MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITÉ MOULOUD MAMMERI, TIZI-OUZOU

FACULTÉ DES SCIENCES

DÉPARTEMENT DE MATHEMATIQUES

THÈSE DE DOCTORAT LMD 3ème CYCLE

Filière : Mathématiques

Spécialité: Recherche opérationnelle

Présentée par :

ZERROUKI Djamel

Sujet :

Optimisation globale et optimisation semi-infinie : théorie, algorithmes et applications.

Devant le jury d'examen composé de :

Mr. MERAKEB Abdelkader	Professeur	UMMTO	Président
Mr. OUANES Mohand	Professeur	UMMTO	Rapporteur
Mr. MARTHON Philippe	Professeur Émérite	Univ. Toulouse, France	Examinateur
Mr. BOUROUBI Sadek	Professeur	USTHB	Examinateur
Mr. MOULAI Mustapha	Professeur	USTHB	Examinateur
Mr. GOUBI Mouloud	Maître de conférence classe A	UMMTO	Examinateur

Année universitaire 2023/2024

Remerciements

Avant tout, je remercie dieu de m'avoir donné le courage pour mener à bien ce travail, malgré tous les obstacles.

Je remercie mon encadreur monsieur ***OUANES Mohand****, professeur à l'université Mouloud Mammeri de Tizi-Ouzou, pour m'avoir encadré et orienté durant mon travail. Je lui témoigne ma respectueuse gratitude.*

J'adresse mes vifs remerciements aux membres du jury pour avoir pris le temps de lire et d'examiner ce travail.

Je ne peux pas oublier de remercier mes parents, ma famille et tous mes proches, pour leurs soutiens et leurs encouragements.

TABLE DES MATIÈRES

TABLE DES FIGURES

LISTE DES TABLEAUX

INTRODUCTION GÉNÉRALE

La recherche opérationnelle [20, 21] peut être définie comme un ensemble de méthodes et de techniques rationnelles visant à trouver les meilleures options opérationnelles afin d'obtenir les résultats souhaités ou les meilleurs possibles. Elle fournit des modèles conceptuels conçus pour analyser et contrôler des situations complexes, permettant aux décideurs de comprendre, d'évaluer les problèmes et d'arbitrer ou de faire les choix les plus efficaces.

L'optimisation globale est une branche des mathématiques appliquées qui cherche à trouver les minima ou les maxima globaux de fonctions sur un ensemble donné. La programmation mathématique, une sous-discipline de l'optimisation [20, 21, 22, 23], traite de la minimisation ou de la maximisation de fonctions sous contraintes multiples [24, 25, 26]. L'optimisation est essentielle dans de nombreux domaines, de la conception de moteurs électriques à la météorologie, à la biologie mathématique et en visant à maximiser les gains, minimiser les coûts, ou améliorer les processus de manière optimale. Cette discipline fait partie intégrante de la recherche opérationnelle.

Les problèmes d'optimisation proposés dans la littérature peuvent être classés en deux catégories : les problèmes convexes avec un cadre théorique large et une variété d'outils numériques, et les problèmes non convexes pour lesquels la recherche a connu une avancée fulgurante en ce sens que beaucoup d'articles ont vu le jour dans ce contexte. Les dernières décennies, l'optimisation non convexe ou globale a connu des développements spectaculaires car beaucoup de problèmes d'optimisation dans la pratique sont de nature non convexe. Par conséquence, ces problèmes peuvent admettre un grand nombre de solutions locales. Donc, il est évident que les recherches réalisées du point de vue théorique et techniques numériques visant à atteindre l'optimalité globale, nous permettent d'améliorer la qualité de notre vie.

Il n'y a pas si longtemps, le domaine d'optimisation globale était peu connu, en effet, il était difficile de trouver des solutions optimales dans le contexte de l'optimisation non convexe par des méthodes classiques. Ainsi, afin de trouver des solutions, les méthodes utilisées sont le

lagrangien augmenté, la méthode SQP, les méthodes de type Newton, la méthode du gradient, etc. Alors que la recherche de solutions optimales multiples paraissaient utopique en général, cela se traduit également par des apports considérables en ressources (temps d'exécution, quantité de mémoire utilisée). En ce sens, l'optimisation globale apporte un nouveau souffle, de nouvelles orientations, et surtout de nouvelles méthodes, tout en permettant de résoudre des problèmes liés à l'optimisation citée plus haut.

Généralement, les méthodes d'optimisation globale peuvent être classées en deux catégories : les méthodes stochastiques (non déterministes) et les méthodes déterministes. Plusieurs méthodes stochastiques sont faciles à mettre en œuvre et ne nécessitent pas une forte régularité sur les fonctions qui définissent le problème à résoudre. En général, ils sont théoriquement limités et ont quelques inconvénients tels que : l'absence de garantie sur la qualité des solutions obtenues. Pour cette catégorie on peut citer quelques méthodes, par exemple : la méthode recuit simulé [27], les algorithmes génétiques [28].

Contrairement à la première catégorie, les méthodes globales déterministes nécessitent généralement des propriétés de régularité sur les fonctions qui définissent le problème à résoudre. Ces méthodes offrent une certitude pour obtenir le minimum globale et ne laissent rien pour le hasard. Elles utilisent des méthodes d'approximation, de décomposition et des techniques de réduction dimensionnelle, pour localiser les solutions optimales. Au cours de ces dernières années, cette catégorie a connu des développements important par Christodoulos Floudas [29, 11, 10], Le Thi Hoai An [30, 13] et d'autres [31, 1]. Plusieurs algorithmes ont été développés à l'aide de la structure particulière de la fonction ou du problème à résoudre. Par exemple la détermination d'une solution optimale se fait à l'aide de l'information de la matrice hessienne, les dérivées ou d'autres informations concernant la structure du problème à résoudre.

Parmi les méthodes les plus prometteuses de résolution des problèmes généraux d'optimisation globale, il existe celles qui introduisent les sous-estimateurs convexes pour les fonctions

non convexes, qui servent à trouver une séquence des bornes inférieures et supérieures qui convergent vers la solution optimale globale.

Dans cette thèse, on s'intéresse aux méthodes d'optimisation globale utilisant des sous-estimateurs, les informations sur les matrices hessiennes d'intervalles et les algorithmes de Séparation et d'évaluation progressive pour localiser les solutions optimales.

Le premier chapitre est consacré aux propriétés des ensembles, des fonctions convexes et d'autres concepts préliminaires en raison de leur importance dans la théorie de l'optimisation. Nous donnerons les définitions concernant les types des minimums et leurs existences. Puis, on passe à l'étude théorique et algorithmique des différents classes des programmes mathématiques d'optimisation convexe avec les différentes méthodes de résolution de ces programmes.

Dans le deuxième chapitre, nous nous intéressons aux méthodes de résolution pour les problèmes d'optimisation globale. nous commençons par les méthodes déterministes de résolution en optimisation globale. Puis, on passe aux méthodes les plus connues parmi celles qui utilisent les algorithmes de Séparation et d'évaluation progressive, et les sous-estimateurs convexes pour relaxer le problème non convexe à un problème convexe qui sera facile à résoudre afin de déterminer une suite des bornes inférieures qui convergent vers la solution optimale globale du problème non convexe original.

Dans le troisième chapitre, nous traitons les bornes sur les valeurs propres des matrices d'intervalles symétriques réels. Nous présentons une nouvelle méthode qui calcule des bornes sur les valeurs propres des matrices d'intervalles symétriques réels. L'objectif est d'appliquer la méthode proposée pour calculer les bornes inférieures sur les valeurs propres de la matrice hessienne d'intervalles symétrique d'une fonction non convexe dans la méthode αBB [10] et de les utiliser pour produire un sous-estimateur plus efficace qui améliore l'algorithme αBB pour résoudre les problèmes d'optimisation globale. Enfin, une collection de problèmes tests trouvés dans la littérature ont été résolus efficacement et les performances de la méthode

proposée sont comparées à celles d'autres méthodes.

Dans le quatrième chapitre nous présentons une étude sur l'optimisation semi-infinie, ainsi que les principales méthodes de résolution pour les programmes semi-infinis linéaires et non linéaires.

CHAPITRE 1

OPTIMISATION CONVEXE

1.1 Introduction

Il y a beaucoup de travaux et de publications qui sont réalisés par des chercheurs sur cette thématique les dernières décennies [20, 21, 32, 33, 34]. L'optimisation est une branche des mathématiques appliquées, cherchant à analyser et à résoudre analytiquement ou numériquement, les problèmes consistant à trouver un optimum sur un certain ensemble. Le mot « optimum » signifie meilleur.

1.2 Optimisation non linéaire

On considère le problème suivant :

$$(P) \begin{cases} minf(x) \\ x \in B \subset \mathbb{R}^n \end{cases}$$

Où $f : \mathbb{R}^n \to \mathbb{R}$ est une fonction non linéaire continue. L'optimisation non linéaire ou la programmation non linéaire consiste à trouver le vecteur x^* pour que la valeur $f(x^*)$ est la valeur minimale définie sur l'ensemble B. Le problème (P) s'appelle un problème de minimisation. On peut considérer le problème de maximisation comme un problème de minimisation lorsqu'on remplace la fonction f par $-f$ c'est-à-dire maximiser la fonction f revient à minimiser la fonction $-f$.

1.3 Classification des problèmes d'optimisation non linéaire

Les problèmes d'optimisation non linéaire sont classifiés en deux classes en fonction de l'espace de recherche d'un optimum, autrement dit l'ensemble faisable.

1.3.1 Problèmes sans contraintes

Les problèmes d'optimisation sans contraintes, ce sont ceux dont l'ensemble de recherche (domaine) est $B = \mathbb{R}^n$, ou tout simplement un pavé de $\mathbb{R}^n$, i.e., $B = \prod_{i=1}^{n}[\underline{x_i}, \overline{x_i}]$ avec $(\underline{x_1}, \underline{x_2}, \underline{x_3}, \ldots, \underline{x_n}) \in \mathbb{R}^n$, $(\overline{x_1}, \overline{x_2}, \overline{x_3}, \ldots, \overline{x_n}) \in \mathbb{R}^n$ et $\underline{x_i} < \overline{x_i}$.

1.3.2 Problèmes avec contraintes

Les problèmes d'optimisation avec contraintes, ce sont ceux dont l'ensemble de recherche est le même que l'espace de recherche des problèmes sans contraintes, avec des conditions supplémentaires sur le domaine de recherche. Ces conditions sont des fonctions qui peuvent être non linéaires s'appellent des contraintes d'égalité ou d'inégalité et permettent de limiter le domaine de recherche.

1.4 Propriété sur la convexité et autres concepts préliminaires

Nous allons ici rappeler quelques propriétés importantes sur la convexité, la théorie des ensembles et leur topologie [20, 35].

1.4.1 Ensembles convexes

La convexité est un outil très important dans l'optimisation non linéaire à cause de l'efficacité des algorithmes basés sur la convexité pour résoudre les problèmes d'optimisation convexe.

Définition 1.4.1. *Soit un ensemble $E \in \mathbb{R}^n$. L'ensemble E est convexe si :*
$\forall x, y \in E, \quad \lambda x + (1-\lambda)y \in E \quad$ *avec* $\lambda \in [0, 1]$.

Définition 1.4.2. *soient les p vecteurs $x_1, x_2, \ldots, x_p$ avec $x_i \in \mathbb{R}^n$ pour $i = 1, 2, \ldots, p$.*
En général, l'ensemble E est convexe si toute combinaison convexe des points de E est dans

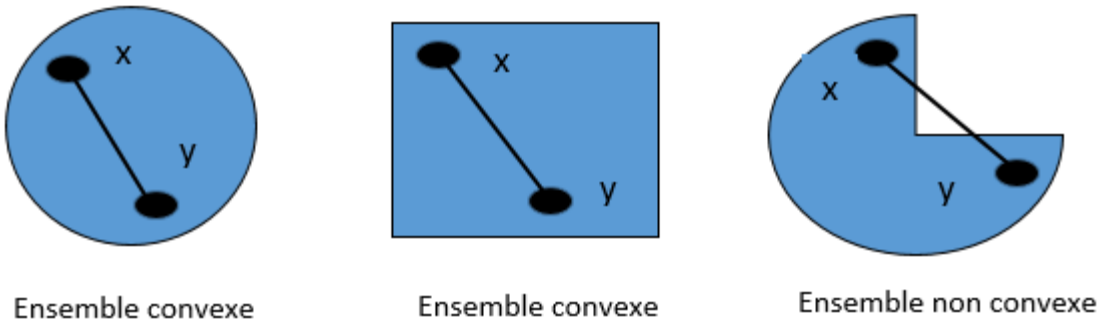

FIGURE 1.1 – Ensembles convexes et ensemble non convexe

E. Le point x est un combinaison convexe des p vecteurs $x_1, x_2, \ldots, x_p$ avec $x_i \in \mathbb{R}^n$, s'il existe $\lambda_1, \lambda_2, \ldots, \lambda_p$ avec $\lambda_i \in \mathbb{R}$ tel que $x = \sum_{i=1}^{p} \lambda_i x_i$ *avec* $\sum_{i=1}^{p} \lambda_i = 1$ *et* $\lambda_i \geq 0$ *pour* $i = 1, 2, \ldots, p$.

Propriété 1.4.1. *Soit E un ensemble convexe de $\mathbb{R}^n$, l'ensemble $C = \{x_c \in \mathbb{R}^n, x \in E | x_c = \lambda x\}$ avec $\lambda \in \mathbb{R}$, est convexe.*

Propriété 1.4.2. *Soient deux ensembles convexes E_1 et E_2 de $\mathbb{R}^n$.*
L'ensemble $C = \{x_c \in \mathbb{R}^n, x_1 \in E_1, x_2 \in E_2 | x_c = x_1 + x_2\}$, est convexe.

Propriété 1.4.3. *Soient deux ensembles convexes E_1 et E_2 de $\mathbb{R}^n$. L'ensemble $C = E_1 \cap E_2$ est convexe.*

L'intersection d'un nombre fini d'ensembles convexes est un ensemble convexe.

1.4.2 Fonctions convexes

Définition 1.4.3. *Soit une fonction $f : E \mapsto \mathbb{R}$, est dite convexe si :* $\forall x_1, x_2 \in E$ *et* $\forall \lambda \in [0, 1]$ *on a* $f(\lambda x_1 + (1 - \lambda) x_2) \leq \lambda f(x_1) + (1 - \lambda) f(x_2)$.
La fonction f est dite strictement convexe si :
$\forall x_1, x_2 \in E$ *avec* $x_1 \neq x_2$ *et* $\forall \lambda \in [0, 1]$ *on a* $f(\lambda x_1 + (1-\lambda) x_2) < \lambda f(x_1) + (1-\lambda) f(x_2)$.

