, donc :

$$\text{On pose } LB(T_2^1) := \frac{8}{2}s_{1,2}^{*2} + \left[\frac{f(b_1^2) - f(a_1^2)}{b_1^2 - a_1^2} - \frac{10}{2}(b_1^2 + a_1^2) \right] s_{1,2}^* + \\ \frac{8}{2}a_1^2b_1^2 + \frac{f(a_1^2)b_1^2 - f(b_1^2)a_1^2}{b_1^2 - a_1^2} = -100.7564121281757 \simeq -100.756412$$

5.3. Pour rentrer dans M les intervalles $T_1^1 : M \rightarrow M \cup$

$$\{T_1^1 : UB^1 - LB(T_1^1) = 63.213700709192111 \simeq 63.213701 \geq \varepsilon = 0.002, \} \setminus \{T^1\}$$

Pour rentrer dans M les intervalles $T_2^1 : M \rightarrow M \cup$

$$\{T_2^1 : UB^1 - LB(T_2^1) = 96.169483030820302 \simeq 96.169483 \geq \varepsilon = 0.002, \} \setminus \{T^1\}$$

ainsi : $M = \{T^1\} = [5.150737, 6.278227], [6.278227, 7.5]$.

$$5.4. \text{ Mettre à jour } UB_1 = \min \{UB_1, f(a_1^1), f(a_1^2), f(b_1^1), f(b_1^2), f(s_{1,1}^*), f(s_{1,2}^*)\} = \\ -4.586929097355391 \simeq -4.586929.$$

6. Mettre à jour $LB_1 = \min \{LB(T) : T \in M\} = \min \{LB(T_1^1), LB(T_2^1)\} = -100.7564121281757 \simeq -100.756412$.

7. On a $LB(T_1^1) = -67.800629806547505 \simeq -67.800630 \not\succeq UB_1 = -4.586929097355391 \simeq -4.586929$.

On a $LB(T_2^1) = -100.7564121281757 \simeq -100.756412 \not\succeq UB_1 = -4.586929097355391 \simeq -4.586929$. Donc on n'en supprime aucun intervalle de T .

8. Poser $k \leftarrow 1 + 1 = 2$.

On refait de la même façon les différentes itérations de l'algorithme ; l'algorithme se termine après 7 itérations et converge vers la solution optimale du problème donné.

3.3.2 Simulation numérique sur la méthode Branch and Bound avec une nouvelle fonction borne inférieure [21], [37], [59]

Soit à résoudre le problème suivant :

$$(P) \begin{cases} \min b(s) \\ s \in [-2, 11] \end{cases}$$

avec $b(s) = \frac{1}{10} - s - \frac{79}{20}s^2 + \frac{71}{10}s^3 + \frac{3}{80}s^4 - \frac{52}{25}s^5 + \frac{1}{6}s^6$.

On explicite les calculs de l'algorithme [37].

On calcule les 3 premières itérations de l'algorithme de Branch and Bound avec une nouvelle fonction borne inférieure :

i)Initialisation : On calcule les différentes constantes de l'algorithme [37]:

On sait que :

$$1. K_\alpha^0 \geq \max\{0, -b''(s)\};$$

$$2. K^0 \geq |b''(s)|;$$

$$3. K_q^0 \geq \max\{0, b''(s)\}.$$

On a $b''(s) = \frac{1}{20}(-158 + 852s + 9s^2 - 832s^3 + 100s^4)$.

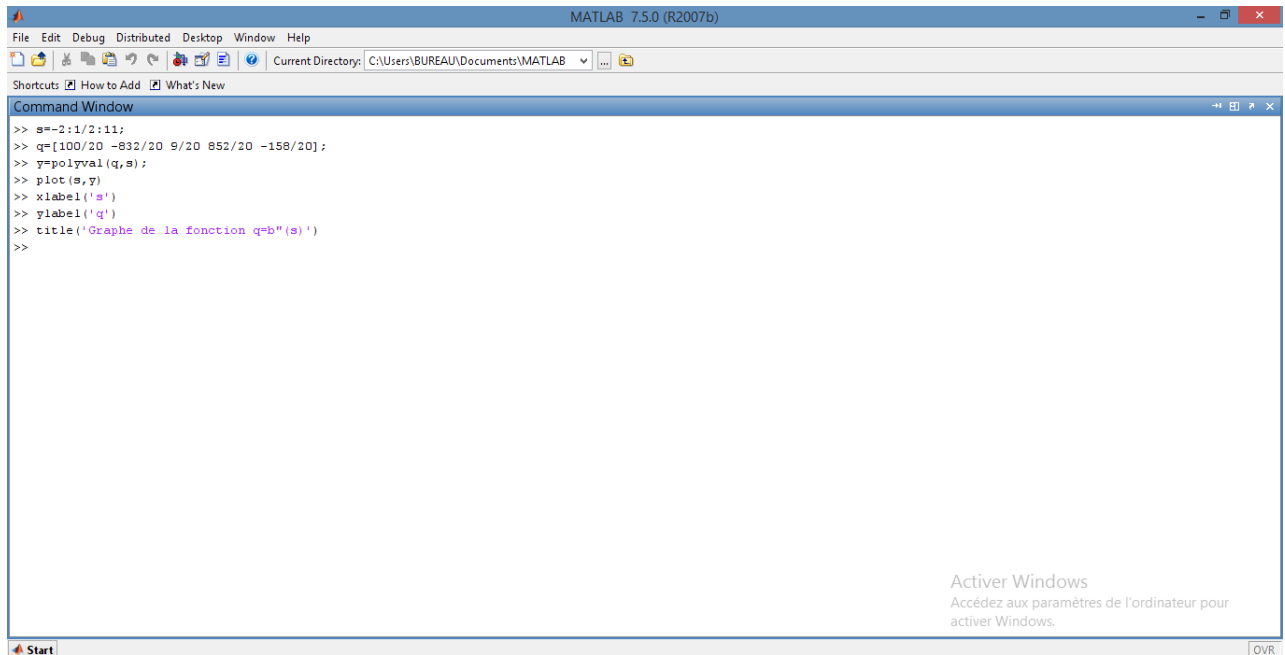


FIGURE 3.2 – Instruction pour tracer le graphe de $b''(s)$ sous Matlab (Command Window)

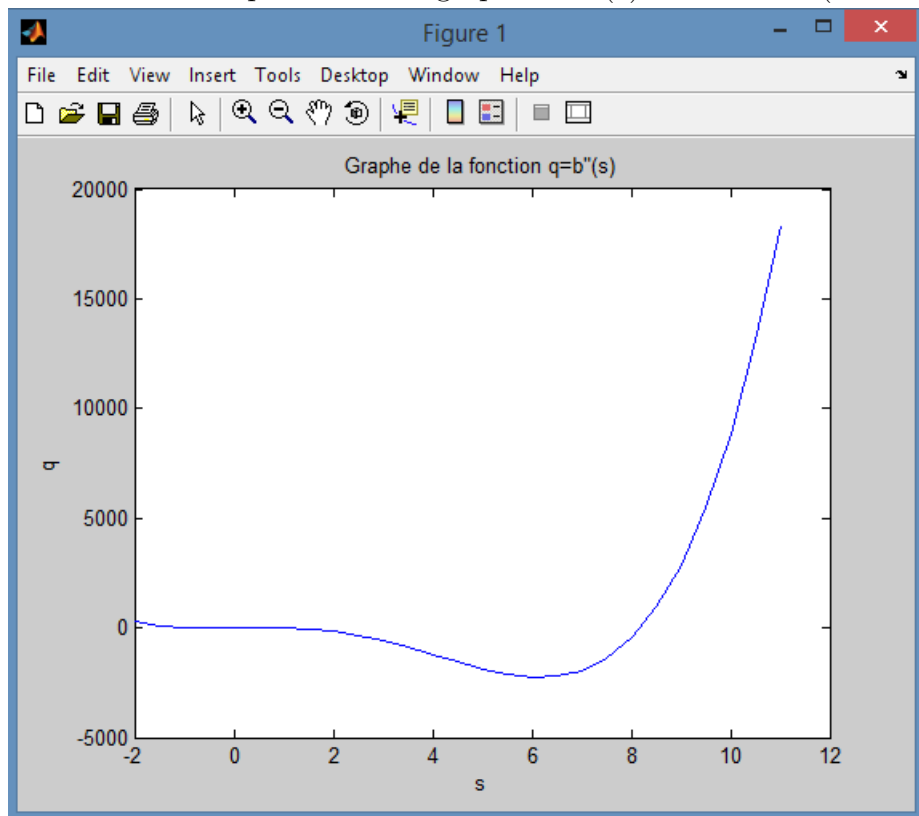


FIGURE 3.3 – La représentation graphique de $b''(s)$ sous Matlab.