Propriété 1.4.4. *Une fonction $f : E \to \mathbb{R}$, est concave si $-f$ est convexe.*

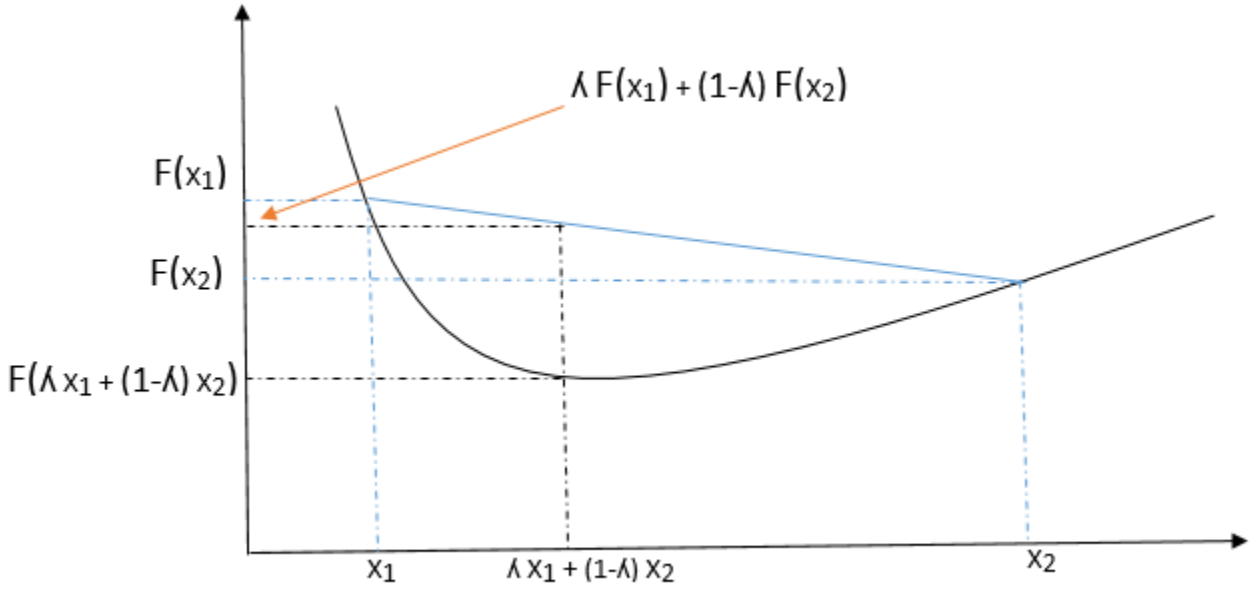

FIGURE 1.2 – Représentation d'une fonction convexe

Définition 1.4.4. *Soit $f : \mathbb{R}^n \to \mathbb{R}$. Si la fonction f est deux fois différentiable, alors la matrice hessienne associe à f, notée $H_f(x)$, est une matrice réelle carrée symétrique d'ordre n.*

Définition 1.4.5. *Soit H une matrice carrée d'ordre n. Si la matrice H est symétrique, alors toutes les valeurs propres de la matrice H, sont dans $\mathbb{R}$.*

Propriété 1.4.5. *Une matrice carrée H d'ordre n, est dite semi définie positive, si et seulement si toutes les valeurs propres λ_i de H, pour $i = 1, \ldots, n$, sont positives ou nulles.*

Propriété 1.4.6. *Une fonction $f : \mathbb{R}^n \to \mathbb{R}$ deux fois différentiable, est dite convexe, si et seulement si la matrice hessienne associe à f, est semi définie positive.*

Propriété 1.4.7. *Soient $f_1, f_2, \ldots, f_p$ des fonctions définies par $f_i : E \to \mathbb{R}$.*
$f = \sum_{i=1}^{p} f_i$ est une fonction convexe.

Théorème 1.4.1 ([36]). *Soit $f_i : \mathbb{R}^n \mapsto \mathbb{R}$ pour $i = 1, \ldots, n$, des fonctions convexes, alors*

$$E_c = \{x \in \mathbb{R}^n : f_i(x) \leq 0, i = 1, \ldots, n\}$$

est un ensemble convexe.

Démonstration. Soit $x_1, x_2 \in E_c$, alors $f_i(x_1) \leq 0$ et $f_i(x_2) \leq 0$ pour $i = 1, \ldots, n$.
soit $x = \lambda x_1 + (1 - \lambda)x_2$, avec $\lambda \in [0, 1]$, alors

$$f_i(x) = f_i(\lambda x_1 + (1 - \lambda)x_2)$$

$$\leq \lambda f_i(x_1) + (1 - \lambda) f_i(x_2) \leq \lambda 0 + (1 - \lambda)0 \leq 0 \Rightarrow f_i(x) \leq 0.$$

Alors $x \in E_c$, et E_c est un ensemble convexe. □

Définition 1.4.6. *Une fonction $f : E \to \mathbb{R}$, est dite quasi-convexe, si*

$$\{x \in E, f(x) \leq \alpha, \forall \alpha \in \mathbb{R}\}$$

est convexe.

Propriété 1.4.8. *Une fonction $f : E \to \mathbb{R}$, est dite quasi-convexe, si et seulement si*
$\forall x_1, x_2 \in E$ *et* $\forall \lambda \in [0, 1]$ *on a* $f(\lambda x_1 + (1 - \lambda)x_2) \leq \max\{f(x_1), f(x_2)\}$.
Une fonction f est quasi-concave, si et seulement si $-f$ est quasi-convexe.

Propriété 1.4.9. *Une fonction $f : E \to \mathbb{R}$. si f est convexe alors f est quasi-convexe.*

Propriété 1.4.10. *Une fonction $f : E \in \mathbb{R} \to \mathbb{R}$ est quasi-convexe si et seulement si au moins l'une des conditions suivantes est vérifiée :*

1. *f est non croissante.*
2. *f est non décroissante.*

3. *Il existe $a, b \in E$ tel que pour $a \leq b$ la fonction f est non croissante et pour $a \geq b$ la fonction f est non décroissante.*

1.5 Minimum local et minimum global

Soit la fonction continue $f : E \to \mathbb{R}$ avec $E \subseteq \mathbb{R}^n$. On considère le problème d'optimisation suivant :

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

Définition 1.5.1 (Minimum local)**.** *Un point $x^{local} \in E$ est un minimum local de f, si $f(x^{local}) \leq f(x)$ pour tout x dans un voisinage V de x^{local}.*

Définition 1.5.2 (Minimum local au sens strict)**.** *Un point $x^{local} \in E$ est un minimum local de f au sens strict, si $f(x^{local}) < f(x)$ pour tout x dans un voisinage V de x^{local} avec $x \neq x^{local}$.*

Définition 1.5.3 (Minimum global)**.** *Un point $x^* \in E$ est un minimum global de f, si $f(x^*) \leq f(x)$ pour tout x dans E.*

Remarque 1.5.1. *Un minimum global est un minimum local. La réciproque n'est pas toujours vraie.*

1.5.1 Théorème d'existence d'un minimum global

Théorème 1.5.1 (Weirstrass [37])**.** *Si $f : E \mapsto \mathbb{R}$ est une fonction continue sur un ensemble non vide compact E, alors le problème (P) admet au moins une solution (c'est à dire la fonction f admet au moins un minimum global dans l'ensemble E).*

Le théorème suivant est un cas particulier du théorème précèdent.

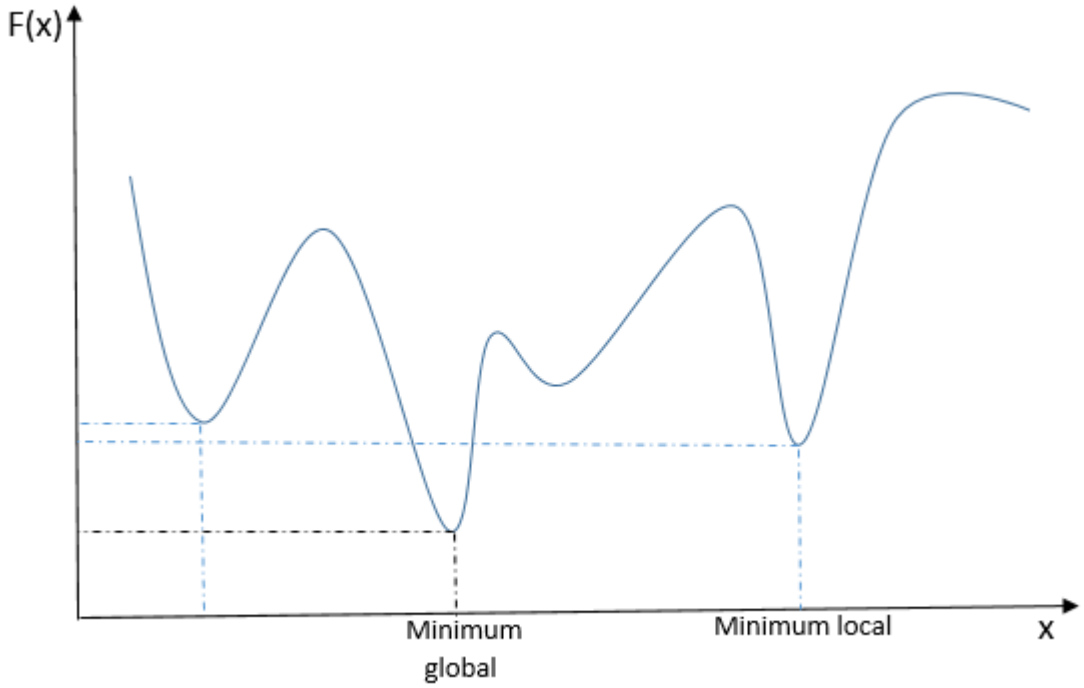

FIGURE 1.3 – Minimum local et minimum global

Théorème 1.5.2 ([38]). *Soit $f : \mathbb{R}^n \mapsto \mathbb{R}$ une fonction continue. Si $\lim_{\|x\|\to+\infty} f(x) = +\infty$, alors le problème (P) admet au moins une solution (c'est à dire la fonction f admet au moins un minimum global dans $\mathbb{R}^n$).*

Théorème 1.5.3 ([32]). *Si $f : X \subset \mathbb{R}^n \mapsto \mathbb{R}$ est une fonction convexe sur un ensemble convexe X, alors tout minimum local est un minimum global de f.*

1.6 Programmation linéaire et programmation quadratique convexe

La programmation linéaire [20, 22] est une partie de l'optimisation convexe. Un problème de la programmation linéaire consiste à optimiser une fonction objectif linéaire sous contraintes linéaires. De manière similaire, on peut rappeler que la programmation quadratique [34, 39, 40], consiste à optimiser une fonction objectif quadratique sous contraintes

linéaire et/ou quadratiques.

1.6.1 Problème de la programmation linéaire

On considère le problème linéaire ou le programme linéaire suivant :

$$(PL)\begin{cases} \sum_{i=1}^{n} c_i x_i \to \min \\ \sum_{i=1}^{n} a_{ij} x_i \leq b_j \quad (j = 1, \ldots, m) \\ x_i \geq 0 \quad (i = 1, \ldots, n) \end{cases}$$

On peut écrire ce problème sous la forme suivante :

$$(PL)\begin{cases} C'x \to \min \\ Ax \leq b \\ \forall x \in \mathbb{R}_+^n \end{cases}$$

où $A \in \mathbb{R}^{m \times n}$ une matrice réelle $(m \times n)$ supposée de plein rang $(rang(A) = m \leq n)$, $C \in \mathbb{R}^n$, $b \in \mathbb{R}^m$ et $x \in \mathbb{R}^n$.

1.6.2 Formulations des programmes linéaires

Il existe deux formes générales d'un programme linéaire avec la condition de réalisabilité de l'ensemble des variables $x_i \geq 0 \quad (i = 1, \ldots, n)$.

La forme canonique

$$(PL)\begin{cases} C'x \to \min \\ Ax \leq b \\ x_i \geq 0 \end{cases}$$

et la forme standard

$$(P)\begin{cases} C'x \to \min \\ Ax = b \\ x_i \geq 0 \end{cases}$$

On peut passer de la forme canonique à la forme standard et vice-versa en utilisant les opérateurs élémentaires.

Définition 1.6.1. *Les solutions du programme linéaire (P) sont données comme suit :*

1. ***Solution*** *est tout vecteur $x \in \mathbb{R}^n$ satisfaisant la contrainte $Ax = b$.*
2. ***Solution réalisable*** *est tout vecteur $x \in \mathbb{R}^n_+$ satisfaisant la contrainte $Ax = b$.*
3. ***Solution optimale*** *est toute solution réalisable noté $x^* \in \mathbb{R}^n_+$ qui optimise la fonction objectif $C'x$.*
4. ***Solution de base*** *est tout vecteur $x \in \mathbb{R}^n$ satisfaisant la contrainte $Ax = b$ et composé de $n - m$ variables nulles et m variables qui constituent une base.*
5. ***Solution de base réalisable*** *est toute solution de base satisfaisant $x \in \mathbb{R}^n_+$.*

Le problème (P) sous forme standard, admet une infinité de solutions réalisables.

On peut définir le dual du problème (P) comme suit :

$$(D) \begin{cases} b'y \to \max \\ A'y \leq C \\ y \in \mathbb{R}^m \end{cases}$$

Le programme (P) est appelé le primal et son dual est le programme (D) et vice-versa.

Le programme dual (D) permet d'obtenir la solution optimale du programme primal (P) en résolvant le programme dual.

1.6.3 Théorèmes généraux de la programmation linéaire

Théorème 1.6.1 ([20, 22])**.** *Soient x et y deux solutions réalisables du programme primal (P) et du programme dual (D) respectivement, alors $C'x \geq b'y$.*

Théorème 1.6.2 ([20, 22])**.** *Soient x^* et y^* deux solutions réalisables de programme primal (P) et de programme dual (D) respectivement. Si $C'x^* = b'y^*$, alors x^* et y^* sont deux solutions optimales du primal et du dual respectivement.*

Théorème 1.6.3 ([20, 22]). *On considère le programme linéaire sous forme standard* (P).

- *Si le programme* (P) *possède une solution réalisable finie, alors il possède une solution de base réalisable.*
- *Si le programme* (P) *possède une solution optimale finie, alors il possède une solution optimale de base réalisable.*

1.6.4 Méthodes de résolution d'un programme linéaire

Dans cette partie, nous rappelons quelques méthodes pour la résolution des problèmes de la programmation linéaire [33, 41, 42].

1. Méthode du simplexe [33]

La méthode du simplexe est un algorithme exact itératif inventé en 1947 par le mathématicien George Dantzig. Cet algorithme permet d'effectuer une exploration dirigée de l'ensemble des solutions de base réalisables, qui est un ensemble des sommets défini par le système des contraintes du problème de la programmation linéaire. La méthode du simplexe est la méthode la plus utilisée pour résoudre des problèmes de la programmation linéaire. Du fait que le nombre des sommets est fini, la valeur optimale est finie, l'algorithme du simplexe converge en un nombre d'itérations fini et ne dépasse pas C_n^m itérations. L'un des problèmes de l'algorithme du simplexe est qu'il risque de cycler, mais il existe des méthode adéquates pour endiguer cette difficulté, par exemple les techniques de Bland 1974 [33]. En effet, l'algorithme du simplexe est exact et fini mais sa complexité dans le pire des cas peut être exponentiel.

2. Méthodes des points intérieurs [41, 42]

Une méthode de points intérieurs est une procédure de résolution générant une suite de points appartenant à l'intérieur de domaine réalisable. Grâce à leur convergence polynomiale, la méthode de points intérieurs est l'une des méthodes les plus utilisées et les plus efficaces dans la résolution des problèmes de la programmation linéaire.

Nous distinguons quatre différents types de méthodes de points intérieurs :

1. Les méthodes projectives.
2. Les méthodes affines.
3. Les méthodes de réduction du potentiel.
4. Les méthodes de trajectoire centrale.

a) Méthodes projectives [23, 41, 42]

Les méthodes projectives sont basées de l'algorithme de Karmarkar [43]. Ces méthodes impliquent des transformations projectives d'un simplexe dans lui même. En 1984, Karmarkar a développé un algorithme qui utilise l'approche non linéaire pour la résolution des problèmes de la programmation linéaire. Grâce à ce dernier développement, le domaine des méthodes des points intérieurs, des fonctions potentielles et des transformations projectives ont connu une véritable révolution. Nous rappelons que sa complexité est $O(n^{3.5}L)$ avec L est le nombre de bits pour traiter les données.

b) Méthodes affines [23, 42]

L'idée de ces méthodes developée par Dikin en 1967 [44], mais elle est restée inconnue jusqu'à 1985-1986 où plusieurs chercheurs l'ont développée. Cet algorithme est basé sur l'algorithme de Karmarkar sauf qu'il n'a pas besoin ni de fonction potentielle ni de transformation projective. La démonstration de sa polynomialité est assez difficile sauf pour le cas primal dual où Monteiro en 1990 [45], a démontré que sa complexité est polynomiale de $(O(nL^2))$ avec L est le nombre de bits pour traiter les données.

c) Méthodes de réduction du potentiel [23, 41, 42]

Ces méthodes ont été développées dans les années 60, pour résoudre les problèmes de la programmation linéaire et non linéaire. Pour la résolution des programmes linéaires en ce temps, ces méthodes n'ont pas reçu beaucoup d'attention à cause de la dominance quasi totale de la méthode du simplexe à cette époque. Grâce aux résultats de l'algorithme de Karmarkar en 1984, le développement à tourné favorablement vers ces méthodes. La complexité des

algorithmes correspondants à ces méthodes est polynomiale.

d) Méthodes de trajectoire centrale [23, 41, 42]

Les méthodes de trajectoire centrale ont été introduites à la même époque que les méthodes de réduction du potentiel, puis ils ont été bien développées au début des années 90. On peut citer une variante pour ces méthodes, ce sont les méthodes barrières logarithmique de type primal dual de trajectoire centrale pour résoudre les problèmes de la programmation linéaire. Théoriquement et pratiquement, ces méthodes montrent un excellent comportement où leur complexité est polynomiale avec une convergence super-linéaire.

1.6.5 Problème de la programmation quadratique convexe

On considère le problème quadratique suivant :

$$(PQ)\begin{cases} \frac{1}{2}\sum_{i=1}^{n}\sum_{k=1}^{n} q_{ik}x_i x_k + \sum_{i=1}^{n} c_i x_i \to \min \\ \sum_{i=1}^{n} a_{ij}x_i = b_j \quad (j = 1, \ldots, m) \\ x_i \geq 0 \quad (i = 1, \ldots, n) \end{cases}$$

On peut écrire ce problème sous la forme suivante :

$$(PQ)\begin{cases} \frac{1}{2}x'Qx + C'x \to \min \\ Ax = b \\ \forall x \in \mathbb{R}^n_+ \end{cases}$$

où Q est une matrice symétrique d'ordre n, A est une matrice $(m \times n)$ supposée de rang plein ($rang(A) = m \leq n$), $C \in \mathbb{R}^n$, $b \in \mathbb{R}^m$ et $x \in \mathbb{R}^n$.

- Le problème (PQ) est convexe si et seulement si la matrice Q est semi-définie positive. Ce problème est noté (PQC)
- Si le problème (PQ) est non convexe (i.e Q n'est pas semi-définie positive) alors il devient un problème d'optimisation globale.
- On peut remarquer qu'un problème linéaire (PL) est considéré comme un problème quadratique avec $Q = 0$.

— Pour le problème quadratique sans contraintes (i.e., le domaine est $\mathbb{R}^n$), on a les propriétés suivantes :

1. Si la matrice Q n'est pas semi-définie positive alors le problème quadratique sans contraintes n'admet pas de solution.
2. Si la matrice Q est définie positive alors le problème quadratique sans contraintes admet une solution unique.

— L'ensemble de solutions réalisables du problème (PQC) est un polyèdre convexe fermé.

On peut définir le problème dual associe au problème original primal (PQ), comme suit :

$$(PQD)\begin{cases} L(z,y,v) = \dfrac{1}{2}z'Qz + C'z + y'(Az-b) - v'z \to \max \\ \dfrac{\partial L}{\partial z}(z,y,v) = Qz + C + A'y - v = 0 \\ z \in \mathbb{R}^n, y \in \mathbb{R}^m, v \in \mathbb{R}^n, v \geq 0. \end{cases}$$

où $L(z,y,v)$ est la fonction de lagrange associe au problème primal (PQ).

Remarque 1.6.1. *Si le problème primal (PQ) n'est pas convexe, alors le problème dual (PQD) peut ne pas avoir de solution à partir de laquelle la solution du primal peut être déduite.*

Pour les deux théorèmes suivants : on note par $f(x) = \dfrac{1}{2}x'Qx + C'x$, la fonction objectif du problème primal (PQ).

Théorème 1.6.4 (**Théorème de la dualité faible, Wolfe 1961 [46]**). *Soient $f(x)$ et $L(x,y,v)$ deux fonctions objectifs du problèmes primal et dual respectivement. Alors*

$$L(x,y,v) \leq f(x) \quad \forall x, v \in \mathbb{R}^n, \forall y \in \mathbb{R}^m, v \in \mathbb{R}^n, v \geq 0.$$

Théorème 1.6.5 (**Théorème de la dualité forte, Wolfe 1961 [46]**). *On note par $x^* \in \mathbb{R}^n$ la solution du problème primal (PQ). s'il existe $y^* \in \mathbb{R}^m$ et $v^* \in \mathbb{R}^n$ avec $v \geq 0$ tel*

que (x^*, y^*, v^*) *est une solution du problème dual* (PQD)*. Alors*

$$f(x^*) = L(x^*, y^*, v^*).$$

1.6.6 Méthodes de résolution d'un programme quadratique convexe

Dans cette partie, nous citons quelques méthodes pour la résolution des programmes quadratiques convexes sans contraintes et avec contraintes [20, 21, 23].

a) Programmes quadratiques convexes sans contraintes

— Méthodes du gradient.

— Méthodes des directions conjuguées.

— Méthode de Newton.

— Méthodes quasi-newtoniennes.

b) Programmes quadratiques convexes avec contraintes

— Méthode du gradient projeté.

— Méthode de Newton par résolution des équations de KKT.

— Méthode de Wilson.

— Méthode d'Uzawa et de Arrow-Hurwiz.

Remarque 1.6.2. *Les définitions et descriptions de ces méthodes avec les conditions nécessaires et suffisantes d'optimalité pour les deux cas (sans contraintes et avec contraintes), seront traités dans la partie de la programmation convexe (cas général).*

1.7 Programmation non linéaire

Dans cette section, on va étudier le problème général de la programmation non linéaire [20, 22, 23, 40, 47]. Les problèmes qu'on va considérer, consistent à optimiser une fonction

non linéaires avec ou sans contraintes non linéaires ou/et linéaires.

1.7.1 Programmation non linéaire sans contraintes

On considère le problème non linéaire sans contraintes suivant :

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

où $f : \mathbb{R}^n \mapsto \mathbb{R}$, est une fonction non linéaire.

Les définitions 1.5.1, 1.5.2 et 1.5.3 décrivent les conditions pour que la fonction f définie sur l'ensemble E, admette un minimum local, minimum local au sens strict et un minimum global.

Définition 1.7.1. *Soit $x \in E \subseteq \mathbb{R}^n$. On dit que x est un point intérieur à E s'il existe un voisinage de x inclu dans l'ensemble E. Autrement dit,*

$$\exists \epsilon > 0, \text{ tel que } B(x, \epsilon) \subset E$$

Définition 1.7.2. *Soit un vecteur $d \in \mathbb{R}^n$ avec $d \neq 0$. Le vecteur d est la direction admissible en un point $x \in E$, s'il existe $\epsilon > 0$ tel que pour tout $\alpha \in [0, \epsilon[$, le point $(x + \alpha d)$ appartient à l'ensemble E.*

Remarque 1.7.1. *Si x est un point intérieur, alors toute direction au point x est une direction admissible.*

Définition 1.7.3. *Soit la fonction $f : E \subseteq \mathbb{R}^n \mapsto \mathbb{R}$. Le vecteur d de direction admissible en point $x \in E$, est dite direction de descente au point x, s'il existe $\epsilon > 0$ tel que pour tout $\alpha \in [0, \epsilon[$, on a $f(x + \alpha d) < f(x)$.*

Remarque 1.7.2. *Si la fonction f est de classe C^1 sur E. Alors, le vecteur d est la direction admissible de descente au point $x \in E$, si et seulement si $d' \nabla f(x) < 0$.*

Lemme 1.7.1 ([47]). *Soit une fonction $f : E \subseteq \mathbb{R}^n \mapsto \mathbb{R}$. Un point x^* est un minimum local de f, s'il n'existe aucune direction admissible de descente au point x^*. Autrement dit, pour toute direction admissible d au point x^*, on a : $d'\nabla f(x^*) \geq 0$.*

Démonstration. Raisonnement par l'absurde où l'hypothèse c'est que le point x^* est un minimum local. On suppose qu'il existe au point x^*, un vecteur d qui est une direction admissible de descente, c'est à dire $d'\nabla f(x^*) < 0$, alors il existe $\overline{x} \in B(x^*, \alpha)$ où $\alpha \in [0, \epsilon[$ tel que $\overline{x} = x^* + \alpha d$. De plus

$$f(\overline{x}) - f(x^*) = f(x^* + \alpha d) - f(x^*) = \alpha d'\nabla f(x^*) + o(\alpha).$$

On sait que $\alpha \to 0 \quad (o(\alpha) \to 0)$ car $B(x^*, \alpha)$ est le voisinage de x^*. Donc

$$f(\overline{x}) - f(x^*) < 0$$

ce qui donne une contradiction avec l'hypothèse, ce qui implique que x^* est un minimum local. □

Théorème 1.7.1 ([47]). *Soit la fonction $f : E \subseteq \mathbb{R}^n \mapsto \mathbb{R}$, différentiable. Si le point x^* est un minimum local de la fonction f, alors*

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

Le point x^* est dit stationnaire.

Démonstration. Soit le point x^* un minimum local de la fonction f. on considère un vecteur $d \in \mathbb{R}^n$ avec $d \neq 0$. Puisque x^* est un minimum local de f, alors il existe $\epsilon > 0$, tel que :

$$\forall \alpha \in [-\epsilon, \epsilon], \quad f(x^*) \leq f(x^* + \alpha d).$$

On pose une fonction $g : \mathbb{R} \mapsto \mathbb{R}$, tel que $g(\alpha) = f(x^* + \alpha d)$. Le point 0 est un minimum local de la fonction g, car x^* est un minimum local de f. La fonction f est différentiable en x^* donc la fonction g est derivable en 0 et $g'(0) =< \nabla f(x^*), d >$.
Alors $g'(0) = 0$ et donc $< \nabla f(x^*), d >= 0$ pour tout $d \in \mathbb{R}^n$ et finalement $\nabla f(x^*) = 0$. □

Théorème 1.7.2 ([47]). *Soit la fonction $f : E \subseteq \mathbb{R}^n \mapsto \mathbb{R}$, deux fois différentiable. Soit un point x^* tel que : $\nabla f(x^*) = 0$ et $\nabla^2 f(x^*) > 0$ (la matrice hessienne de f est définie positive). Alors le point $x^* \in \mathbb{R}^n$ est un minimum local au sens strict de la fonction f.*

Démonstration. On considère le développement de taylor au voisinage de point $x^* \in \mathbb{R}^n$,

$$f(x) = f(x^*) + \nabla' f(x^*)(x - x^*) + \frac{1}{2}(x - x^*)' \nabla^2 f(x^*)(x - x^*) + \|(x - x^*)\|^2 g(x - x^*),$$

avec $g(x - x^*) \to 0$ quand $x \to x^*$. On pose un vecteur de direction admissible $d \in \mathbb{R}^n$, ainsi $x = x^* + \alpha d$, où $\alpha \in]0, \epsilon[$. Supposons que x^* vérifie la condition $\nabla f(x^*) = 0$, alors $f(x) = f(x^* + \alpha d) =$

$$f(x^*) + \frac{1}{2}(x^* + \alpha d - x^*)' \nabla^2 f(x^*)(x^* + \alpha d - x^*) + \|(x^* + \alpha d - x^*)\|^2 g(x^* + \alpha d - x^*),$$

de plus, la matrice hessienne $\nabla^2 f(x^*) > 0$ (définie positive) et $g(x - x^*) \to 0$ quand $x \to x^*$ ($g(\alpha d) \to 0$ quand $\alpha \to 0$), alors

$$f(x^* + \alpha d) - f(x^*) = \frac{1}{2}(x^* + \alpha d - x^*)' \nabla^2 f(x^*)(x^* + \alpha d - x^*)$$

$$f(x^* + \alpha d) - f(x^*) = \frac{1}{2}\alpha^2 d' \nabla^2 f(x^*) d > 0 \Rightarrow f(x^* + \alpha d) > f(x^*), \quad \forall \alpha \in]0, \epsilon[.$$

Ce qui implique que le point x^* est un minimum local au sens strict de la fonction f. □

Méthodes de résolution des problèmes non linéaires sans contraintes [20, 40, 33, 48, 38]

— Méthodes du gradient.

— Méthodes des directions conjuguées.

— Méthodes de Newton.

— Méthodes quasi-newtoniennes.

Description générale de l'algorithme des méthodes de descente

Les méthodes de descente sont basées sur le vecteur de direction de descente qu'on peut noter $d_k \in \mathbb{R}^n$, et un pas optimal qu'on peut noter $\alpha_k > 0$. Alors, l'algorithme général des méthodes de descente est donné comme suit.

Algorithme 1 Algorithme général des méthodes de descente

1. **Initialisation :** $x_0 \in E \subseteq \mathbb{R}^n, d_0 \in \mathbb{R}^n, \alpha_0 > 0$.
2. **Tant que** Test d'arrêt
   (a) Calcul de la direction de descente d_k.
   (b) Calcul du pas optimal α_k.
   (c) Calcul du nouveau point $x_{k+1} = x_k + \alpha_k d_k$.
3. **Fin tant que**
4. La solution optimale est $(x_k, f(x_k))$.

1.7.2 Programmation non linéaire avec contraintes

On considère le problème non linéaire avec contraintes suivant :

$$(PNL) \begin{cases} \min f(x) \\ g_i(x) = 0, \quad i = 1, \ldots, I_M \\ g_i(x) \leq 0, \quad i = I_{M+1}, \ldots, I_K \\ x \in \mathbb{R}^n \end{cases}$$

où $f : \mathbb{R}^n \to \mathbb{R}$, est la fonction objectif, les fonctions $g_i : \mathbb{R}^n \to \mathbb{R}$ pour $i = 1, \ldots, I_M$, sont les contraintes d'égalités et les fonctions $g_i : \mathbb{R}^n \to \mathbb{R}$ pour $i = I_{M+1}, \ldots, I_K$, sont les

contraintes d'inégalités. Au moins une de ces fonctions est non linéaire.

Définition 1.7.4. *Une contrainte d'inégalité $g_i(x)$ pour $i = I_{M+1} \ldots, I_K$, est dite :*

- *Active en x si $g_i(x) = 0$.*
- *Inactive en x si $g_i(x) < 0$.*
- *violée en x si $g_i(x) > 0$.*

Soit $x \in \mathbb{R}^n$. L'ensemble des indices des contraintes actives est donné comme suit :

$$A(x) = \{1, \ldots, I_M\} \cup I_a(x),$$

où $I_a(x) = \{i : g_i(x) = 0, i \in \{I_{M+1}, \ldots, I_K\}\}$, est l'ensemble des indices des contraintes d'inégalités actives au point x.

Définition 1.7.5. *On définit l'ensemble des solutions admissibles $S_A D$, comme suit :*

$$S_A D = \{x \in \mathbb{R}^n : g_i(x) = 0, i \in \{1, \ldots, I_M\}, \qquad g_i(x) \leq 0, i \in \{I_{M+1}, \ldots, I_K\}\}.$$

Définition 1.7.6. *Soit $x \in S_A D$. L'ensemble des directions admissibles au point x est donné comme suit :*

$$D_A(x, S_A D) = \{d \in \mathbb{R}^n : (x + \alpha d) \in S_A D, \forall \alpha \in [0, \epsilon], \epsilon > 0, d \neq 0\}.$$

Définition 1.7.7. *Soit $x \in S_A D$. L'ensemble des directions admissibles linéaires au point $x \in$, est défini comme suit :*

$$D_A L(x) = \left\{d \in \mathbb{R}^n : d' \nabla g_i(x) = 0, i = 1, \ldots, I_M, \qquad d' \nabla g_i(x) \leq 0, i = I_{M+1}, \ldots, I_K\right\}.$$

Définition 1.7.8. *Soit $x \in S_A D$. L'ensemble des directions admissibles séquentielles au point x, est défini comme suit :*

$$D_A S(x) = \left\{ \begin{array}{c} d \in \mathbb{R}^n : \quad x + \alpha_k d_k \in S_A D \\ d_k \to d \ pour \ \alpha_k > 0, \quad \alpha_k \to 0, \quad k = 1, 2, \ldots \end{array} \right\}.$$

Lemme 1.7.2 ([20, 21, 23]). *Soit $x \in S_A D$. Si toutes les contraintes $g_i(x)$ pour $i = 1, \ldots, I_K$ sont différentiables, alors*

$$D_A(x, S_A D) \subseteq D_A S(x) \subseteq D_A L(x).$$

Lemme 1.7.3 (Fiacco et McCormick [49]). *Soit $x \in S_A D$. On dit que le point x satisfait les conditions de qualification des contraintes (QC), si toutes les fonctions gradients $\nabla g_i(x)$ sont linéairement indépendantes, où $i \in A(x)$.*

Définition 1.7.9. *Soit $x \in S_A D$. Un vecteur $d \in \mathbb{R}^n$, est dite direction admissible de descente au point x, si :*

$$d' \nabla f(x) < 0,$$

avec $D_A D = \left\{ d \in \mathbb{R}^n : d' \nabla f(x) < 0 \right\}$ est l'ensemble des directions admissibles de descente au point x.

Théorème 1.7.3 ([21]). *Soit $x^* \in S_A D$, est un minimum local du problème (PNL). Si toutes les fonctions $f(x)$ et $g_i(x)$ pour $i = 1, \ldots, I_K$ sont différentiables, alors :*

$$d' \nabla f(x^*) \geq 0, \quad \forall d \in D_A S(x^*).$$

Dans la suite, nous énonçons les conditions nécessaires et suffisantes du théorème de karush, Kuhn et Tucker (KKT) d'optimalité.