Pour avoir les valeurs de K_α et K_q ; on doit comparer entre $b''(s)$ et 0.

Si $b''(s) \geq 0$ alors $\max := b''(s)$, sinon $\max := 0$.

Ceci peut se faire :

A. En étudiant le signe de $b''(s)$ (polynôme de degré (4)), ensuite si c'est

négalif le maximum est 0, sinon le maximum = maximum de $b''(s)$;

B· où bien en cherchant le maximum de $b''(s)$ (par l'utilisation de logiciel Matlab) et le comparer ensuite à 0.

On choisit la deuxième méthode (c'est à dire on calcule le maximum de $b''(s)$ et on le compare par la suite à zéro).

Le maximum de $b''(s)$ réalise $(b''(s))' = 0$.

On pose $g'(s) = (b''(s))'$.

On a $g'(s) = \frac{3}{10}(200s^2 - 832s + 3)$.

On résout l'équation $g'(s) = 0$.

$$g'(s) = 0 \Leftrightarrow \frac{1}{10}(426 + 9s - 1248s^2 + 200s^3) = 0$$

Les solutions de cette équation sont : $s_1 = \frac{2619}{424}$

$$s_2 = \frac{1168}{1885}$$

$$s_3 = \frac{-1029}{1849}$$

Pour savoir la nature de ces extremum, on doit calculer $(b''(s))'' = g''(s)$

$$g''(s) = \frac{3}{10}(200s^2 - 832s + 3)$$

On fais l'étude de signe de $g''(s)$:

$$g''(s) = 0 \Leftrightarrow \frac{3}{10}(200s^2 - 832s + 3) = 0 \Leftrightarrow (200s^2 - 832s + 3) = 0$$

On a $\Delta = 689824 > 0$, ainsi l'équation admetts deux solutions disjointes, qui sont :

$$s_1 = \frac{2764}{665},$$

et

$$s_2 = \frac{54}{14963}.$$

On trace le tableau de signe de $g''(s)$:

s	-2	$\frac{54}{14963}$	$\frac{2764}{665}$	11	
$g''(s)$	+	0	-	0	+

TABLE 3.1 – Tableau de signe de $g''(s)$.

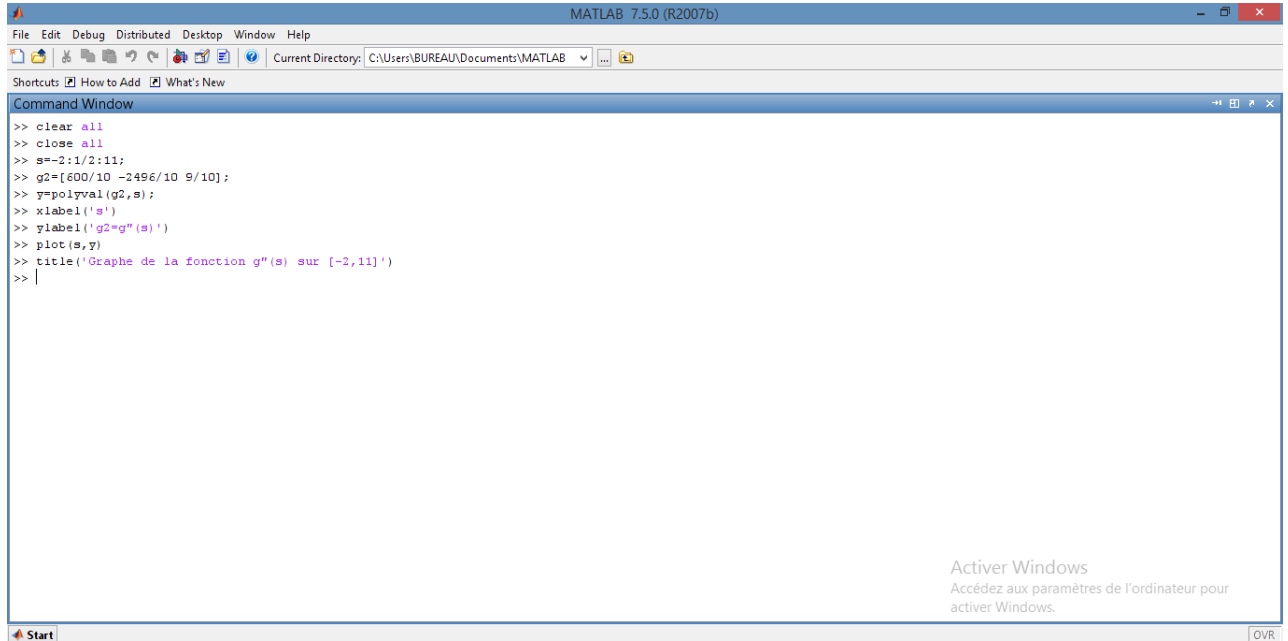


FIGURE 3.4 – Instruction pour tracer le graphe de $b''(s)$ sous Matlab (Command Window)

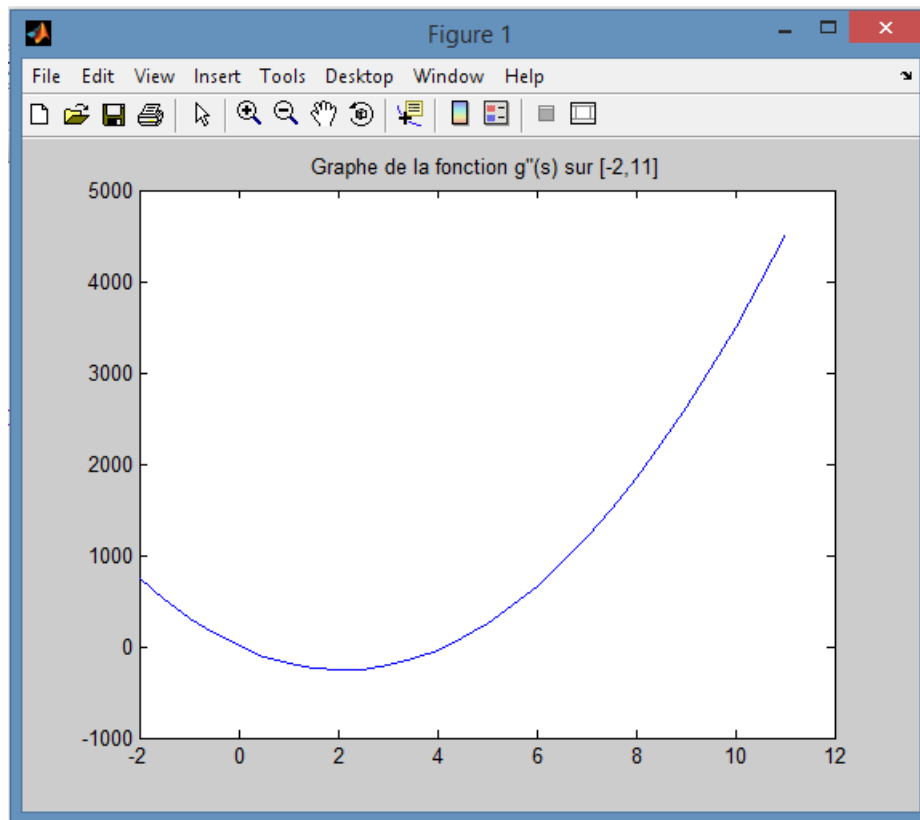


FIGURE 3.5 – La représentation graphique de $g''(s)$ sous Matlab.

On remarque que sur les intervalles $[-2, \frac{54}{14963}]$ et $[\frac{2764}{665}, 11]$, $g''(s) \geq 0$, ainsi

la fonction $b''(s)$ est convexe, donc elle admet deux minimums locaux sur ces deux intervalles $[-2, \frac{54}{14963}]$ et $[\frac{2764}{665}, 11]$.

On remarque que sur l'intervalle $[\frac{54}{14963}, \frac{2764}{665}]$ $g''(s) \leq 0$, ainsi la fonction $b''(s)$ est concave sur cet intervalle, donc elle admet un maximum local sur l'intervalle $[\frac{54}{14963}, \frac{2764}{665}]$ $g''(s) \leq 0$

Ainsi la fonction $b''(s)$ admet un maximum local sur l'intervalle $[\frac{54}{14963}, \frac{2764}{665}] \subset [-2, 11]$ qui est $s_2 = \frac{1168}{1885}$.