Nous considérons la fonction de Lagrange suivante :

$$L(x, \lambda) = f(x) + \sum_{i=1}^{I_K} \lambda_i g_i(x),$$

où λ est dit vecteur des multiplicateurs de Lagrange.

Théorème 1.7.4 (Condition nécessaire d'optimialité du première ordre (KKT) [47]). *Si $x^* \in S_A D$, est un minimum local du problème (PNL), alors il existe un vecteur des multiplicateurs de lagrange λ^*, tel que :*

$$\left\{\begin{array}{cc} \nabla f(x^*) + \sum_{i=1}^{I_K} \lambda_i^* \nabla g_i(x^*) = 0 & \\ g_i(x^*) = 0 & i = 1, \ldots, I_M \\ g_i(x^*) \leq 0 & i = I_{M+1}, \ldots, I_K \\ \lambda_i^* \geq 0 & i = I_{M+1}, \ldots, I_K \\ \lambda_i^* g_i(x^*) = 0 & i = I_{M+1}, \ldots, I_K \end{array}\right\}$$

Théorème 1.7.5 (Condition nécessaire d'optimalité du deuxième ordre [47]). *Soit $x^* \in S_A D$, un minimum local du problème (PNL). Soit un vecteur des multiplicateurs de lagrange λ^*, tel que (x^*, λ^*) satisfaisant les conditions de (KKT), alors on a :*

$$\left\{\begin{array}{cc} d' \nabla^2 L(x^*, \lambda^*) d \geq 0 & \forall d \neq 0 \\ d' \nabla g_i(x^*) = 0 & i = 1, \ldots, I_M \\ d' \nabla g_i(x^*) = 0 & i = I_{M+1}, \ldots, I_K \end{array}\right\}$$

Théorème 1.7.6 (Condition suffisante d'optimalité de deuxième ordre [47]). *Si un point $x^* \in S_A D$, satisfait :*

$$\left\{\begin{array}{cc} \nabla f(x^*) + \sum_{i=1}^{I_K} \lambda_i^* \nabla g_i(x^*) = 0 & \\ g_i(x^*) = 0 & i = 1, \ldots, I_M \\ g_i(x^*) \leq 0 & i = I_{M+1}, \ldots, I_K \\ \lambda_i^* \geq 0 & i = I_{M+1}, \ldots, I_K \\ \lambda_i^* g_i(x^*) = 0 & i = I_{M+1}, \ldots, I_K \\ \lambda_i > 0 & \forall i \in A(x) \\ d' \nabla^2 L(x^*, \lambda^*) d > 0 & \forall d \neq 0 \\ d' \nabla g_i(x^*) = 0 & i = 1, \ldots, I_M \\ d' \nabla g_i(x^*) = 0, g_i(x^*) = 0 & i = I_{M+1}, \ldots, I_K \end{array}\right\}$$

alors, le point x^* est un minimiseur local au sens strict du problème (PNL).

On considère le problème non linéaire suivant :

$$(PNL) \left\{\begin{array}{cc} \min f(x) & \\ g_i(x) = 0, & i = 1, \ldots, I_M \\ g_i(x) \leq 0, & i = I_{M+1}, \ldots, I_K \\ x \in \mathbb{R}^n & \end{array}\right.$$

où les fonctions $f(x)$ et $g_i(x)$ pour $i = 1, \ldots, I_K$, sont différentiables et (au moins une des fonction est non linéaire).

Théorème 1.7.7 (Fritz John [23]). *Si le point $x^* \in \mathbb{R}^n$ est solution du problème (PNL), alors il existe $\lambda_0 \in \mathbb{R}_+$, $\lambda \in \mathbb{R}_+^{I_K - I_M}$ et $\mu \in \mathbb{R}^{I_M}$, tel que :*

$$(PNL) \left\{\begin{array}{c} \sum_{i=I_{M+1}}^{I_K} \lambda_i g_i(x^*) = 0 \\ \lambda_0 \nabla f(x^*) + \sum_{i=I_{M+1}}^{I_K} \lambda_i \nabla g_i(x^*) + \sum_{j=1}^{I_M} \mu_j \nabla g_j(x^*) = 0 \\ (\lambda_0, \lambda, \mu) \neq 0 \end{array}\right.$$

1.7.3 Méthodes de résolution d'un programme non linéaires avec contraintes

Nous rappelons par la suite, quelques méthodes pour la résolution des problèmes non linéaires (PNL) [20, 21, 22, 23, 50, 51, 52].

— Méthode SQP - algorithmes newtoniens.

— Méthode de Newton par résolution des équations de KKT.

— Méthode des points intérieurs.

— Méthode du lagrangien augmenté.

Méthode SQP - algorithmes newtoniens

La base des algorithmes newtoniens, est la linéarisation des équations caractérisant la solution que l'on cherche, fournies par les conditions d'optimalité de première ordre. L'utilité de ces algorithmes dits primaux-duaux, est de générer à la fois une suite primale $\{x_k\}_k$ convergeant vers la solution optimale x^* du problème (PNL), et une suite duale $\{\lambda_k\}_k$ de multiplicateurs convergeant vers un multiplicateur optimal λ^* associe à x^*.

Concernant notre problème (PNL), on commence par le lagrangien suivant :

$$L(x, \lambda, \mu) = f(x) + \lambda' g(x) + \mu' h(x),$$

avec $g(x) = (g_{I_{M+1}}(x), \ldots, g_{I_K}(x))'$, $h(x) = (g_1(x), \ldots, g_{I_M}(x))'$ et pour la simplicité on note $I_M = M$ et $I_K - I_M = K$, alors les vecteurs des multiplicateurs de lagrage : $\lambda \in \mathbb{R}_+^K$, $\mu \in \mathbb{R}^M$. D'après la condition nécessaire d'optimalité de première ordre (KKT), on a :

$$\nabla L(x, \lambda, \mu) = 0 \Leftrightarrow \begin{pmatrix} \nabla f(x) + \lambda' \nabla g(x) + \mu' \nabla h(x) \\ g(x) \\ h(x) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}.$$

Pour résoudre ce système d'équations non linéaires, nous utilisons la méthode de Newton

dont l'itération est comme suit :

$$\nabla^2 L(x_k, \lambda_k, \mu_k) \begin{pmatrix} x_{k+1} - x_k \\ \lambda_{k+1} - \lambda_k \\ \mu_{k+1} - \mu_k \end{pmatrix} = -\nabla L(x_k, \lambda_k, \mu_k)$$

Posons : $d_x = x_{k+1} - x_k$, $d_\lambda = \lambda_{k+1}$ et $d_\mu = \mu_{k+1}$. Alors

$$\begin{pmatrix} \nabla^2_{xx} L(x_k, \lambda_k, \mu_k) & \nabla' g(x_k) & \nabla' h(x_k) \\ \nabla' g(x_k) & 0 & 0 \\ \nabla' h(x_k) & 0 & 0 \end{pmatrix} \begin{pmatrix} d_x \\ d_\lambda - \lambda_k \\ d_\mu - \mu_k \end{pmatrix} = \begin{pmatrix} -\nabla_x L(x_k, \lambda_k, \mu_k) \\ -g(x_k) \\ -h(x_k) \end{pmatrix}.$$

Cette itération est bien définie à condition que la matrice $\nabla^2_{xx} L(x_k, \lambda_k, \mu_k)$ soit inversible. Ce sera le cas si :

1. toutes les fonctions gradients des contraintes sont linéairement indépendantes (la condition de qualification des contraintes).
2. $d' \nabla g_i(x_k) = 0$, $d' \nabla h_j(x_k) = 0$ et $\forall d \neq 0$ te que $d' \nabla^2 L(x_k, \lambda_k, \mu_k) d > 0$ (la condition suffisante d'optimalité de deuxième ordre).

Revenons au système suivant :

$$\begin{cases} \nabla^2_{xx} L(x_k, \lambda_k, \mu_k) d_x + \nabla' g(x_k)(d_\lambda - \lambda_k) + \nabla' h(x_k)(d_\mu - \mu_k) &= -\nabla_x L(x_k, \lambda_k, \mu_k) \\ \nabla' g(x_k) d_x &= -g(x_k) \\ \nabla' h(x_k) d_x &= -h(x_k) \end{cases}$$

avec $\nabla_x L(x_k, \lambda_k, \mu_k) = \nabla f(x_k) + \lambda_k \nabla' g(x_k) + \mu_k \nabla' h(x_k)$, alors

$$\begin{cases} \nabla^2_{xx} L(x_k, \lambda_k, \mu_k) d_x + \nabla' g(x_k) d_\lambda + \nabla' h(x_k) d_\mu &= -\nabla f(x_k) \\ \nabla' g(x_k) d_x + g(x_k) &= 0 \\ \nabla' h(x_k) d_x + h(x_k) &= 0 \end{cases}$$

Le système précédent est connu comme conditions d'optimalité de première ordre du problème quadratique suivant :

$$(PQ_k)\begin{cases}\min_{d_x} d'_x\nabla^2_{xx}L(x_k,\lambda_k,\mu_k)d_x+\nabla' f(x_k)d_x\\ \nabla' g(x_k)d_x+g(x_k)\leq 0\\ \nabla' h(x_k)d_x+h(x_k)=0\end{cases}$$

Les itérations de Newton pour résoudre les conditions de KKT, sont équivalentes à résoudre une suite de problèmes quadratiques (PQ_k). Alors, la méthode SQP consiste à remplacer le problème initial par une suite de problèmes quadratiques avec contraintes linéaires plus facile à résoudre. L'algorithme SQP est donné comme suit :

Algorithme 2 Algorithme SQP

1. **Initialisation :** $x_0 \in E \subseteq \mathbb{R}^n$ un point initial, $\lambda_0 \in \mathbb{R}^K_+$, $\mu_0 \in \mathbb{R}^M$, $\epsilon > 0$.
2. k=0.
3. **Tant que** $\|\nabla L(x_k,\lambda_k)\| > \epsilon$
   (a) Résoudre le sous-problème quadratique :
   $$(PQ_k)\begin{cases}\min_{d_x} d'_x\nabla^2_{xx}L(x_k,\lambda_k,\mu_k)d_x+\nabla' f(x_k)d_x\\ \nabla' g(x_k)d_x+g(x_k)\leq 0\\ \nabla' h(x_k)d_x+h(x_k)=0\end{cases}$$
   et obtenir la solution primale d_k et les multiplicateurs d_λ et d_μ associés aux contraintes d'inégalité et d'égalité respectivement.
   (b) Calculer les nouveaux vecteurs $x_{k+1} = x_k + d_k$, $\lambda_{k+1} = d_\lambda$, $\mu_{k+1} = d_\mu$ et $k = k+1$.
4. **Fin tant que**
5. Retourner x_k.

1.7.4 Programmation convexe

Dans cette partie, on va traiter le problème général d'optimisation convexe [20, 21, 23, 47, 48, 53]. Les problèmes qu'on va considérer, consistent à optimiser une fonction convexe sous contraintes convexes.

On considère le problème convexe suivant :

$$(PNLC)\begin{cases} \min f(x) \\ Ax = b \\ g_i(x) \leq 0, \qquad i = 1, \ldots, K \\ x \in \mathbb{R}^n \end{cases}$$

où $f : \mathbb{R}^n \mapsto \mathbb{R}$ est une fonction convexe et les fonctions g_i sont convexes sur $\mathbb{R}^n$ $(i = 1, \ldots, K)$, alors le domaine des solutions admissibles est convexe.

Théorème 1.7.8 ([36])**.** *Soit x^* un minimum local du problème $(PNLC)$, alors x^* est un minimum global du $(PNLC)$.*

Démonstration. On note par E_c le domaine des solutions admissible du problème $(PNLC)$. Soit $x^* \in E_c$ un minimum local du problème $(PNLC)$, donc il existe un voisinage $V \subseteq E_c$ de x^*, tel que : $\forall x \in V$ on a : $f(x^*) \leq f(x)$. Soit un point $x_0 \in E_c$ tel que $f(x_0) < f(x^*)$. On considère un point $x \in E_c$ tel que $x = \lambda x^* + (1 - \lambda)x_0$ avec $\lambda \in [0, 1]$. Alors par la convexité de $(PNLC)$ on a

$$f(x) \leq \lambda f(x^*) + (1 - \lambda)f(x_0) < \lambda f(x^*) + (1 - \lambda)f(x^*) = f(x^*)$$

$$\Rightarrow f(x) < f(x^*)$$

On peut prendre λ suffisamment proche de 1 de tel sort que $x \in V(x^*)$, donc $f(x) \geq f(x^*)$. Ce qui donne une contradiction avec l'inégalité $f(x) < f(x^*)$. Donc x^* est un minimum global. □

Lemme 1.7.4 ([36]). *L'ensemble des minimums globaux du problème $(PNLC)$, est convexe.*

Démonstration. Soit x_1^*, x_2^* deux minimums globaux du problème $(PNLC)$, donc $f(x_1^*) = f(x_2^*)$.
Soit $x = \lambda x_1^* + (1-\lambda)x_2^*$, alors

$$f(x) = f(\lambda x_1^* + (1-\lambda)x_2^*) \leq \lambda f(x_1^*) + (1-\lambda) f(x_2^*)$$

$$= \lambda f(x_2^*) + (1-\lambda) f(x_2^*) = f(x_2^*).$$

D'autre part, puisque x_1^* et x_2^* sont des minimums globaux, on sait que $f(x) \geq f(x_1^*) = f(x_2^*)$. Alors

$$f(x) \leq f(x_2^*) \text{ et } f(x) \geq f(x_2^*) \Rightarrow f(x) = f(x_1^*) = f(x_2^*)$$

et donc l'ensemble des minimums globaux est convexe. □

Lemme 1.7.5 ([36]). *Soit x^* un minimum local du problème $(PNLC)$. Si la fonction $f(x)$ est strictement convexe, alors x^* est un minimum global unique.*

Démonstration. Soit $f(x)$ une fonction strictement convexe. Supposons que $x1^*$ et $x2^*$ deux minimums globaux du problème $(PNLC)$, tel que $x1^* \neq x2^*$ avec $f(x1^*) = f(x2^*)$.
Soit $x = \lambda x1^* + (1-\lambda)x2^*$ avec $\lambda \in]0,1[$. Alors par la convexité strict de la fonction $f(x)$, on a

$$f(x) < \lambda f(x1^*) + (1-\lambda) f(x2^*) = f(x2^*) \Rightarrow f(x) < f(x2^*)$$

Mais cela contredit le fait que $x2^*$ est un minimum global. Alors le minimum global est unique. □

Théorème 1.7.9 ([36]). *Soit un point x^* satisfaisant les conditions KKT pour le problème $(PNLC)$, alors x^* est un minimum global du problème $(PNLC)$.*

Démonstration. Soit (x^*, λ^*) est un point KKT. On sait que $\lambda_i^* \geq 0$ pour $i = 1, \ldots, K$, et

on peut avoir un point x tel que $g_i(x) \leq 0$ pour $i = 1, \ldots, K$, alors

$$L(x, \lambda^*) = f(x) + \sum_{i=1}^{K} \lambda_i^* g_i(x) \leq f(x),$$

Donc il est normal d'avoir l'inégalité suivante :

$$L(x, \lambda^*) \leq f(x).$$

De plus, on a

$$L(x, \lambda^*) \geq L(x^*, \lambda^*) + (x - x^*)\nabla_x L(x, \lambda) \geq L(x^*, \lambda^*) \geq f(x^*).$$

Alors

$$f(x) \geq f(x^*).$$

□

1.7.5 Dualité en programmation convexe

L'objectif de la dualité est de trouver une transformation du problème primal vers le problème dual. La théorie de la dualité [20, 21, 23] est souvent associée à la notion de multiplicateurs de Lagrange. En optimisation on peut trouver des méthodes de résolution primales, des méthodes duales ou encore des méthodes primales duales.

Nous considérons le problème convexe primal suivant :

$$(PNLC) \begin{cases} \min_{x \in \mathbb{R}^n} f(x) \\ g_i(x) \leq 0, \qquad i = 1, \ldots, K \end{cases}$$

Définition 1.7.10. *Le problème dual du problème convexe $(PNLC)$ de wolfe [46], est donné comme suit :*

$$(PNLCD)\begin{cases}\max_{x\in\mathbb{R}^n,\lambda\in\mathbb{R}^K} L(x,\lambda) = \max_{x\in\mathbb{R}^n,\lambda\in\mathbb{R}^K} f(x) + \lambda' g(x)\\ \nabla_x L(x,\lambda) = 0, \quad \lambda_i \geq 0, \quad i = 1,\ldots,K.\end{cases}$$

avec $g(x) = (g_1(x),\ldots,g_K(x))$.

Théorème 1.7.10 ([23]). *Soit* (x^*,λ^*) *est un point maximum du problème dual* $(PNLCD)$*, et si la matrice hessienne de la fonction Lagrange est régulière au point* x^* *et si les contraintes du problème dual* $(PNLCD)$ *sont qualifiées en* (x^*,λ^*)*, alors le point* x^* *est un minimum global du problème primal* $(PNLC)$*.*

Démonstration. Donnons la fonction de Lagrange associe au problème dual $(PNLCD)$, comme suit :

$$LD((x,\lambda),\beta) = L(x,\lambda) + \beta'\nabla_x L(x,\lambda).$$

Soit (x^*,λ^*) un maximum du problème dual $(PNLCD)$. Soit $\beta^* \in \mathbb{R}^n$ un vecteur des multiplicateurs de Lagrange associe au problème dual $(PNLCD)$. Alors on a :

$$\nabla_\lambda LD((x^*,\lambda^*),\beta^*) = g(x^*) + \beta^*\nabla g(x^*) \leq 0.$$

$$\nabla_x LD((x^*,\lambda^*),\beta^*) = \nabla_x L(x^*,\lambda^*) + \beta^*\nabla^2_{xx} L(x^*,\lambda^*) = 0.$$

On sait que $\nabla_x L(x^*,\lambda^*) = 0$ ((x^*,λ^*) est un point admissible du $(PNLCD)$), et la matrice hessienne de la fonction Lagrange est régulière, alors

$$\beta^*\nabla^2_{xx} L(x^*,\lambda^*) = 0 \Rightarrow \beta^* = 0.$$

Donc,

$$g(x^*) + \beta^*\nabla g(x^*) \leq 0 \Rightarrow g(x^*) \leq 0.$$

Ce qui donne x^* est un minimum global du problème primal $(PNLC)$. □

Théorème 1.7.11 (**Théorème de la dualité faible [54]**). *Soit x^* un minimum global du problème primal $(PNLC)$. Si les fonction f et g_i pour $i = 1, \ldots, K$, sont différentiables, alors (x^*, λ^*) est une solution du problème dual $(PNLCD)$, avec*

$$f(x^*) = L(x^*, \lambda^*).$$

Théorème 1.7.12 (**Théorème de la dualité forte [54]**). *Soit une solution réalisable du problème primal $(PNLC)$ notée $\overline{x}$, et une solution réalisable du problème dual $(PNLCD)$, notée $(x, \overline{\lambda})$, alors on a*

$$f(\overline{x}) \geq L(x, \overline{\lambda}).$$

Maintenant, on introduit l'application du point selle sur la fonction de Lagrange pour un problème non linéaire (PNL).
Soit la fonction de Lagrange

$$L(x, \lambda) = f(x) + \sum_{i=1}^{K} \lambda_i g_i(x).$$

Définition 1.7.11. *On dit que le couple (x^*, λ^*) avec $x^* \in \mathbb{R}^n$ et $\lambda^* \in \mathbb{R}^K$, est un point selle du lagrangien $L(x, \lambda)$, si on a*

$$L(x^*, \lambda) \leq L(x^*, \lambda^*) \leq L(x, \lambda^*).$$

Théorème 1.7.13 ([55]). *Si (x^*, λ^*) est un point selle de la fonction $L(x, \lambda)$, alors x^* est un minimum local du problème (PNL). En plus (x^*, λ^*) satisfait les conditions de KKT.*

1.8 Conclusion

Dans ce chapitre, les aspects fondamentaux concernant l'optimisation convexe ont été présentés. Ainsi, l'étude théorique et algorithmique avec les méthodes de résolution des problèmes d'optimisation convexe ont été traités.

CHAPITRE 2

OPTIMISATION NON CONVEXE ET OPTIMISATION GLOBALE

2.1 Introduction

Au cours des dernières années, le domaine de la recherche sur l'optimisation globale s'est considérablement enrichi grâce à l'accroissement de puissance de calcul des machines ou des ordinateurs. Ces progrès nous ont permis de résoudre des problèmes auparavant assez difficiles ou insolubles. Dans ce chapitre, nous nous intéressons à l'étude théorique et algorithmique des problèmes d'optimisation globale [9]. Ces problèmes consistent à optimiser des fonctions objectifs non convexes sur un ensemble des contraintes qui sont aussi non convexes. La différence fondamentale distinguant l'optimisation globale de l'optimisation locale, c'est que la solution du problème d'optimisation globale est le minimum ou maximum absolu sur l'ensemble du domaine réalisable.

2.2 Méthodes déterministes de résolution en optimisation globale

Dans cette section, nous allons présenter quelques méthodes déterministes pour la résolution des problèmes d'optimisation globale [1, 2, 3, 4, 5, 6, 7, 8, 9, 56, 57, 58].

2.2.1 Programmation anti-convexe [1, 2, 3, 4, 5, 6]

Soit le problème d'optimisation globale anti-convexe suivant :

$$\begin{cases} \min f(x) \\ g(x) \geq 0 \\ x \in E \subseteq \mathbb{R}^n \end{cases}$$

où les fonctions f et g sont convexes sur l'ensemble convexe fermé E dans $\mathbb{R}^n$.

Théorème 2.2.1 ([4]). *Une solution optimale du problème de la programmation anti-convexe, se trouve dans l'intersection de l'ensemble E avec la frontière de l'ensemble G, où*

$$G = \{x \in \mathbb{R}^n : g(x) \geq 0\}.$$

2.2.2 Méthode de Séparation et d'évaluation progressive [1, 4, 5, 7, 8]

La méthode de Séparation et Evaluation (Branch and Bound) est une méthode itérative pour résoudre une grande classe des problèmes d'optimisation globale. Cette méthode est basée sur la subdivision du domaine des solutions réalisables en plusieurs sous domaines de plus en plus petits. A chaque itération et pour chaque sous domaine, on construit une borne inférieure ou/et une borne supérieure de la fonction objectif afin d'éliminer les sous domaines ne contenant pas la solution optimale globale, puis on sélectionne sous certaines conditions, le nouveau sous domaine pour la prochaine itération. Cette méthode utilise les outils d'analyse convexe.

Principe de la méthode Branch and Bound

On considère le problème d'optimisation globale suivant :

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

Où $f : \mathbb{R}^n \mapsto \mathbb{R}$, est une fonction non convexe, et l'ensemble E est le domaine des solutions réalisable qui est convexe ou non convexe.

L'algorithme de Branch and Bound consiste à engendrer deux suites $\{UB_k\}$ et $\{LB_k\}$ des bornes supérieures et des bornes inférieures respectivement de la valeur minimale de la fonction objectif f du problème (PG). On notera E_{ki} les sous-ensembles de la subdivision de l'ensemble E_k à l'itération k, alors les bornes inférieure et supérieure à l'itération k, sont données comme suit :

$$\begin{cases} LB_k = \min_i LB_{ki} \\ UB_k = \min_i UB_{ki} \end{cases}$$

Tout sous-ensemble sur lequel la borne inférieure dépasse UB_k sera éliminé car la solution globale du problème (PG), ne peut être atteint sur cet ensemble. On peut représenter cette méthode par une arborescence qui a pour racine l'ensemble E, et pour sommets les sous-ensembles E_k qui s'obtiennent par une subdivision successive. Voir figure 2.1.

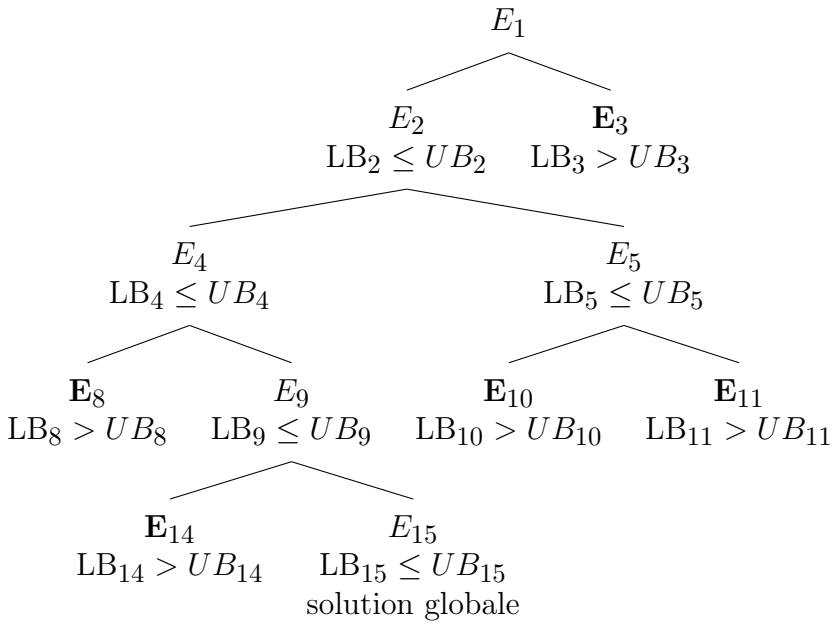

FIGURE 2.1 – Exemple d'une arborescence de la subdivision des sous-ensembles.

Remarque 2.2.1. *On peut subdiviser tout ensemble E_k en plusieurs sous-ensembles et non pas forcement en deux sous-ensembles.*

Algorithme général de la méthode Branch and Bound

Pour bien clarifier le déroulement des étapes annoncées ci-dessus, on peut les résumer sur l'algorithme suivant :

Algorithme 3 Algorithme Branch and Bound

1. **Initialisation :** Poser $E_1 = E$ et construire l'ensemble $I_1 = \{E_1\}$, poser $k = 1$ et fixer $\epsilon > 0$.
2. Construire les problèmes des bornes inférieure et supérieure de la fonction objectif f sur E_k, ensuite calculer les bornes LB_k et UB_k.
3. **Tant que** $UB_k - LB_k \geq \epsilon$ et $I_k \neq \emptyset$
   (a) subdiviser l'ensemble E_k en deux sous-ensembles E_{k1} et E_{k2}, tel que

$$\bigcup_{i=1}^{2} E_{ki} = E_k \text{ et } \bigcap_{i=1}^{2} \widehat{E_{ki}} = \emptyset,$$

   où $\widehat{E_{ki}}$ est l'intérieur de l'ensemble E_{ki}.
   (b) Construire les problèmes des bornes inférieure et supérieure de la fonction objectif f sur l'ensemble E_{ki}, et calculer les bornes LB_{ki} et UB_{ki}, pour $i = 1, 2$.
   (c) Poser :

$$UB_{k+1} = \min\{UB_{k1}, UB_{k2}, UB_k\}$$

   (d) Poser $I_{k+1} = I_k \cup \{E_{ki} : UB_{k+1} - LB_{ki} \geq \epsilon\} \setminus \{E_k\}$.
   (e) Éliminer de I_{k+1} tout ensemble E_k tel que : $UB_{k+1} - LB_k < \epsilon$.
   (f) Poser $k = k + 1$ et si $I_k \neq \emptyset$, choisir de I_k un sous-ensemble E_k et calculer LB_k.
4. **Fin tant que**
5. Retourner $\min f(x) = UB_k$ et x^* tel que $f(x^*) = UB_k$.

2.2.3 Méthode des plans coupants

On considère le problème anti-convexe linéaire suivant :

$$(PCI) \begin{cases} \min c'x \\ Ax \leq b \\ g(x) \geq 0 \\ x \geq 0 \end{cases}$$

où A est une matrice de type $(m \times n)$, $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$ et $g : \mathbb{R}^n \mapsto \mathbb{R}$ est une fonction convexe. Soit $P = \left\{x \in \mathbb{R}^n_+ : Ax \leq b\right\}$ un espace polyédrique. $G = \{x \in \mathbb{R}^n : g(x) \geq 0\}$. On note par $D = P \cap G$ avec $D \neq \emptyset$, l'ensemble des solutions réalisables.

Principe général et l'algorithme de la méthode des plans coupants

Soit x^0 la solution optimale non dégénérée du problème linéaire (PL) associe au problème (PCI) :

$$(PL)\begin{cases}\min c'x\\ x \in P.\end{cases}$$

Supposons que $N(x^0)$ est l'ensemble des sommets voisins du point x^0 où

$$N(x^0) = \left\{x^1, x^2, \ldots, x^n\right\}.$$

On choisit de l'ensemble $N(x^0)$ un sommet x^j et on définit $z^j = x^j - x^0$, la direction de x^0 vers x^j.

On suppose que les sommets voisins de x^0 sont admissibles pour le problème (PCI). En démarrant du point x^0, l'objectif est de trouver le plus proche intersection de la surface de $g(x)$ avec la frontière du polyèdre P. Pour cela, le point à rechercher doit vérifier l'hypothèse $g(x^0) < 0$ et $g(x^j) \geq 0, x^j \in N(x^0)$. Cette intersection notée $y^j = x^0 + \alpha z^j$, est trouvée à partir de la solution du sous problème suivant :

$$(SP)\begin{cases}\min \alpha\\ g(x^0 + \alpha z^j) = 0\\ 0 \leq \alpha \leq 1.\end{cases}$$

$y^j = x^0 + \alpha^j(x^j - x^0)$, où α^j est la solution du sous problème (SP). Alors on a :

$$Ay^j \leq b,$$

$$g(y^j) = g(x^0 + \alpha^j(x^j - x^0)) = 0$$

$$c'y^j = c'(x^0 + \alpha^j(x^j - x^0))$$

$$= (1 - \alpha^j)c'x^0 + \alpha^j c'x^j \leq c'x^j.$$

Donc, y^j est un point admissible pour le problème (PCI) avec une meilleure valeur de la fonction objectif que $c'x^j$. On note par $x^* = y^j$ la solution optimale actuelle. Maintenant, s'il existe un autre point $x^k \in N(x^0)$ tel que $c'x^k < c'x^*$, alors le sous problème (SP) sera résolu et x^* est mise à jour en conséquence. Le processus est continue jusqu'à $N(x^0)$ est épuisé $(N(x^0) = \emptyset)$.

2.2.4 Méthode d'approximation extérieure

L'idée de cette méthode, introduite dans la programmation convexe à la fin des années cinquante par Cheney et Goldstein [59], ensuite par Hoffman [60] et Tuy [61], a été développée sous le nom d'approximation extérieure, pour la minimisation d'une fonction concave sous des contraintes convexe, puis plus généralement pour les problèmes d'optimisation non convexe par Mayane et Polak [62] et Tuy [6]. Le grand principe de la méthode d'approximation extérieure est de faire des coupes linéaires sur un ensemble qui contient l'ensemble réalisable du problème original, de façon qu'on ne coupe jamais l'ensemble réalisable tout en se rapprochant de plus en plus de la solution optimale du problème original.

Principe de la méthode d'approximation extérieure

On considère le problème d'optimisation non convexe suivant :

$$(P)\begin{cases} \min f(x) \\ h_i(x) \leq 0 \quad i = 1, \ldots, m. \\ g(x) \geq 0. \end{cases}$$

où f, h_i et g sont des fonction convexes de $\mathbb{R}^n \mapsto \mathbb{R}$. Posons

$h(x) = \max_{i=1,\ldots,m} h_i(x)$,

$$E = \{x \in \mathbb{R}^n : h_i(x) \leq 0\}, \quad G = \{x \in \mathbb{R}^n : g(x) \geq 0\}.$$

Le principe de cette méthode est de relaxer et remplacer le problème original (P) par une suite de problèmes (Q_k) qui sont facile à résoudre et dont les solutions obtenues convergent

vers la solution optimale du problème original (P). Pour la réalisation d'un algorithme d'approximation extérieure, on a les deux phases suivantes :

1. Construire des sous problèmes (Q_k) qui sont faciles à résoudre.
2. Construire $l_k(x)$ pour que la suite des solutions $\overline{x}_k$ converge.

Algorithme de la méthode d'approximation extérieure

Avant de citer l'algorithme de cette méthode, on note que :

— w est la solution optimale du problème d'optimisation non convexe (P), sans la contrainte anti-convexe.

— ∂G représente la frontière de l'ensemble G.