Première itération :

i)- **On calcul les différentes constantes de l'algorithme :** Sur l'intervalle $[\frac{54}{14963}, \frac{2764}{665}]$ on a : $s_2 = \frac{1168}{1885}$ et $b''(s_2) = -751/1273$ Ainsi :

$$K_\alpha^0 \geq \max\{0, -(\frac{-751}{1273})\} = \max\{0, \frac{751}{1273}\} = \frac{751}{1273}$$

$$K^0 \geq |-\frac{751}{1273}| = \frac{751}{1273}.$$

$$K_q^0 \geq \max\{0, \frac{-751}{1273}\} = 0.$$

ii)- **On applique le test de convexité concavité**

On a $K_q^0 \geq \max\{0, \frac{-751}{1273}\} = 0. \Rightarrow$ que b est convexe, toute recherche locale donne une solution optimale sur l'intervalle $[\frac{54}{14963}, \frac{2764}{665}] \subset [-2, 11]$ mais pas sur l'intervalle $[-2, 11]$ tout entier (voir le graphe de la fonction $b(s)$, ainsi **le test de convexité/concavité n'est pas vérifié sur l'intervalle $[-2, 11]$** .

Remarque 3.3.1 *Le test de convexité est le premier critère d'arrêt de l'algorithme proposé dans [37], s'il est vérifié l'algorithme s'arrête et fournit une solution optimale globale dès la première itération, ce qui n'est pas possible par l'application d'autres algorithmes de recherche de minimum(s) global(aux), comme l'algorithme proposé dans [21], et celui donné dans [59]. Également, c'est un critère utilisé pour accélérer la convergence de cet algorithme([37]).*

Deuxième itération :

Donc on continue l'application des différentes itérations de cet algorithme.

iii) **Poser** $k := 0$; $T^0 = [-2, 11]$; $M := \{T^0\}$

iv) **Résoudre le problème convexe**

$$\min \{LB_0(s) : s \in T^0\}$$

avec $LB_0(s) = \frac{K_q^0 b(s) + K_\alpha^0 (b(s^0) \frac{s^1-s}{s^1-s^0} + b(s^1) \frac{s-s^0}{s^1-s^0} - \frac{K_\alpha^0 K_q^0}{2} (s-s^0)(s^1-s))}{K_\alpha^0 + K_q^0}$ pour obtenir une solution optimale s_0^* .

On aura donc :

$$LB_0(s) = -\frac{232017}{59} - \frac{338796}{43}s$$

Ainsi, on résout le problème convexe suivant :

$$\begin{cases} \min -\frac{232017}{59} - \frac{338796}{43}s \\ s \in [-2, 11] \end{cases}$$

Le minimum est réalisé pour $s_0^* = 11$ de valeur optimale $f(s_0^*) = -\frac{209921873}{2537} = -\frac{330945}{4}$.

v) Poser $UB_0 := \min\{b(s^0), b(s^1), b(s_0^*)\} = b(s^0) = b(s_0^*) = b(11) = -\frac{241725}{8}$

Poser $\bar{s}^0$ est la meilleure solution courante c'est-à-dire $UB_0 = b(\bar{s}^0) = b(11)$.

Soit $LB_0 := LB(T^0) = LB_0(s_0^*) = -\frac{330945}{4}$.

Étapes des itérations : on a : $UB_0 - LB_0 = 52520.625 > \varepsilon = 0.002$.

Donc, on continue l'application des différentes instructions de l'algorithme.

2.1 Soit $T^0 := [s_1^0, s_1^1] = [-2, 11] \in M$ un intervalle tel que $LB_0 = LB(T^0) = LB(11) = -\frac{330945}{4}$ et soit $s_0^* = 11$ la meilleure solution de problème convexe en $T^0 = [-2, 11]$.

2.2 Puisque $s_0^* = 11$ donc on garde toujours le même intervalle qui est $[-2, 11]$.

Puisque, on a appliqué le test de convexité/concavité, on a résolu le problème convexe,..., on pose $k := 0 + 1 = 1$, et on passe directement à **troisième itération** de l'algorithme.

Troisième itération :

Puisque pour $k = 1$ on a toujours le même intervalle, donc on aura toujours les même instructions et les même résultats que la première et la deuxième itération ainsi la valeur $LB_1 = -\frac{330945}{4}$.

Calculons les valeurs de LB_k par l'algorithme de α BB :

On sait que $LB_\alpha(s) = b(s) - \frac{K_\alpha}{2}(s - s^0)(s^1 - s)$.

Ainsi $LB_\alpha(s_0^*) = LB_\alpha(11) = b(s_0^*) = b(11) = -241725/8$ pour $k = 0$, et la même valeur pour $k = 1$ (puisque on a la même solution optimale pour les deux itération.

Itérations	$LB(s_0^*) = LB(11)$	$LB_\alpha(s_0^*) = LB_\alpha(11)$
$k = 0$	$-\frac{330945}{4}$	$-\frac{241725}{8}$
$k = 1$	$-\frac{330945}{4}$	$-\frac{241725}{8}$

TABLE 3.2 – Tableau de comparaison entre la méthode de BB avec une nouvelle fonction borne inférieure et la méthode α BB.

Sur chaque itération, on remarque que $LB(s) < LB_\alpha(s)$, ainsi pour cet exemple la méthode α BB est meilleure que la méthode BB avec une nouvelle fonction borne inférieure, dans d'autre exemple ou le test de convexité concavité est vérifié (test qui accélère la convergence de l'algorithme proposé dans [37]), l'algorithme BB est meilleur que l'algorithme α BB.

Fct	NbF	GoF	NbO	GoO	GsO
					7.853981
1	16	-1	7	-1	14.137166
					20.420352
2	8	-1	3	-1	6.283185
3	1	0	1	0	0.999991
4	64	-17.58287	27	-17.58287	6.894525
5	1024	-0.020903	1	-0.020903	0.067812
6	8	-0.952897	13	-0.952896	2.839347
7	8	-6.262872	1	-6.262872	6.920063
8	16	0.077590	3	0.077589	0.902209
9	16	0.211315	1	0.211314	0.224882
10	16	-0.478362	5	-0.478361	0.724896
11	4	-5.815675	3	-5.815674	5.872866
12	8	-7.047444	5	-7.047444	5.134338
13	8	-4.60138	5	-4.601307	5.199786
14	4	-0.14110	3	-0.141100	0.408237
15	16	-0.870885	11	-0.870885	4.858047
16	16	-9.031249	13	-9.031249	5.791796
17	1	0.475689	1	0.475688	-0.787891
					3.141584
18	8	0	5	0	6.283180
19	64	-1	3	-1	-0.000044
20	1	1	1	1	0
21	16	1	3	1	0.000055
22	4	-0.918397	3	-0.918397	3.251079

FIGURE 3.6 – Tableau qui prouve l'efficacité de la méthode BB avec une nouvelle fonction borne inférieure par rapport à la méthode α BB.

Où **NbF**, **NbO** : est le nombre de sous intervalles générés de la méthode α BB et BB avec une nouvelle fonction borne inférieure respectivement.

GoF, **GoO** : est l'optimum global généré par la méthode α BB et la méthode BB avec une nouvelle fonction borne inférieure respectivement.

GsO : est la solution optimale globale générée par la méthode BB avec une

nouvelle fonction borne inférieure.

Remarque 3.3.2 *On remarque que le nombre de sous intervalles générés par la méthode BB avec une nouvelle fonction borne inférieure est plus petit que le nombre de sous intervalles générés par la méthode de α BB, (c'est-à-dire que cette première méthode converge vers la solution optimale en un nombre moins d'itérations).*

D'après le tableau, la solution globale générée, et la valeur optimale globale trouvée par l'algorithme proposé en [37] sont meilleure que celles générées de la méthode proposée en [59].

Ainsi, l'algorithme proposé en [37] est meilleur que celui donné en [59] et génère une meilleure solution optimale globale.

3.4 Conclusion

Dans ce chapitre, on a explicité les calculs de l'algorithme Branch and Bound avec une fonction borne inférieure quadratique [21], à travers certaines itérations (trois itérations).