Algorithme 4 Algorithme d'approximation extérieure

1. **Initialisation :** Soit $\overline{x}_1 \in E \cap \partial G$, poser $y_1 = f(\overline{x}_1)$, former un polytope P_1 où le compact convexe $\{x \in E : f(x) \leq y_1\}$ est contenu dans P_1, fixer $k = 1$.
2. **Faire**
   (a) Résoudre le sous problème (Q_k) tel que :

$$(Q_k) : \max(g(x), x \in P_k).$$

   La solution est noté z_k.
   (b) **Si** $g(z_k) = 0$, alors **Stop** (Solution optimale).
   (c) **Sinon** chercher x_k dans le segment $[w, z_k]$ tel que $\max\{f(x_k) - y_k, g(x_k)\} = 0$.
   (d) **Si** $x_k \in E$ alors $v_k \in \partial f(x_k)$

$$l_k(x) =< v_k, (x - x_k) > .$$

   (e) **Si** $x_k \notin E$ alors $v_k \in \partial h(x_k)$

$$l_k(x) =< v_k, (x - x_k) > +h(x_k)$$

   (f)

$$P_{k+1} = P_k \cap \{x \in \mathbb{R}^n : l_k(x) \leq 0\}$$

$$\overline{x}_{k+1} = \begin{cases} x_k & \text{si } x_k \in E \text{ et } g(x_k) = 0 \\ \overline{x}_k & \text{Sinon.} \end{cases}$$

   (g) $y_{k+1} = f(\overline{x}_{k+1})$.
3. **Jusqu'au** $(g(z_k) = 0)$

2.2.5 Méthode D.C : différence de fonctions convexes

Les fonctions DC possèdent de nombreuses propriétés qui ont été établies à partir des années 50 par Landis [63] et Hartman [64]. La classe des fonctions DC a été introduite en optimisation au milieu des années 80. Distinguons deux grandes approches DC :

1. L'approche combinatoire en optimisation globale continue.
2. L'approche de l'analyse convexe en optimisation non convexe.

Cette méthode a été proposée par Pham Dinh et Le thi Hoai an [65], et utilise les techniques d'optimisation globale (Méthode Branch and Bound, technique de coupe ...)

pour la résolution des problèmes d'optimisation globale.

La programmation D.C

Définition 2.2.1. *Une fonction $f : \mathbb{R}^n \mapsto \mathbb{R}$ est dite DC si elle peut s'écrire comme suit :*

$$f(x) = g(x) - h(x),$$

avec g et h des fonctions convexes de $\mathbb{R}^n$ dans $\mathbb{R}$.

La programmation DC consiste à résoudre des programmes mathématiques dont la fonction objectif est une fonction DC. Le problème de la programmation DC est donnée comme suit :

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

où $f : \mathbb{R}^n \mapsto \mathbb{R}$ est une fonction DC.

Définition 2.2.2. *Soit une fonction $f : \mathbb{R}^n \mapsto \mathbb{R}$. La fonction f est dite semi continue inférieure au point x_0, si'il existe un voisinage V de x_0 tel que :*

$$\forall \epsilon > 0, \forall x \in V : \quad f(x) \geq f(x_0) - \epsilon.$$

La fonction f est dite propre si

$$\forall x_0 \in \mathbb{R}^n : \quad \lim_{x \to x_0} f(x) \neq -\infty \text{ et } \lim_{x \to x_0} f(x) \neq +\infty.$$

On note par Γ_0 l'ensemble des fonction convexes, semi continues inférieure et propres dans $\mathbb{R}^n$.

Définition 2.2.3. *Soit une fonction $f : \mathbb{R}^n \mapsto \mathbb{R}$, convexe et propre. On dit que v_0 est le*

sous gradient de la fonction f au point x_0, si :

$$\forall x \in \mathbb{R}^n : \quad < v_0, x - x_0 > + f(x_0) \leq f(x).$$

On note par $\partial f(x_0)$, l'ensemble des sous gradients de la fonction f au point x_0, appelé sous différentiel.

On dit que v_0 est le ϵ-sous gradient de la fonction f au point x_0, si

$$\forall x \in \mathbb{R}^n : \quad < v_0, x - x_0 > + f(x_0) \leq f(x) + \epsilon.$$

On note par $\partial_\epsilon f(x_0)$, l'ensemble des ϵ-sous gradients de la fonction f au point x_0, appelé ϵ-sous différentiel.

Définition 2.2.4. *Soit une fonction $f : \mathbb{R}^n \mapsto \mathbb{R}$. On note par f^* la fonction conjuguée de la fonction f, et définie comme suit :*

$$f^*(y) = sup \{< x, y > - f(x), x \in \mathbb{R}^n\}, y \in \mathbb{R}^n.$$

La fonction conjuguée f est l'enveloppe supérieure des fonctions affines continues $< x, y > - f(x)$ sur $\mathbb{R}^n$.

Propriété 2.2.1. *Si $f \in \Gamma_0$, alors*

- *$f \in \Gamma_0 \Longleftrightarrow f^* \in \Gamma_0$, et $f = f^*$.*
- *$y \in \partial f(x) \Longleftrightarrow f(x) + f^*(y) =< x, y >$ et $y \in \partial f(x) \Longleftrightarrow x \in \partial f^*(y)$.*
- *$\partial f(x)$ est un ensemble convexe fermé.*
- *Si $\partial f(x) = \{y\}$ alors f est differenstiable en x et $\nabla f(x) = y$.*

Théorème 2.2.2 (Condition nécessaire d'optimalité local [66]). *Soit x^* un minimum local de la fonction $f(x) = g(x) - h(x)$, alors*

$$\partial h(x^*) \subset \partial g(x^*).$$

Théorème 2.2.3 (Condition suffisante d'optimalité local [67, 68]). *Si un point x^* admet un voisinage V tel que :*

$$\partial g(x^*) \cap \partial h(x^*) \neq \emptyset, \quad \forall x \in V \cap dom(g).$$

Alors x^* est un minimum local de la fonction $f = g - h$.

Corollaire 2.2.1. *Si $x^* \in int(dom(h))$ avec*

$$\partial h(x^*) \in int(\partial g(x^*)),$$

alors x^* est un minimum local de la fonction $f = g - h$.

Théorème 2.2.4 (Optimalité globale [69]). *Soit une fonction $f = g - h$ avec $g, h \in \Gamma_0$. un point x^* est un minimum global de la fonction $f(x)$ sur $\mathbb{R}^n$, si et seulement si*

$$\partial_\epsilon h(x^*) \subset \partial_\epsilon g(x^*), \quad \forall \epsilon > 0.$$

Dualité en optimisation DC

En 1978, la dualité DC introduite par Toland [70] considérée comme généralisation des travaux de Pham Dinh [71] en (1975), concernant la maximisation convexe. On vas présenter quelques résultats principaux sur la dualité DC. Pour plus de détails Voir [68, 70].

Soient deux fonctions $g(x)$ et $h(x)$ convexes dont $g, h \in \Gamma_0$. Le problème primal (P) et son dual (D) sont donnés comme suit :

$$(P)\begin{cases}\inf f(x) = g(x) - h(x)\\ x \in \mathbb{R}^n.\end{cases} \qquad (D)\begin{cases}\inf -f^*(y) = h^*(y) - g^*(y)\\ y \in \mathbb{R}^n.\end{cases}$$

où f^* est la fonction conjuguée de la fonction f.

Théorème 2.2.5 ([70]). *Soient g et h deux fonctions tel que $g, h \in \Gamma_0$, Alors*

1. $\inf_{x\in\mathbb{R}^n} g(x) - h(x) = \inf_{y\in\mathbb{R}^n} h^*(y) - g^*(y)$.
2. *Si y_0 est un minimum de la fonction $h^* - g^*$ sur $\mathbb{R}^n$, alors tout $x_0 \in \partial g(y_0)$ est un minimum de $g - h$ sur $\mathbb{R}^n$.*
3. *Si x_0 est un minimum de la fonction $g - h$ sur $\mathbb{R}^n$, alors tout $y_0 \in \partial g(x_0)$ est un minimum de $h^* - g^*$ sur $\mathbb{R}^n$.*

Parfois, pour résoudre un problème d'optimisation DC, il est plus facile de résoudre le problème dual (D) que le problème primal (P). Le théorème précèdent montre que la résolution du problème primal (P) implique la résolution du problème dual (D) et vice-versa.

Théorème 2.2.6 ([67, 68]). *Soit $x_0 \in dom(\partial h)$ un minimum local de la fonction $f = g - h$. Soient $y_0 \in \partial h(x_0)$ et V un voisinage de x_0 tel que :*

$$g(x) - h(x) \geq g(x_0) - h(x_0), \quad \forall x \in V \cap dom(g).$$

Si

$$x_0 \in int(dom(g^*)) \text{ et } \partial g^*(y_0) \subset V,$$

alors y_0 est un minimum local de la fonction $-f^* = h^* - g^*$.

Algorithme DCA [72]

La construction de l'algorithme DCA proposé par Pham Dinh Tao en (1986), est basé sur la caractérisation des solutions locales des problèmes primal (P) et dual (D), en optimisation DC.

Algorithme 5 Algorithme DCA

1. **Initialisation :** Soit x_0 une solution initiale. $k = 0$ et $\epsilon > 0$.
2. Calculer $y \in \partial h(x_k)$, pour x_k connu.
3. Calculer $x_{k+1} \in \partial g^*(y_k)$.
4. **Si** $\|x_{k+1} - x_k\| < \epsilon$, **STOP**.
5. **Sinon** $k = k + 1$.

L'algorithme DCA consiste à construire deux suites $\{x_k\}$ et $\{y_k\}$, où la première est la solution du problème primal (P) et la deuxième est la solution du problème dual (D). Les deux suites sont liées par dualité et vérifient les propriétés suivantes :

— Les deux suites $\{g(x_k) - h(x_k)\}$ et $\{h^*(y_k) - g^*(y_k)\}$ sont décroissantes.

— L'algorithme DCA s'arrête à l'itération $(k+1)^{ieme}$, lorsque $(g-h)(x_{k+1}) = (g-h)(x_k)$, et le point x_k est un point critique de $g - h$, de même pour le point y_k est un point critique de $h^* - g^*$.

2.2.6 Méthode par analyse d'intervalles [9]

Les techniques d'analyse d'intervalles offrent de nouvelles méthodes de résolution des problèmes d'optimisation globale. Grâce à l'information globale fournie par ces techniques, l'algorithme basé sur ces techniques, assure que l'optimum déterminé et un optimum global et non pas piégé par un optimum local.

Principe de la méthode d'analyse d'intervalles

Soit le problème d'optimisation globale suivant :

$$(P) \begin{cases} \min f(x) \\ h_j(x) = 0, \quad j = 1, \ldots, M. \\ g_i(x) \leq 0, \quad i = 1, \ldots, K. \\ x \in \mathbf{X} \subset \mathbb{R}^n. \end{cases}$$

où les fonctions f, h_j et g_j peuvent être non convexes. $\mathbf{X}$ est un sous domaine convexe de $\mathbb{R}^n$. Le principe de la résolution de problème d'optimisation globale (P) par l'analyse d'intervalles

est basé sur l'utilisation d'un algorithme branch and bound par intervalles pour la recherche de l'optimum global.

Définition 2.2.5. *Soit une fonction* $f : \boldsymbol{X} \subset \mathbb{R}^n \mapsto \mathbb{R}$. *L'image directe de* f *sur* $\boldsymbol{X}$ *est l'intervalle noté* $f(\boldsymbol{X})$, *de largeur minimale tel que*

$$\forall x \in \boldsymbol{X}, f(x) \in f(\boldsymbol{X}).$$

Définition 2.2.6. *Une fonction* $F : \boldsymbol{X} \subset \mathbb{R}^n \mapsto \mathbb{R}$ *est dite fonction d'inclusion de* f *si et seulement si*

$$\forall x \in \boldsymbol{Y} \subseteq \boldsymbol{X}, f(x) \in F(\boldsymbol{Y}).$$

La fonction d'inclusion de f *est une fonction retournant un minorant et un majorant de la fonction* f *sur un intervalle.*

Définition 2.2.7. *Soit une fonction* $f : \mathbb{R}^n \mapsto \mathbb{R}$. *Une extension naturelle aux intervalles d'une fonction consiste à réécrire la fonction* f *en remplaçant toutes les occurrences de la variable par l'intervalle correspondant et en remplaçant les opérateurs classiques par les opérateurs équivalents en arithmétique d'intervalle. L'extension naturelle est notée* $EN_f(\boldsymbol{X})$.

Propriété 2.2.2. *L'extension naturelle aux intervalles de* f *est une fonction d'inclusion.*

Théorème 2.2.7 ([9])**.** *Soit une fonction* $f : \boldsymbol{X} \subset \mathbb{R}^n \mapsto \mathbb{R}$. *Si l'extension naturelle de* f *possède seulement une seule occurrence de chaque variable, alors l'image directe de la fonction* f *sur* $\boldsymbol{X}$ *est exactement l'extension naturelle* $EN_f(\boldsymbol{X})$. *Autrement dit*

$$\forall \boldsymbol{Y} \subseteq \boldsymbol{X}, f(\boldsymbol{Y}) = EN_f(\boldsymbol{Y}).$$

Algorithme Branch and Bound par intervalles : IBBA [9]

Les algorithmes de Branch and Bound par intervalles, ont été développé dans les années 70. Le développement de ces algorithmes était très théorique au début et était plutôt avare

de résultats numériques. La structure de l'algorithme qu'on va présenter est tiré du livre de Ratschek et Rokne [73]. Cet algorithme permet de résoudre le problème d'optimisation globale (P) à condition que toutes les fonctions du problème (P) doivent avoir une expression explicite pour pouvoir utiliser les arithmétiques d'intervalles.

Algorithme 6 Algorithme Branch and Bound par intervalles : IBBA

1. **Initialisation :** Poser $\mathbf{Y} = \mathbf{X}$, $y = F^L(\mathbf{X})$, $z = F^U(\mathbf{X})$ et l'ensemble $L = \{(\mathbf{Y}, y)\}$, $\epsilon > 0$
2. **Repeter**
   (a) Extraire de l'ensemble L un élément $(\mathbf{Y}, y)$.
   (b) Diviser $\mathbf{Y}$ en $\mathbf{Y}_1$ et $\mathbf{Y}_2$ tel que $\mathbf{Y} = \mathbf{Y}_1 \cup \mathbf{Y}_2$.
   (c) **pour** $j = 1, 2$ **faire**
      i. Calculer $F(\mathbf{Y}_j)$ et poser $y_j = F^L(\mathbf{Y}_j)$.
      ii. **Si** $y_j \leq z - \epsilon$ et toutes les contraintes sont satisfaites sur $\mathbf{Y}_j$, alors insérer le couple $(\mathbf{Y}_j, y_j)$ dans L.
      iii. Poser $z = \min\left\{z, F^U(\mathbf{Y}_j)\right\}$
      iv. **Si** z a changé alors supprimer de L tous les éléments $(\mathbf{Y}, y)$ tel que $z - \epsilon < y$.
      v. **Finsi ; Finsi**
   (d) **Fin pour**
3. **Jusqu'à** $\left(z - \min_{(\mathbf{Y},y)\in L} y \leq \epsilon\right)$

2.3 Méthodes de résolution basées sur l'algorithme Branch and Bound

2.3.1 Méthode αBB [10, 11, 12]

La méthode αBB est développée par Androulakis, Maranas, Floudas et d'autres [10, 11, 12, 29]. Cette méthode est basée sur l'algorithme de branch and bound pour la résolution des problèmes d'optimisation globale avec contraintes. L'importance centrale de cette méthode est le calcul du paramètre α qui garantit la convexité de la fonction borne inférieure utilisée

par l'algorithme αBB durant la résolution du problème d'optimisation globale. Le problème d'optimisation globale considéré est le suivant :

$$(P)\begin{cases} \min f(x) \\ h_j(x) = 0, \quad j = 1, \ldots, M. \\ g_k(x) \leq 0, \quad k = 1, \ldots, K. \\ Ax \leq b. \\ \underline{x} \leq x \leq \overline{x}. \end{cases}$$

où f, h et g sont des fonctions non linéaires et au moins une de ces fonctions est non convexe. A est une matrice carrée de type $(n \times n)$, $b \in \mathbb{R}^n$ et $\underline{x}, \overline{x} \in \mathbb{R}^n$ avec $\underline{x} < \overline{x}$.

La fonction borne inférieure

La fonction borne inférieure (sous estimateur) d'une fonction non convexe $f : \mathbb{R}^n \mapsto \mathbb{R}$, est donnée comme suit :

$$LB_\alpha(x) = f(x) - \frac{1}{2}\sum_{i=1}^{n} \alpha_i (x_i - \underline{x}_i)(\overline{x}_i - x_i), \tag{2.1}$$

$$\text{avec} \quad \alpha_i \geq \max\left\{0, -\min_{\underline{x} \leq x \leq \overline{x}} \lambda_i(x)\right\},$$

avec $\lambda_i(x)$ les valeurs propres de la matrice hessienne de f.

Théorème 2.3.1 ([10]). *La fonction $LB_\alpha(x)$ coïncide avec la fonction $f(x)$ aux extrémités du domaine $[\underline{x}, \overline{x}]$.*

Démonstration. Soit x^0 un sommet du domaine $[\underline{x}, \overline{x}]$, alors pour chaque $i = 1, \ldots, n$ on a $(x_i^0 - \underline{x}_i)(\overline{x}_i - x_i^0) = 0$, ce qui implique que $LB_\alpha(x^0) = f(x^0)$. □

Théorème 2.3.2 ([10]). *$LB_\alpha(x) \leq f(x)$ pour tout point x sur le domaine $[\underline{x}, \overline{x}]$.*

Démonstration. Par construction de la fonction $LB_\alpha(x)$, on a :

$$\frac{1}{2}\sum_{i=1}^{n} \alpha_i (x_i - \underline{x}_i)(\overline{x}_i - x_i) \geq 0,$$

alors, $LB_\alpha(x) \leq f(x), \quad \forall x \in [\underline{x}, \overline{x}]$. □

Théorème 2.3.3 ([10]). *La fonction $LB_\alpha(x)$ est convexe sur le domaine $[\underline{x}, \overline{x}]$.*

Démonstration. C'est une conséquence directe de la définition des paramètres α_i. □

Théorème 2.3.4 ([10]). *La distance maximale de séparation entre la fonction $f(x)$ et la fonction borne inférieure $LB_\alpha(x)$ sur le domaine $[\underline{x}, \overline{x}]$, est donnée comme suit :*

$$\max_{\underline{x} \leq x \leq \overline{x}} f(x) - LB_\alpha(x) = \frac{1}{4} \sum_{i=1}^{n} \alpha_i (\overline{x}_i - \underline{x}_i)^2. \tag{2.2}$$

Démonstration.

$$\max_{\underline{x} \leq x \leq \overline{x}} f(x) - LB_\alpha(x) = \max_{\underline{x} \leq x \leq \overline{x}} f(x) - f(x) + \frac{1}{2} \sum_{i=1}^{n} \alpha_i (x_i - \underline{x}_i)(\overline{x}_i - x_i)$$

$$= \max_{\underline{x} \leq x \leq \overline{x}} \frac{1}{2} \sum_{i=1}^{n} \alpha_i (x_i - \underline{x}_i)(\overline{x}_i - x_i) = \frac{1}{4} \sum_{i=1}^{n} \alpha_i (\overline{x}_i - \underline{x}_i)^2.$$

□

La fonction borne inférieure améliorée

Une fonction borne inférieure améliorée [74, 75] utilisée dans la méthode αBB, est donnée comme suit :

$$LB_\gamma(x) = f(x) - \sum_{i=1}^{n} \left(1 - e^{\gamma_i (x_i - \underline{x}_i)}\right) \left(1 - e^{\gamma_i (\overline{x}_i - x_i)}\right), \tag{2.3}$$

où $\gamma = (\gamma_1, \ldots, \gamma_n)$ est un vecteur des valeurs non négatives.

Théorème 2.3.5 ([74]). *(i) La fonction $LB_\gamma(x)$ coïncide avec la fonction $f(x)$ aux extrémités du domaine $[\underline{x}, \overline{x}]$.*
(ii) $LB_\gamma(x) \leq f(x)$ pour tout x sur le domaine $[\underline{x}, \overline{x}]$.

Théorème 2.3.6 ([74]). *Il existe une valeur $\gamma \in [\underline{\gamma}, \overline{\gamma}]$ tel que $LB_\gamma(x)$ est convexe et $LB_\gamma(x) \geq LB_\alpha(x)$, sur le domaine $[\underline{x}, \overline{x}]$.*

Principe général de la méthode αBB

La méthode αBB était proposée pour localiser la solution optimale du problème d'optimisation globale (P). Cette méthode génère une séquence des bornes inférieures et supérieures qui convergent vers la valeur optimale. Les bornes inférieures sont obtenus par les solutions des problème convexes en remplaçant les fonctions non convexes du problème (P) par des fonctions borne inférieure $LB_\alpha(x)$. Les bornes supérieures sont des solutions du problème (P), obtenu par des méthodes de recherche local. Par la suite, l'algorithme αBB est présenté pour détailler le principe de la méthode αBB.

Algorithme de la méthode αBB

Algorithme 7 Algorithme αBB

1. **Initialisation :** Fixer $k = 1$, $\epsilon_c > 0$ et $\epsilon_f > 0$. Initialiser $x^{c,k}$, $UB = +\infty$ et $LB = -\infty$. Poser $[\underline{x}^k, \overline{x}^k] = [\underline{x}, \overline{x}]$. $\Lambda = \left\{[\underline{x}^k, \overline{x}^k]\right\}$.
2. Extraire le premier élément de la liste Λ, noté $[\underline{x}^k, \overline{x}^k]$. Résoudre localement le problème (P) sur $[\underline{x}^k, \overline{x}^k]$, x^k_{up} est la solution local. **Si** x^k_{up} est ϵ_f-réalisable et $f(x^k_{up}) < UB$, alors mettre à jour la borne supérieure $UB : UB = f(x^k_{up})$ et $x^* = x^k_{up}$.
3. Construire le problème borne inférieure (P_L) en remplaçant les fonctions nonconvexes par les fonctions $LB_\alpha(x)$. Résoudre le problème (P_L) sur $[\underline{x}^k, \overline{x}^k]$. Soit x^k_{low} la solution optimale du (P_L). **Si** $LB_\alpha(x^k_{low}) \geq LB$, alors mettre à jour $LB = LB_\alpha(x^k_{low})$, et $x^{c,k+1} = x^k_{low}$.
4. **Si** $(LB_\alpha(x^k_{low}) - UB \geq \epsilon_c)$, alors $[\underline{x}^k, \overline{x}^k]$ peut être supprimé en toute sécurité.
   Sinon, partitionner le sous-domaine courant $[\underline{x}^k, \overline{x}^k]$ en deux sous-domaines, en coupant le plus grand intervalle comme suit :

$$\left([\underline{x}^k_1, \overline{x}^k_1], \ldots, \left[\underline{x}^k_j, \frac{\overline{x}^k_j + \underline{x}^k_j}{2}\right], \ldots, [\underline{x}^k_n, \overline{x}^k_n]\right),$$

   et

$$\left([\underline{x}^k_1, \overline{x}^k_1], \ldots, \left[\frac{\overline{x}^k_j + \underline{x}^k_j}{2}, \overline{x}^k_j,\right], \ldots, [\underline{x}^k_n, \overline{x}^k_n]\right),$$

   où $[\underline{x}^k_j, \overline{x}^k_j]$, est supposé le plus grand intervalle du sous-domaine courant. Ajouter les deux sous-domaines obtenus dernièrement, à la liste Λ, en respectant l'ordre de l'emplacement tel que la liste Λ soit reste en ordre décroissante par rapport au bornes inférieures.
5. **Si** $\Lambda = \emptyset$, alors arrêter. La solution optimale est x^* et sa valeure est $f(x^*)$.
   Sinon, faire $k = k + 1$ et aller à l'étape 2.

2.3.2 Méthode de borne inférieure quadratique [13]

Cette méthode est développée en 2014 par M. Ouanes, H.A. Le Thi, T.P. Nguyen et A. Zidna [13], pour les fonctions multivariables générales en optimisation globale. Le principe de cette méthode est le même que la méthode αBB. Cependant, cette méthode diffère dans la fonction borne inférieure et l'algorithme de branch and bound pour la résolution des problèmes d'optimisation globale. Le problème considéré ici est le suivant :

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

où

$$B=\prod_{i=1}^{n}[x_i^0,x_i^1]$$

est un hyperrectangle dans $\mathbb{R}^n$ et $f:\mathbb{R}^n\mapsto\mathbb{R}$ est une fonction C^2-continue et différentiable.

La fonction borne inférieure

La fonction borne inférieure (sous estimateur) quadratique de la fonction f, est donnée comme suit :

$$LB_q(x)=L_hf(x)-\frac{1}{2}K\sum_{i=1}^{n}(x_i^1-x_i)(x_i-x_i^0). \tag{2.4}$$

où $L_hf(x)$ est un interpolant multi-linéaire de la fonction $f(x)$ dans B. K est un nombre réel positif qui satisfait l'inégalité $K\geq\|H_f(x)\|$ pour tout x dans B, avec $H_f(x)$ la matrice hessienne de f.

Soient $w_0(x_i)$ et $w_1(x_i)$ deux réelles valeurs définies par

$$w_0(x_i)=\frac{x_i^1-x_i}{x_i^1-x_i^0},\qquad\qquad w_1(x_i)=\frac{x_i-x_i^0}{x_i^1-x_i^0}.$$

La construction de la fonction $L_hf(x)$ est donnée comme suit :

$$L_hf(x)=\sum_{i_n=0}^{1}\left(\dots\left(\sum_{i_1=0}^{1}f(x_1^{i_1},\dots,x_n^{i_n})w_{i_1}(x_1)\right)\dots\right)w_{i_n}(x_n).$$

Pour le cas uni-dimensionnelle où $n=1$, la fonction borne inférieure $LB_q(x)$ devient comme suit :

$$LB_q(x)=f(x^0)w_0(x)+f(x^1)w_1(x)-\frac{1}{2}K(x^1-x)(x-x^0),$$

avec $k\geq|f''(x)|$ pour tout x dans $B=[x^0,x^1]$.

Théorème 2.3.7 ([13]). *(i) Pour tout sommet x dans $V(B)$ on a $LB_q(x)=f(x)$.*

(ii) Pour tout x dans B, si $\max_{x\in B} |f''_{x_i x_i}(x)| \leq K$, alors $LB_q(x) \leq f(x)$.

Théorème 2.3.8 ([13]). *La fonction borne inférieure $LB_q(x)$ définie dans* (2.4) *est convexe si*

$$K \geq \max_{x\in B} \max_{i=1,\ldots,n} \sum_{j=1, j\neq i}^{n} |f''_{x_i x_j}(x)|.$$

Pour plus de détails sur la preuve des deux théorèmes précédents voir [13].

Algorithme de Branch and Bound

On note par LB_k, UB_k et s_k respectivement la meilleure borne inférieure, la meilleure borne supérieure de $f(x)$ et la meilleure solution du problème (P) à l'itération k.

Algorithme 8 Algorithme de borne inférieure quadratique

1. **Initialisation :** Fixer $\epsilon > 0$.
2. Calculer K une borne supérieure de $\|H_f\|$ sur B.
3. Mettre $T^0 = B$. **Pour** $i = 1, \ldots, n$, **faire** $h_i = x_i^1 - x_i^0$.
4. Résoudre le problème convexe suivant pour obtenir une solution optimale $\underline{s}_0$

$$\min \left\{ LB_q(x) : x \in T^k \right\}. \tag{2.5}$$

5. Mettre $LB_0 = LB(T^0) := LB_q(\underline{s}_0)$. Mettre $UB_0 = \min \left\{ \min_{v \in V(B)} f(v), f(\underline{s}_0) \right\}$.
6. **Si** $UB_0 - LB_0 \leq \epsilon$ **alors** $\underline{s}_0$ est une solution ϵ-optimale. **STOP**.
   Sinon Mettre $M \leftarrow \left\{ T^0 \right\}$, $k \leftarrow 1$.
7. **Étape de sélection**
   (a) Soit $T^k = \prod_{i=1}^n [a_i, b_i] \in M$ un hyper-rectangle tel que $LB_k = LB(T^k)$, et $\underline{s}_k$ est la solution du problème (2.5).
   (b) Calculer K_k tel que $\|H_f(x)\| \leq K_k, \forall x \in T^k$.
8. Couper T^k en deux sous-rectangles T_1^k et T_2^k par la w-subdivision par $\underline{s}_k$.
9. **Pour** $i = 1, 2$ **faire** Résoudre le problème convexe (2.5) avec k remplacé par k_i pour obtenir une solution optimale $\underline{s}_{ki}$.
10. **Étape de mise à jour**
    (a) Mettre à jour la borne supérieure $UB_k = \min \{UB_{k-1}, f(\underline{s}_{k1}), f(\underline{s}_{k2})\}$.
    (b) Soit s_k la meilleure solution actuel, i.e. $f(s_k) = UB_k$.
    (c) Mettre $M \leftarrow M \cup \left\{ T_i^k : LB(T_i^k) < UB_k - \epsilon, i = 1, 2 \right\} / \left\{ T^k \right\}$.
    (d) Mettre à jour la borne inférieure $LB_k = \min \{LB(T) : T \in M\}$.
11. Si $M = \emptyset$ **alors** s_k est la solution optimale et quitter l'algorithme.
    Sinon Mettre $k \leftarrow k + 1$ et retourner à l'étape 7.

2.4 Problèmes d'optimisation non convexe du monde réel

Nombreux problèmes du monde réel sont modélisés en optimisation globale. Par la suite, nous citerons quelques problèmes réels sélectionnés à partir de différentes applications réelles [76, 77, 78, 79, 80, 81, 82].

2.4.1 Problèmes de procédés chimiques industriels

Plusieurs problèmes de processus chimiques ont été proposés qui sont très compliqués [77, 78, 79].

a) Conception de réseau d'échangeurs de chaleur [77, 83]

La forme optimale de la structure de l'échangeur de chaleur est considérée dans ce problème.

$$\begin{cases} \min \quad 35x_1^{0.6} + 35x_2^{0.6} \\ 200x_1x_4 - x_3 = 0 \\ 200x_2x_6 - x_5 = 0 \\ x_3 - 10000(x_7 - 100) = 0 \\ x_5 - 10000(300 - x_7) = 0 \\ x_3 - 10000(600 - x_8) = 0 \\ x_5 - 10000(900 - x_9) = 0 \\ x_4 ln(x_8 - 100) - x_4 ln(600 - x_7) - x_8 + x_7 + 500 = 0 \\ x_6 ln(x_9 - x_7) - x_6 ln(600) - x_9 + x_7 + 600 = 0 \\ 0 \leq x_1 \leq 10, 0 \leq x_2 \leq 200, 0 \leq x_3 \leq 100, 0 \leq x_4 \leq 200, \\ 1000 \leq x_5 \leq 2000000, 0 \leq x_6 \leq 600, 100 \leq x_7 \leq 600, 100 \leq x_8 \leq 600, \\ 100 \leq x_9 \leq 900, \end{cases}$$

où la variable x_i pour $i = 1, \ldots, 9$, représente le flux de processus. Les contraintes d'égalité définissant les espaces requises pour chaque échangeur et les coûts d'espace. La fonction objectif représente le coût total.

b) Problème de regroupement de Haverly [78, 84]

Ce problème consiste en une fonction objectif linéaire et des contraintes non linéaires et a la forme suivante :

$$\begin{cases} \max \quad 9x_1 + 15x_2 - 6x_3 - 16x_4 - 10(x_5 + x_6) \\ x_7 + x_8 - x_4 - x_3 = 0 \\ x_1 - x_5 - x_7 = 0 \\ x_2 - x_6 - x_8 = 0 \\ x_9x_7 + x_9x_8 - 3x_3 - x_4 = 0 \\ x_9x_7 + 2x_5 - 2.5x_1 \leq 0 \\ x_9x_8 + 2x_6 - 1.5x_2 \leq 0 \\ 0 \leq x_1, x_3, x_4, x_5, x_6, x_8 \leq 100, 0 \leq x_2, x_7, x_9 \leq 200, \end{cases}$$

où les variables représentent les quantités des substances consommées et des substances produites. Les contraintes représentent les capacités des substances. La fonction objectif représente le bénéfice total des substances.

c) Conception de réseau de réacteurs [79, 85]

Un problème de conception de réseau de réacteurs est résolu pour concevoir une séquence de deux réacteurs à cuve agitée en continu. Le but principal de ce problème est d'optimiser la concentration du produit. Ce problème est formulé comme suit

$$\begin{cases} \max \quad x_4 \\ k_1x_5x_2 + x_1 - 1 = 0 \\ k_3x_5x_3 + x_3 + x_1 - 1 = 0 \\ k_2x_6x_2 - x_1 + x_2 = 0 \\ k_4x_6x_4 + x_2 - x_1 + x_4 - x_3 = 0 \\ x_5^{0.5} + x_6^{0.5} \leq 0 \\ 0 \leq x_4, x_3, x_2, x_1 \leq 1, \\ 0.00001 \leq x_6, x_5 \leq 16. \end{cases}$$

où, $k_3 = 0.0391908, k_4 = 0.9k_3, k_1 = 0.09755988$ et $k_2 = 0.99k_1$, sont les constantes de vitesse pour la première et la deuxième réaction dans le premier et le second réacteur respectivement. Les variables $x_1, \ldots, x_4$ représentent les concentrations des produits. x_5 et x_6 représentent

les temps de réaction pour le premier et le second réacteur respectivement. Les contraintes représentent la conservation des concentrations des produites avec le temps des réactions.