De plus, on a montré comment exécuter l'algorithme de la méthode Branch and Bound avec une nouvelle fonction borne inférieure, en comparant la borne inférieure générée par cet algorithme, et celle générée par l'algorithme présenté dans l'article [59]. Les calculs sont appliqués sur des exemples présentés dans l'article [21].

Prouver l'efficacité de la méthode proposée dans [37], par rapport à celle donnée dans [59], à travers des simulations numérique sur un exemple tiré de l'article [37].

conclusion générale et perspectives

L'optimisation locale (convexe), globale (non convexe), jouent un rôle primordiale dans la résolution des problèmes de maximisation ou de minimisation d'actualité.

Nous nous sommes intéressés dans le premier chapitre de ce mémoire, à donner des notions de base et des généralités sur l'optimisation locale convexe ; qui consiste à chercher l'optimum(s) local(aux) d'un problème avec ou sans contraintes.

L'un des inconvénients majeurs des méthodes locales, est qu'elles sont piégées en un extremum local. Aussi, elles fournissent peu d'intérêt en pratique. D'où la nécessité de faire un passage à l'optimisation globale.

Dans le deuxième chapitre de ce travail, on s'est intéressé à définir l'optimisation globale. En outre, donner ses différentes approches ; déterministes, qui s'appliquent particulièrement aux problèmes avec contraintes ou avec une structure particulière. Les approches exactes, sont difficiles à appliquer sur les problèmes non linéaires, d'où la nécessité de passer aux méthodes stochastiques, qui sont des méthodes appliquées pour la résolution des problèmes avec ou sans contraintes. Pour accélérer la convergence de l'une des méthodes stochastiques (les algorithmes génétiques), on a hybridé ces derniers (algorithmes génétiques) avec une méthode déterministe qui est IBB (méthode de Branch and Bound par intervalles), pour avoir une méthode hybride dénommée par HIG (Algorithmes génétiques d'intervalle hybride).

Le troisième chapitre, est consacré pour la simulation numérique, en donnant un exemple pratique simulé sous le langage Matlab ; qui prouvent l'efficacité de la méthode Branch and Bound avec une fonction borne in-

férieure quadratique. Ainsi, on a explicité les calculs, l'exécution de l'algorithme, et prouver l'efficacité de la méthode BB avec une nouvelle fonction borne inférieure, en la comparant avec l'algorithme proposé dans l'article [59].

Comme perspectives, il est souhaitable d'élargir le domaine d'application des différentes méthodes d'optimisation globale aux cas multidimensionnels, et de développer en particulier un algorithme Branch and Bound avec une fonction borne inférieure quadratique, un algorithme Branch and Bound avec une nouvelle fonction borne inférieure, pour les problèmes à plusieurs variables.

Bibliographie

- [1] M. Minoux, Programmation mathématique théorie et algorithmes, 2^{ème} édition, éditions : TEC & DOC, (2008).
- [2] J. Baptiste H. Urruty, Optimisation et analyse convexe Exercices corrigés, EDP Sciences, (2009).
- [3] M. Ouanes , Cours d'optimisation non linéaire, Master 01, Recherche opérationnelle, (2021/2022).
- [4] M. Chebbah, Thèse Doctorat : Résolution et implémentation de problèmes en optimisation globale, Université de Tizi-Ouzou, (2020).
- [5] S. Boulimier, Thèse de Doctorat : Optimisation globale avec Local Solver, Université de Grenoble Alpes, (2020).
- [6] S. Belahcene, Mémoire de Magister : Résolution de problèmes de programmation semi-infinie et introduction à l'optimisation globale, université de Tizi-Ouzou, (2005).
- [7] M. Ouanes, Thèse de Doctorat d'Etat : Optimisation Semi-infinie et Optimisation Globale : Théorie et Algorithmes, université de Tizi-Ouzou, (2006).
- [8] M. Minoux, Programmation mathématique théorie et algorithmes, tome 2, Dunod, (1983).
- [9] M. Delfour, Introduction à l'optimisation et au Calcul Semi-différentiel : Cours et exercices corrigés, Dunod, (2012).
- [10] J.Wiley&Sons, Convex sets and their applications, New york, (1982), (Book dedicated to F.A. Valentine).

- [11] [http: //quentin.mrgt.fr/files/teaching/aca-2016/chap1.pdf](http://quentin.mrgt.fr/files/teaching/aca-2016/chap1.pdf)
- [12] Cours de Master MF-CSMI de l'université de Strasbourg : Convexité en optimisation, convexité forte. Téléchargeable sur le site: [https: //irma.math.unistra.fr/privat/documents/M1_Optim/Alpha_cvx.pdf](https://irma.math.unistra.fr/privat/documents/M1_Optim/Alpha_cvx.pdf).
- [13] T. Bayen, Cours d'optimisation numérique, Université de Montpellier 2, M1 Mathématique statistique et application -Optimisation Numérique, (2016-2017).
- [14] A.V. Levy and A. Montalov, The tunneling algorithm for the global minimization of functions, SIAM J. Sci. Statist. Comput., 6, No. 1, 15-29, (1985),.
- [15] Y. Yao, Dynamic tunneling algorithm for global optimization. Systems, Man and Cybernetic IEEE Transactions on, 19(5) : 1222-1230, (1989),.
- [16] C.G. Moles, P. Mendes and J.R. Banga, Parameter Estimation in Biochemical Pathways : A Comparison of Global Optimization Methods, Cold Spring Harbor Laboratory Press ISSN 1088-9051/03\$5.00 ; www.genome.org, 2467-2469, (2003).
- [17] DJ. Guettal, Thèse de Doctorat : Efficacité et fiabilité des méthodes utilisant l'approche 'Branch-and-Bound' pour l'optimisation globale non convexe, Université de Ferhat Abbas Settif 1, (2014).
- [18] A. Ketfi-cherif, Thèse de Doctorat : Sur certaines méthodes d'optimisation globale basées sur l'introduction de fonctions auxiliaires, Université de Ferhat Abbas Settif 1,(2016).
- [19] N. Boukhari, Thèse de Doctorat LMD : Optimisation des algorithmes évolutionnaires par hybridation pour la résolution des problèmes multi-objectifs, Université de Mostapha Stambouli, Mascara, (2021).
- [20] F. Harfouchi, Thèse de Doctorat : Contribution à l'optimisation par colonies d'abeilles artificielles : Application à la conception de systèmes intelligents, Université de Bougara-Boumerdes, (2019).

-
- [21] H.A. Le Thi and M. Ouanes, Convex quadratic underestimation and branch and bound for univariate global optimization with nonconvex constraint, *Rairo Operation Research* , EDP sciences, 285-302, (2006).
 - [22] H.A. Le Thi and T. Dinh, A Branch-and-Bound method via DC Optimization Algorithm and Ellipsoidal technique for Box Constrained Non convex quadratic Programming Problems. *J.Global optim*, 171-206, (1998).
 - [23] T.Q. Phong, H.A Le Thi and P.D. Tao, On the global solution of lineary constrained indefinite quadratic minimization problems by decomposition branch and bound method . *RAIRO : Oper.* 31-49, res.30 (1996).
 - [24] J.E. Falk et R.M. Soland, An algorithm for separable nonconvex programming problems. *Manage.sci.* **15**, 550-569, (1969).
 - [25] C. de Boor, A pratical Guide to Splines, *Applied Mathematical Sciences*. Springer-Verlag (1978).
 - [26] A.A. Zhigljavsky, *Theory of Global Random Search*, Kluwer Academic Publishers, Dordrecht, J.pinter, (1991)
 - [27] M. Rahal, Thèse de Doctorat : Extentsion de certaines méthodes de recouvrement en optimisation globale, Université de Ferhat Abbas de Setif, (2009).
 - [28] E. Mohamed Zeriab, Thèse en cotutelle : Contribution à l'optimisation globale : approche déterministe et stochastique et application, *Mathématiques générales [math.GM]* 58-65. INSA de Rouen; Université Mohammed V-Agdal. Français. NNT : 2009ISAM0010. tel-00560887, (2009).
 - [29] F. Debbat et F.T. Bendimerad., 3rd International Conference : Sciences of Electronic, Technologies of Informations and Telecommunication, Les algorithmes d'Optimisation Globale : Application Réseaux intelligent d'Antennes, 2, March,27-31,TUNISIA, (2005).
 - [30] H. Hanaâ, Thèse de Doctorat en Cotuelle : Hybridation d'algorithmes mathématiques en optimisation globale et leurs applications, Univer-

- sité de Mohammed V-Agdal, Rabat, École Mohammadia d'ingénieurs, Institut National des Sciences Appliquées de Rouen, (2013).
- [31] J.H. Holland, *Adaptation in Natural and Artificial Systems*, A Bradford Book, The MIT Press, Combridge, Massachusetts, London, England, (1975).
 - [32] D.E. Goldberg, *Genetic algorithms for search, optimization, and machine learning*. In : Addison-Wesley MA : (ed), Reading, United States Departement of Interior, Manuel Lujan, Jr., Secretary, Bureau of Mines, T S Ary, Director, (1989).
 - [33] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi, *Optimization by Simulated Anealing*. Science, May 671-680, (1983).
 - [34] V. Cerny, *Thermodynamical approach to the traveling salemán problem : an efficient simulation algorithm*. Journal of Optimization Theory And Applications : Vol. 45, No. 1, January, 41-51, (1985)
 - [35] R. Sanguthevar, *On Simulated Annealing and Nested Annealing*, Dept. of Computer and Information Science and Engineering, University of Florida, Gainesville, FL 32611, 1-12, (2000).
 - [36] R. Chelouah and P. Siarry, *A hybrid method combining continuous tabu search and Nelder-Mead simplex algorithms for the global optimization of multimínima functions*. European Journal of Operational Research , Volume 161, Issue 3, 636-654, (2005).
 - [37] M. Ouanes, M. Chebbah, A. Zidna, *Combination of two underestimators for univariate global optimization*, RAIRO-Operations Reseach, 52(1), 177-186, (2018).
 - [38] I.P. Androulakis, C.D. Marinas, C.A. Floudas, *α BB : A global optimization method for general constrained nonconvex problems*, J. Glob. Optim., 7, 337-363, (1995).
 - [39] A. Chikhi, N. Bouafia, *Mémoire de Master : Méthode hybride en optimisation globale*, Université de Mouloud Mammeri-Tizi Ouzou, (2013).

-
- [40] S. Skelboe, Computation of rational interval functions. *BIT Numerical Mathematics*, 14(1), 87-95, (1974).
 - [41] K. Ichida et Y. Fujii : An interval arithmetic method for global optimization, *Computing*, 23(1), 85-97, (1979).
 - [42] E.R. Hansen et W.G. William, *Global Optimization Using Interval Analysis*, Marcel Dekker Inc., New York, 2^{ème} édition, (2004).
 - [43] R.B. Kearfott, An interval branch and bound algorithm for bound constrained optimization problems, *Journal of Global optimization*, 2(3), 259-280, (1992).
 - [44] R. Hammer, M. Hocks, U. Kulishet, D. Ratz, *Numerical Toolbox for Verified Computing I*. Springer-Verlag, Berlin, (1993).
 - [45] H. Ratschek and J. Rokne, *New Computer Methods for Global Optimization*, Ellis Horwood Ltd., Chichester, (1988).
 - [46] E.R. Hansen, *Global Optimization using Interval Analysis*, Marcel Dekker, New York, (1992).
 - [47] F. Glover, Future paths for integer programming and links to artificial intelligence, *Computers and Operations Research*, 13 : 533-549, (1986).
 - [48] C.D. Maranas and C.A. Floudas, Global Minimum Potential Energy Conformations of small Molecules, *J.Glob.Opt.*, 4 :135, (1994a).
 - [49] I.P. Androulakis, C.D. Maranas and C.A. Floudas, α BB : A Global Optimization Method for General Constrained Nonconvex Problems, Kluwer Academic Publishers, Boston. Manufactured in the Netherlands, 1-27, (1995).
 - [50] B.A. Murtagh and M.A. Saunders, *MINOS5.0 Users Guide*, Systems Optimization Laboratory, Dept. of Operations Research, Stanford University, CA., Appendix A : MINOS5.0, Technical Report SOL 20-83, (1983).
 - [51] R. Horst and H. Tuy, *Global Optimization*, Springer-Verlag, Berlin, Germany, (1990).

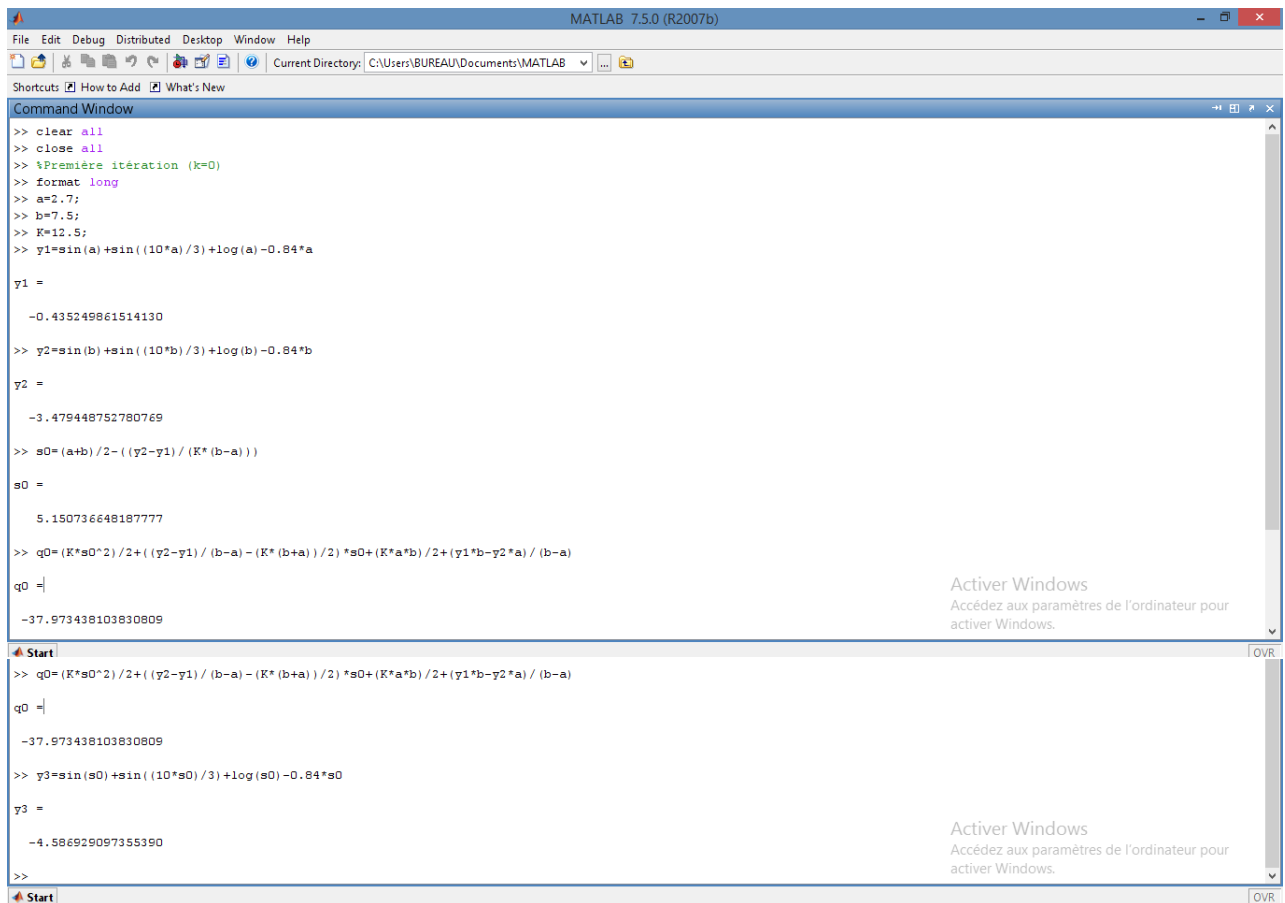
- [52] C.D. Maranas and C.A. Floudas, Global Optimization in Generalized Geometric Programming, Submitted to Comp. chem. Engnr, 351-369, (1994c).
- [53] [http://archives.umc.edu.dz/dz/bitstream/handle/123456789/8823/HAM6543.pdf?sequence = 1](http://archives.umc.edu.dz/dz/bitstream/handle/123456789/8823/HAM6543.pdf?sequence=1), HAM6543, (2016).
- [54] H. Horst and H. Tuy, Global optimization, Deterministic Approach, Springer-Verlag, Berlin, (1993).
- [55] R. Horst, P.M. Pardalos, Handbook of Global Optimization, Kluwer Academic Publishers, Dordrecht, (1995).
- [56] R. Horst, P.M. Pardalos and N.V. Thoai, Introduction to Global Optimization, Kluwer Academic Publishers, Dordrecht, (1995).
- [57] R.G. Strongin and Ya. D. Sergeyev, Global multidimensional optimization on parallel computer, Parallel Computing, vol. 18, 1259-1273, (1992).
- [58] <https://homepages.laas.fr/yariba/enseignement/manuel-matlab.pdf>.
- [59] E.G. Chrysanthos, A.F. Christodoulos, Tight convex underestimators for C^2 -continuous problems: I.univariate functions, Springer Science-Busness Media, 51-67, (2008).

Annexes

Simulation de la méthode de [21] sous Matlab

Annexe 1

Première itération



```
MATLAB 7.5.0 (R2007b)
File Edit Debug Distributed Desktop Window Help
Current Directory: C:\Users\BUREAU\Documents\MATLAB
Shortcuts How to Add What's New

Command Window
>> clear all
>> close all
>> %Première itération (k=0)
>> format long
>> a=2.7;
>> b=7.5;
>> K=12.5;
>> y1=sin(a)+sin((10*a)/3)+log(a)-0.84*a

y1 =
-0.435249861514130

>> y2=sin(b)+sin((10*b)/3)+log(b)-0.84*b

y2 =
-3.479448752780769

>> s0=(a+b)/2-((y2-y1)/(K*(b-a)))

s0 =
5.150736648187777

>> q0=(K*s0^2)/2+((y2-y1)/(b-a)-(K*(b+a))/2)*s0+(K*a*b)/2+(y1*b-y2*a)/(b-a)

q0 =
-37.973438103830809

>> q0=(K*s0^2)/2+((y2-y1)/(b-a)-(K*(b+a))/2)*s0+(K*a*b)/2+(y1*b-y2*a)/(b-a)

q0 =
-37.973438103830809

>> y3=sin(s0)+sin((10*s0)/3)+log(s0)-0.84*s0

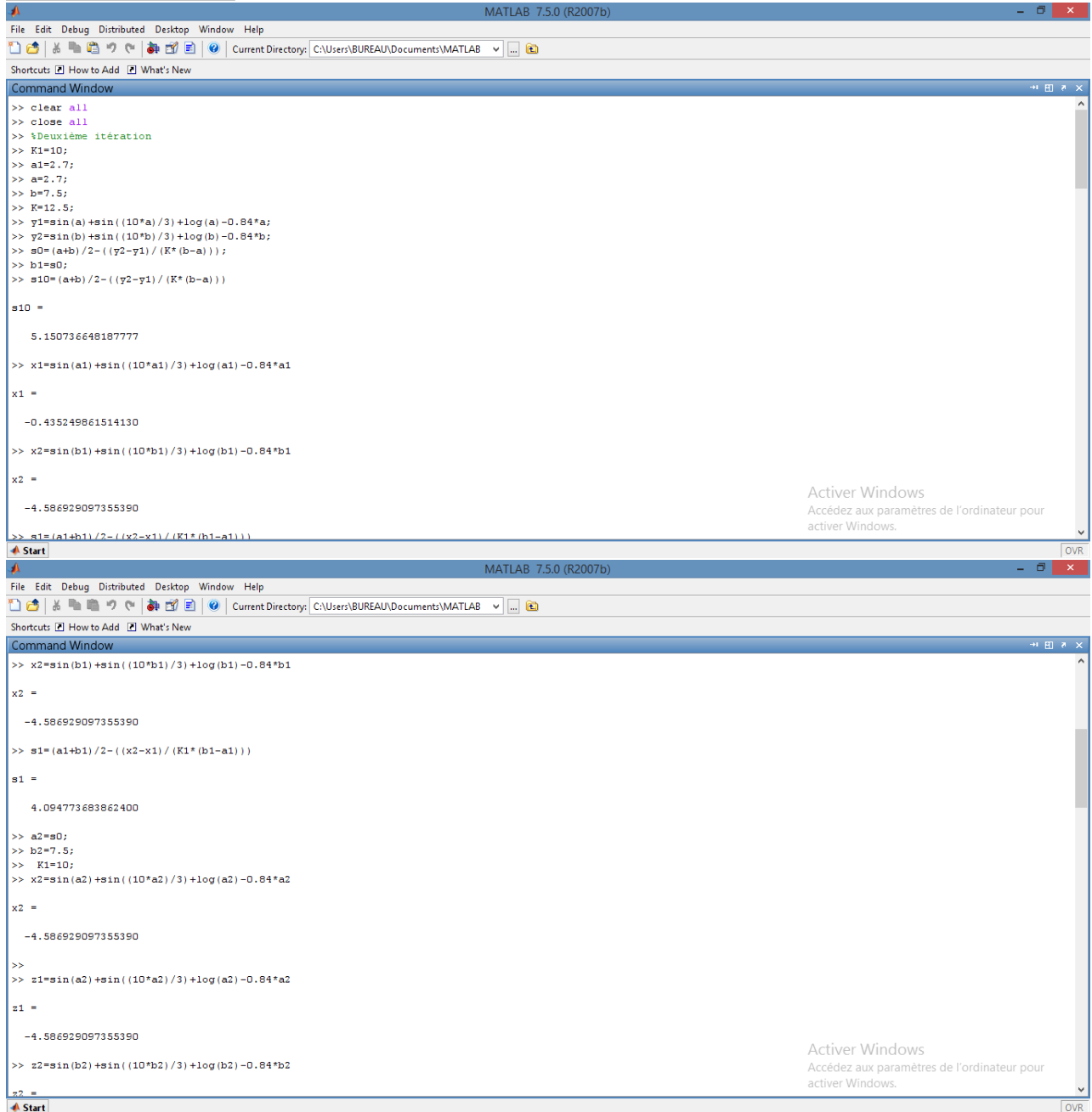
y3 =
-4.586929097355390

>>
```

FIGURE 3.7 – 1^{ère} itération de la méthode Branch and Bound avec une fonction borne inférieure quadratique simulée sous Matlab.

Annexe 2

Deuxième itération



The image displays two screenshots of the MATLAB 7.5.0 (R2007b) Command Window, showing the execution of a script for the second iteration of a numerical method.

Top Screenshot:

```

>> clear all
>> close all
>> %Deuxième itération
>> K1=10;
>> a1=2.7;
>> a=2.7;
>> b=7.5;
>> K=12.5;
>> y1=sin(a)+sin((10*a)/3)+log(a)-0.84*a;
>> y2=sin(b)+sin((10*b)/3)+log(b)-0.84*b;
>> s0=(a+b)/2-((y2-y1)/(K*(b-a)));
>> b1=s0;
>> s10=(a+b)/2-((y2-y1)/(K*(b-a)))

s10 =

    5.150736648187777

>> x1=sin(a1)+sin((10*a1)/3)+log(a1)-0.84*a1

x1 =

   -0.435249861514130

>> x2=sin(b1)+sin((10*b1)/3)+log(b1)-0.84*b1

x2 =

   -4.586929097355390

>> s1=(a1+b1)/2-((x2-x1)/(K1*(b1-a1)))
  
```

Bottom Screenshot:

```

>> x2=sin(b1)+sin((10*b1)/3)+log(b1)-0.84*b1

x2 =

   -4.586929097355390

>> s1=(a1+b1)/2-((x2-x1)/(K1*(b1-a1)))

s1 =

    4.094773683862400

>> a2=s0;
>> b2=7.5;
>> K1=10;
>> x2=sin(a2)+sin((10*a2)/3)+log(a2)-0.84*a2

x2 =

   -4.586929097355390

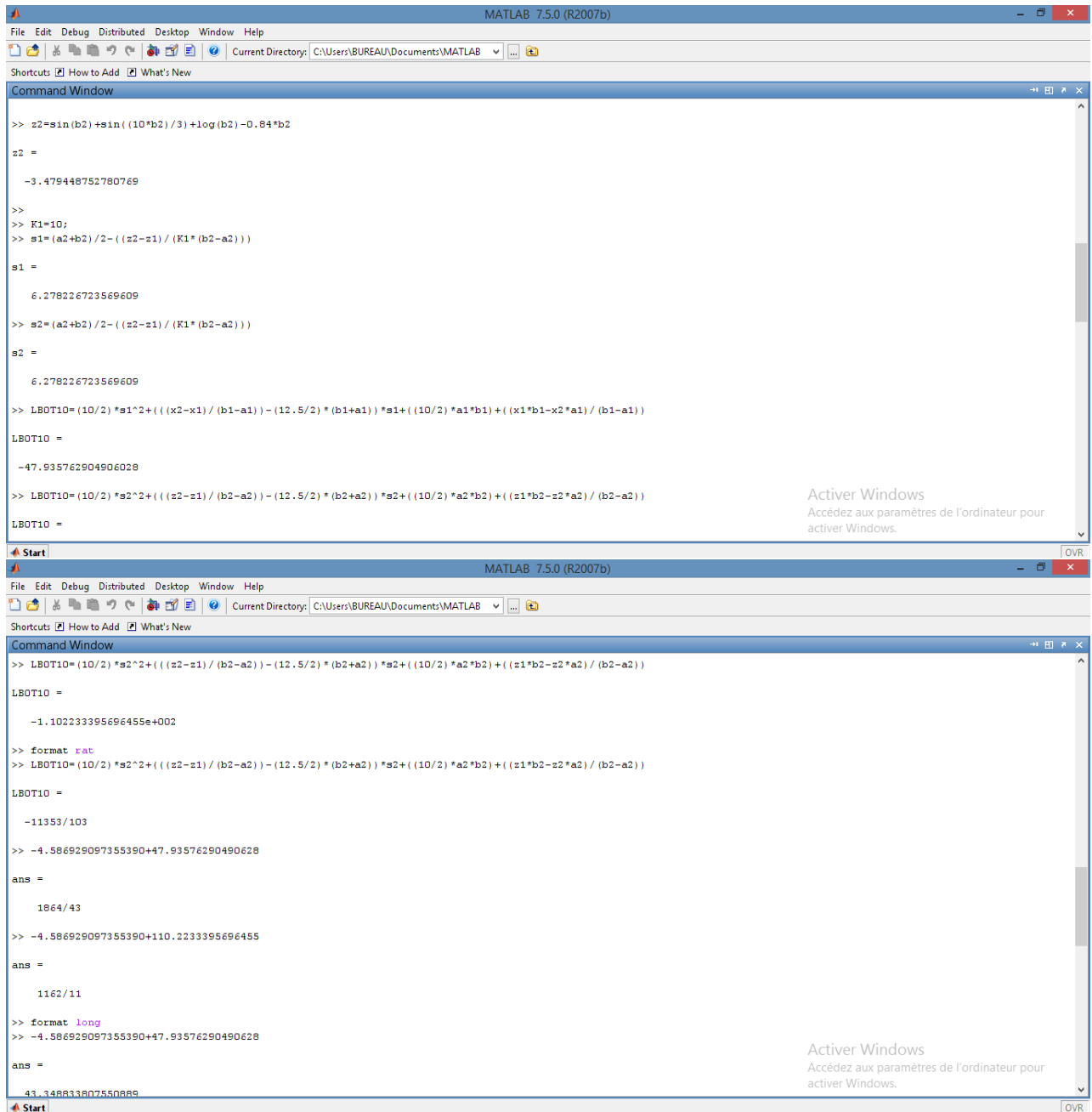
>>
>> z1=sin(a2)+sin((10*a2)/3)+log(a2)-0.84*a2

z1 =

   -4.586929097355390

>> z2=sin(b2)+sin((10*b2)/3)+log(b2)-0.84*b2

z2 =
  
```



MATLAB 7.5.0 (R2007b)

File Edit Debug Distributed Desktop Window Help

Current Directory: C:\Users\BUREAU\Documents\MATLAB

Shortcuts: How to Add What's New

Command Window

```
>> z2=sin(b2)+sin((10*b2)/3)+log(b2)-0.84*b2
z2 =
-3.479448752780769
>>
>> K1=10;
>> s1=(a2+b2)/2-((z2-z1)/(K1*(b2-a2)))
s1 =
6.278226723569609
>> s2=(a2+b2)/2-((z2-z1)/(K1*(b2-a2)))
s2 =
6.278226723569609
>> LBOT10=(10/2)*s1^2+(((x2-x1)/(b1-a1))-(12.5/2)*(b1+a1))*s1+((10/2)*a1*b1+((x1*b1-x2*a1)/(b1-a1))
LBOT10 =
-47.935762904906028
>> LBOT10=(10/2)*s2^2+(((z2-z1)/(b2-a2))-(12.5/2)*(b2+a2))*s2+((10/2)*a2*b2+((z1*b2-z2*a2)/(b2-a2))
LBOT10 =
```

Activer Windows
Accédez aux paramètres de l'ordinateur pour activer Windows.

Start

MATLAB 7.5.0 (R2007b)

File Edit Debug Distributed Desktop Window Help

Current Directory: C:\Users\BUREAU\Documents\MATLAB

Shortcuts: How to Add What's New

Command Window

```
>> LBOT10=(10/2)*s2^2+(((z2-z1)/(b2-a2))-(12.5/2)*(b2+a2))*s2+((10/2)*a2*b2+((z1*b2-z2*a2)/(b2-a2))
LBOT10 =
-1.102233395696455e+002
>> format rat
>> LBOT10=(10/2)*s2^2+(((z2-z1)/(b2-a2))-(12.5/2)*(b2+a2))*s2+((10/2)*a2*b2+((z1*b2-z2*a2)/(b2-a2))
LBOT10 =
-11353/103
>> -4.586929097355390+47.93576290490628
ans =
1864/43
>> -4.586929097355390+110.2233395696455
ans =
1162/11
>> format long
>> -4.586929097355390+47.93576290490628
ans =
43.348833807550889
```

Activer Windows
Accédez aux paramètres de l'ordinateur pour activer Windows.

Start

The figure consists of two screenshots of the MATLAB 7.5.0 (R2007b) Command Window. The first screenshot shows the following commands and outputs:

```
>> format long
>> -4.586929097355390+47.93576290490628

ans =

43.348833807550889

>> -4.586929097355390+47.93576290490628

ans =

43.348833807550889

>> s1=4.094773683862400

s1 =

4.094773683862400

>> s2=6.278226723569609

s2 =

6.278226723569609

>> K1=sin(s1)+sin((10*s1)/3)+log(s1)-0.84*s1

K1 =

-1.961850954449177
```

The second screenshot shows the following commands and outputs:

```
K1 =

-1.961850954449177

>> K2=sin(s2)+sin((10*s2)/3)+log(s2)-0.84*s2

K2 =

-2.567410400047483

>> UB0=-4.586929097355390

UB0 =

-4.586929097355390

>> LBO=-110.2233395696455

LBO =

-1.102233395696455e+002

>> G=UB0-LBO

G =

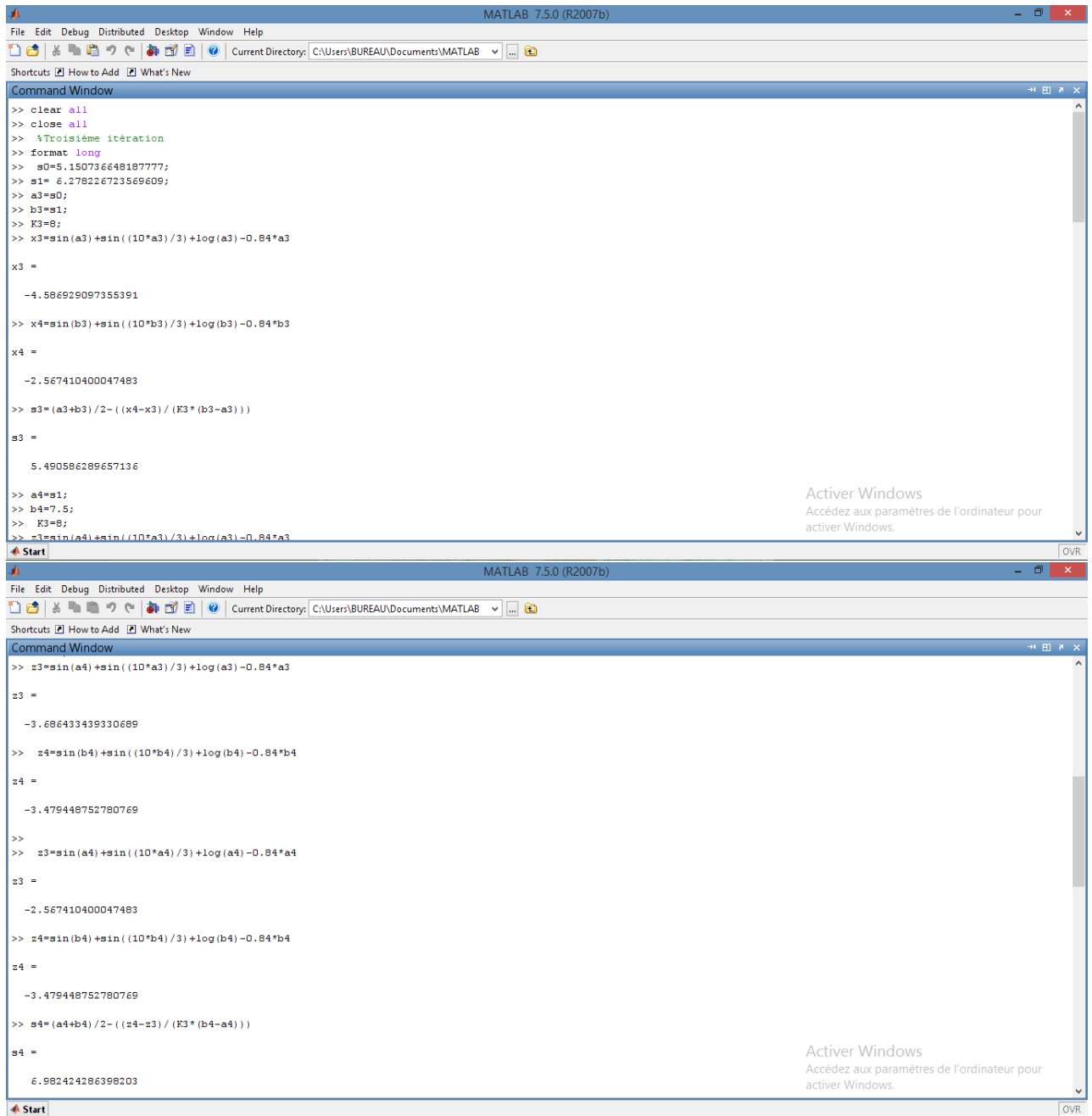
1.056364104722901e+002

>>
```

FIGURE 3.8 – 2^{ème} itération de la méthode Branch and Bound avec une fonction borne inférieure quadratique simulée sous Matlab.

Annexe 3

Troisième itération



The image displays two screenshots of the MATLAB 7.5.0 (R2007b) Command Window, showing a series of iterative calculations. The first screenshot shows the initialization of variables and the calculation of x3 and x4. The second screenshot shows the calculation of z3 and z4, and the final calculation of s4.

First Screenshot:

```

>> clear all
>> close all
>> %Troisième itération
>> format long
>> s0=5.150736648187777;
>> s1= 6.278226723569609;
>> a3=s0;
>> b3=s1;
>> K3=8;
>> x3=sin(a3)+sin((10*a3)/3)+log(a3)-0.84*a3

x3 =

-4.586929097355391

>> x4=sin(b3)+sin((10*b3)/3)+log(b3)-0.84*b3

x4 =

-2.567410400047483

>> s3=(a3+b3)/2-((x4-x3)/(K3*(b3-a3)))

s3 =

5.490586289657136

>> a4=s1;
>> b4=7.5;
>> K3=8;
>> z3=sin(a4)+sin((10*a3)/3)+log(a3)-0.84*a3

```

Second Screenshot:

```

>> z3=sin(a4)+sin((10*a3)/3)+log(a3)-0.84*a3

z3 =

-3.686433439330689

>> z4=sin(b4)+sin((10*b4)/3)+log(b4)-0.84*b4

z4 =

-3.479448752780769

>> z3=sin(a4)+sin((10*a4)/3)+log(a4)-0.84*a4

z3 =

-2.567410400047483

>> z4=sin(b4)+sin((10*b4)/3)+log(b4)-0.84*b4

z4 =

-3.479448752780769

>> s4=(a4+b4)/2-((z4-z3)/(K3*(b4-a4)))

s4 =

6.982424286398203

```

The figure consists of two screenshots of the MATLAB 7.5.0 (R2007b) Command Window. The top screenshot shows the calculation of LB11 and UB1. The bottom screenshot shows the calculation of LB12 and the final UB1-LB12 difference.

Top Screenshot:

```

>> LBT11=(8/2)*s3^2+(((x4-x3)/(b3-a3))-(10/2)*(b3+a3))*s3+((8/2)*a3*b3+((x3*b3-x4*a3)/(b3-a3))

LBT11 =

-67.800629806547505

>> LBT12=(8/2)*s4^2+(((z4-z3)/(b4-a4))-(10/2)*(b4+a4))*s4+((8/2)*a4*b4+((z3*b4-z4*a4)/(b4-a4))

LBT12 =

-1.007564121281757e+002

>> UB1=-4.586929097355391

UB1 =

-4.586929097355391

>> LB11=-67.800629806547505

LB11 =

-67.800629806547505

>> UB1-LB11

ans =

63.213700709192111

```

Bottom Screenshot:

```

>> UB1-LB11

ans =

63.213700709192111

>> UB1=-4.586929097355391

UB1 =

-4.586929097355391

>> LB12=-100.7564121281757

LB12 =

-1.007564121281757e+002

>> UB1=-4.586929097355391

UB1 =

-4.586929097355391

>> UB1-LB12

ans =

96.169483030820302

>>

```

FIGURE 3.9 – 3^{ème} itération de la méthode Branch and Bound avec une fonction borne inférieure quadratique simulée sous Matlab.

الملخص

تتناول هذه الرسالة التحسين الشامل : الخوارزميات والتطبيقات. يجذب التحسين الشامل انتباه الباحثين ليس فقط لأنه يحدث في العديد من التطبيقات الحقيقية و لكن لأن مشاكله تسمح بوجود إمتداد للحالة متعددة الأبعاد. كما أنه يوفر حلاً أفضل من أي شخص آخر لا يمكن العثور على أفضل من هذا الأخير (أفضل حل). في هذه الرسالة، ذكرنا المفاهيم الأساسية للتحسين المحدب وغير المحدب. محدب. اذكر العموميات والطرق (التعريف ، المبدأ ، الخوارزمية ، النظريات وتقارب البراهين). محاكاة وإعطاء أمثلة عملية تثبت كفاءة وتقارب طريقة - دالة تربيعية منخفضة الحد [٢١] ، من خوارزمية طريقة الفرع والربط مع وظيفة ملزمة جديدة أقل [٣٧] مقارنة بالخوارزمية التي تمت دراستها في المقالة [٥٩]. الكلمات الرئيسية : التحسين الشامل، التحسين المحدب وغير المحدب ، الطريقة الحتمية الطريقة العشوائية ، الطريقة الهجينة ، المحاكاة العددية.

Résumé

Ce mémoire traite l'optimisation globale : algorithmes et applications. L'optimisation globale attirent l'attention des chercheurs non seulement parce qu'elle surviennent dans plusieurs applications réelles, mais parce que ses problèmes admettent une extension au cas multidimensionnel. Aussi, elle fournit une meilleure solution, que personne ne peut trouver meilleure que cette dernière (meilleure solution). Dans ce mémoire, nous avons rappelé les notions de base de l'optimisation convexe et non convexe. Donner les généralités, les méthodes (définition, principe, algorithme, théorèmes et preuves de convergence). Simuler, et donner des exemples pratiques qui prouvent l'efficacité et la convergence de la méthode BB avec une fonction borne inférieure quadratique [21], de l'algorithme de la méthode Branch and Bound avec une nouvelle fonction borne inférieure [37], par rapport à l'algorithme étudié dans l'article [59].

Mots clés : Optimisation convexe et non convexe, optimisation globale, méthodes déterministes, méthodes stochastiques, méthodes hybrides, simulation numérique.

Abstract

This dissertation deals with global optimization: algorithms and applications. Global optimization attracts the attention of researchers not only because it occurs in several real applications, but because its problems admit a extension to the multidimensional case. Also, it provides a better solution than anyone else. cannot find better than the latter (best solution). In this dissertation, we have recalled the basic notions of convex and non-convex optimization. convex. Give the generalities, the methods (definition, principle, algorithm, theorems and convergence proofs). Simulate, and give practical examples that prove the

efficiency and convergence of the BB method with a quadratic lower bound function [21], of the Branch and Bound method algorithm with a new bound function lower [37], compared to the algorithm studied in the article [59].

Keywords: Convex and non-convex optimization, global optimization, deterministic method, stochastic method, hybrid method, numerical simulation.