2.4.2 Problèmes de conception mécanique

L'optimisation globale a beaucoup d'applications dans le domaine de la mécanique et de l'ingénierie [80, 81, 82].

a) Conception de ressort en tension/compression [80]

L'objectif principal de ce problème est de minimiser le poids d'un ressort de tension ou de compression. Ce problème consiste de quatre contraintes et trois variables sont utilisées pour calculer le poids : le diamètre du fil (x_1), la moyenne du diamètre de la bobine (x_2) et le nombre des bobines actives (x_3). Les contraintes représentent la flexion minimale, la contrainte de cisaillement, la fréquence de surtension et les limites de diamètre extérieur. Ce problème est définit comme suit :

$$\begin{cases} \min \quad x_1^2 x_2(2+x_3) \\ 1 - \dfrac{x_2^3 x_3}{71785 x_1^4} \leq 0 \\ \dfrac{4x_2^2 - x_1 x_2}{12566(x_2 x_1^3 - x_1^4)} + \dfrac{1}{5108 x_1^2} - 1 \leq 0 \\ 1 - \dfrac{140.45 x_1}{x_2^2 x_3} \leq 0 \\ \dfrac{x_1 + x_2}{1.5} - 1 \leq 0 \\ 0.05 \leq x_1 \leq 2.00. \\ 0.25 \leq x_2 \leq 1.30 \\ 2.00 \leq x_3 \leq 15.0. \end{cases}$$

b) Conception de réservoir sous pression [81]

L'objectif principal de ce problème est de minimiser le coût total de soudage, la matière et la forme du réservoir. Ce problème comporte quatre contraintes qui doivent être satisfaites, et quatre variables sont utilisées pour calculer la fonction objectif : épaisseur de la coque (z_1), épaisseur de la tête (z_2), rayon intérieur (x_3) et longueur du réservoir sans inclure la tête (x_4). Les contraintes représentent les limites minimales sur l'épaisseur des parois. Ce problème est donné comme suit :

$$\begin{cases} \min \quad 1.7781 z_2 x_3^2 + 0.622 z_1 x_3 x_4 + 3.1661 z_1^2 x_4 + 19.84 z_1^2 x_3 \\ 0.00954 x_3 \leq z_2 \\ 0.0193 x_3 \leq z_1 \\ x_4 \leq 240 \\ -\pi x_3^2 x_4 - \frac{4}{3}\pi x_3^3 \leq 1296000 \\ 10 \leq x_4, x_3 \leq 200 \\ 1 \leq x_2, x_1 \leq 99 \text{ (variables entières)} \\ z_1 = 0.0625 x_1, \quad z_2 = 0.0625 x_2. \end{cases}$$

c) Problème de conception de treillis à trois barres [80, 82]

Ce problème d'optimisation est pris du génie civil. L'objectif principal de ce problème est de minimiser le poids total des structures de barres. Les variables représentent le stress des barres. Les contraintes de ce problème représentent le stress, la déviation et la déformation des barres. Ce problème consiste à minimiser une fonction objectif linéaire sous trois contraintes non linéaires.

$$\begin{cases} \min \quad l\left(x_2 + 2\sqrt{2x_1}\right) \\ \dfrac{x_2}{2x_2x_1 + \sqrt{2}x_1^2}P - \sigma \leq 0 \\ \dfrac{x_2 + \sqrt{2}x_1}{2x_2x_1 + \sqrt{2}x_1^2}P - \sigma \leq 0 \\ \dfrac{1}{x_1 + \sqrt{2}x_2}P - \sigma \leq 0 \\ 0 \leq x_1, x_2 \leq 1. \\ l = 100, P = 2, \text{ et } \sigma = 2, \end{cases}$$

où l est la distance entre une barre et une barre et une barre. P et σ sont les coefficients de pression sur l'origine.

2.5 Conclusion

Dans ce chapitre nous avons présenté quelques méthodes déterministes de résolution en optimisation globale. Nous avons vu des méthodes basées sur les sous-estimateurs et l'algorithme de branch and bound pour résoudre les problèmes d'optimisation globale. Enfin, nous avons cité quelques problèmes d'optimisation globale du monde réel.

CHAPITRE 3

CONTRIBUTION À L'OPTIMISATION GLOBALE

3.1 Arithmétique d'intervalle

On définit un intervalle $\mathbf{x}$ comme étant une paire ordonnée de nombres réels $[\underline{x}, \overline{x}]$, avec $\underline{x} \leq \overline{x}$. $\underline{x}$ est la borne inférieure de $\mathbf{x}$ et $\overline{x}$ est la borne supérieure de $\mathbf{x}$. Alors,

$$\mathbf{x} = [\underline{x}, \overline{x}] = \{x \in \mathbb{R} | \underline{x} \leq x \leq \overline{x}\}.$$

Maintenant, on note par $\mathbb{I}$ l'ensemble des intervalles.

$$\mathbb{I} = \{\mathbf{x} = [\underline{x}, \overline{x}] | \underline{x}, \overline{x} \in \mathbb{R} \text{ et } \underline{x} \leq \overline{x}\}.$$

Soient $\mathbf{x}$, $\mathbf{y}$ deux intervalles dans l'ensemble $\mathbb{I}$. On va redéfinir dans l'ensemble $\mathbb{I}$ les opérations usuelles de l'arithmétique euclidienne. La plupart des opérations et des relations gardent leur définition habituelle sur l'ensemble $\mathbb{I}$, mais leurs propriétés sont souvent affaiblies. On donne comme exemple, la relation d'ordre $<$ est une relation d'ordre partiel sur l'ensemble $\mathbb{I}$:

$$\mathbf{x} < \mathbf{y} \Leftrightarrow \overline{x} < \underline{y}.$$

L'union et l'intersection de deux intervalles sont donnés comme suit :

$$\mathbf{x} \cup \mathbf{y} = [\min\left(\underline{x}, \underline{y}\right), \max\left(\overline{x}, \overline{y}\right)];$$

$$\mathbf{x} \cap \mathbf{y} = \begin{cases} \emptyset & \text{si } \overline{x} < \underline{y} \text{ ou } \overline{y} < \underline{x}, \\ [\max\left(\underline{x}, \underline{y}\right), \min\left(\overline{x}, \overline{y}\right)] & \text{sinon.} \end{cases}$$

Les opérations sur les ensembles gardent la même définition pour les intervalles.

On définit la largeur de l'intervalle $\mathbf{x}$, par $w(\mathbf{x}) = \overline{x} - \underline{x}$. Soit un vecteur d'intervalles $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n) \in \mathbb{I}^n$, alors la largeur de $\mathbf{x}$ est donnée par : $w(\mathbf{x}) = \max_{i \in \{1,\ldots,n\}} w(\mathbf{x}_i)$.

On note par $mid(\mathbf{x})$, le milieu d'un intervalle $\mathbf{x}$, où $mid(\mathbf{x}) = \dfrac{\overline{x} + \underline{x}}{2}$. Soit un vecteur d'intervalles $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n) \in \mathbb{I}^n$, alors le milieu de $\mathbf{x}$ est donné par :

$$mid(\mathbf{x}) = \left(\frac{\overline{x}_1 + \underline{x}_1}{2}, \frac{\overline{x}_2 + \underline{x}_2}{2}, \ldots, \frac{\overline{x}_n + \underline{x}_n}{2} \right).$$

Les opérations usuelles de l'arithmétique d'intervalles sont définis de la manière que l'intervalle résultant soit le plus petit intervalle contenant tous les images des éléments des intervalles de départ. Les opérations binaires de base sont données comme suit :

$$\begin{cases} \mathbf{x} + \mathbf{y} = & [\underline{x} + \underline{y}, \overline{x} + \overline{y}]; \\ \mathbf{x} - \mathbf{y} = & [\underline{x} - \overline{y}, \overline{x} - \underline{y}]; \\ \mathbf{x} \times \mathbf{y} = & [\min\left\{\underline{x} \times \underline{y}, \underline{x} \times \overline{y}, \overline{x} \times \underline{y}, \overline{x} \times \overline{y}\right\}, \\ & \max\left\{\underline{x} \times \underline{y}, \underline{x} \times \overline{y}, \overline{x} \times \underline{y}, \overline{x} \times \overline{y}\right\}]; \\ \mathbf{x} \div \mathbf{y} = & \mathbf{x} \times [\dfrac{1}{\overline{y}}, \dfrac{1}{\underline{y}}], \qquad \text{Si } 0 \notin \mathbf{y}. \end{cases}$$

Ce principe peut s'étendre pour les fonctions unaires. Quelques exemples d'opérations usuelles sont données comme suit :

$$\begin{aligned} \sqrt{\mathbf{x}} = & \quad [\sqrt{\underline{x}}, \sqrt{\overline{x}}], & \text{si } \underline{x} \geq 0; \\ log(\mathbf{x}) = & \quad [log(\underline{x}), log(\overline{x})], & \text{si } \underline{x} > 0; \\ exp(\mathbf{x}) = & \quad [exp(\underline{x}), exp(\overline{x})]; & \\ |\mathbf{x}| = & \begin{cases} [\underline{x}, \overline{x}], & \text{si } \underline{x} > 0; \\ [|\overline{x}|, |\underline{x}|], & \text{si } \overline{x} < 0; \\ [0, \max(|\underline{x}|, |\overline{x}|)], & \text{si } 0 \in \mathbf{x}. \end{cases} \end{aligned}$$

Plusieurs extensions ont notamment été proposées pour inclure les intervalles infinis, ou pour gérer la division par un intervalle contenant 0, cette arithmétique appelée : **arithmétique d'intervalles étendue** a été proposée par Kahan et Hanson [86, 87] et développéé par Hansen [88].

Notre objectif dans ce chapitre est de présenter le contenu de notre article publié dans YUJOR journal [89].

BOUNDS ON EIGENVALUES OF REAL SYMMETRIC INTERVAL MATRICES FOR αBB METHOD IN GLOBAL OPTIMIZATION

3.2 Introduction

On considère le problème suivant

$$(Pb) \quad \begin{cases} \min f(x) \\ x \in [\underline{x}, \overline{x}] \subset \mathbb{R}^n \end{cases}$$

où $f : \mathbb{R}^n \to \mathbb{R}$ est une fonction non convexe et C^2-continue avec $x \in \mathbb{R}^n$, où $x_i \in \mathbf{x}_i = [\underline{x}_i, \overline{x}_i]$, $\underline{x}_i < \overline{x}_i$, pour $i = 1, \ldots, n$. (Pb) est appelé un problème d'optimisation globale. Plusieurs méthodes ont été étudiées dans la littérature pour résoudre les problèmes d'optimisation globale. L'algorithme de branch and Bound αBB [29, 11, 10, 12] est basé sur la construction d'un sous-estimateur convexe pour les fonctions générales C^2. L'efficacité de l'algorithme αBB dépend de l'efficacité du sous-estimateur. Dans la méthode αBB [10], le sous-estimateur a été défini comme suit :

$$F(x) = f(x) - \frac{1}{2} \sum_{i=1}^{n} \alpha_i (x_i - \underline{x_i})(\overline{x_i} - x_i),$$

avec

$$\alpha_i \geq \max \left\{ 0, - \min_{x \in [\underline{x}, \overline{x}]} \lambda_i(x) \right\},$$

où $\lambda_i(x)$ est la $i$ème valeur propre de la matrice hessienne de $f(x)$. Le choix de α_i est d'assurer la convexité du sous-estimateur. Si on suppose que la propriété de convexité est garantie, alors la sélection d'une plus petite valeur α_i conduit à un sous-estimateur plus efficace. Puisque on travail sur une matrice hessienne d'intervalles symétrique de $f(x)$, il est difficile de calculer les bornes exactes sur les valeurs propres des matrices d'intervalles symétriques

générales, en fait, il est montré dans [90, 91, 92] qu'un tel problème est, en général, NP-difficile. Diverses méthodes dans [11, 93, 94, 95, 96, 97, 98, 99] ont étudié le calcul des valeurs propres des matrices d'intervalles. Aussi, les méthodes dans [100, 101, 102] ont étudié la stabilité des systèmes dynamiques d'intervalles, les matrices d'intervalles paramétriques générales et d'autres résultats liés avec les bornes des valeurs propres. Pour la méthode αBB, la méthode la plus favorable pour calculer les bornes des valeurs propres des matrices d'intervalles est la méthode de Gerschgorin [11]. C'est une méthode facile pour calculer chaque paramètre α_i. Sinon, les méthodes dans [96, 95, 97, 99] sont moins faciles pour calculer les bornes sur les valeurs propres des matrices d'intervalles, mais elles peuvent produire des bornes plus efficaces par rapport à celles produites par la méthode de Gerschgorin.

Notre objectif dans cet article est de développer une méthode utile d'un point de vue pratique, pour calculer des bornes les plus efficaces possible sur les valeurs propres des matrices d'intervalles symétriques réelles.

Une matrice d'intervalles est définie comme un ensemble des matrices tel que

$$\mathbf{A} := [\underline{A}, \overline{A}] = \left\{ A \in \mathbb{R}^{n \times n}; \underline{A} \leq A \leq \overline{A} \right\},$$

où $\underline{A}, \overline{A} \in \mathbb{R}^{n \times n}$, sont des matrices données et $A =: (a_{ij})$, avec $a_{ij} \in \mathbb{R}$ pour $i, j = 1, \ldots, n$. $\underline{A} =: (\underline{a}_{ij})$ et $\overline{A} =: (\overline{a}_{ij})$, avec $\underline{a}_{ij}, \overline{a}_{ij} \in \mathbb{R}$. L'inégalité $\underline{A} \leq \overline{A}$, est considérée comme une comparaison entre les éléments $\underline{a}_{ij}$ de $\underline{A}$ avec les éléments $\overline{a}_{ij}$ de $\overline{A}$ pour $i, j = 1, \ldots, n$.
Nous définissons également une matrice d'intervalles symétrique par :

$$\mathbf{A}^s := \left\{ A \in \mathbf{A} | A^T = A \right\},$$

où $A \in \mathbb{R}^{n \times n}$, est une matrice symétrique réelle. On désigne la matrice médian et la matrice

rayon de la matrice d'intervalles $\mathbf{A}$, respectivement par :

$$A_c := \frac{1}{2}(\underline{A} + \overline{A}), \quad A_\Delta := \frac{1}{2}(\overline{A} - \underline{A}).$$

Une matrice symétrique réelle admette n valeurs propres réelles triées dans un ordre décroissant comme suit :

$$\lambda_1(A) \geq \lambda_2(A) \geq \cdots \geq \lambda_n(A).$$

Et pour une matrice d'intervalles symétrique réelle, les valeurs propres sont définies comme suit :

$$\lambda_i(\mathbf{A}^s) = [\underline{\lambda_i}(\mathbf{A}^s), \overline{\lambda_i}(\mathbf{A}^s)] := \{\lambda_i(A) | A \in \mathbf{A}^s\}, \quad i = 1, \ldots, n.$$

La structure de cet article est la suivante. Dans la section 3.3, plusieurs méthodes pour calculer les bornes des valeurs propres des matrices intervalles symétriques sont présentées. La section 3.4, présente une nouvelle approche et les principaux résultats. Les résultats numériques sont présentés dans la section 3.5.

3.3 Travaux existant dans la littérature

La méthode de Rohn développée dans [99], calcule les bornes sur les valeurs propres de la matrice d'intervalles symétrique $\mathbf{A}^s$, comme suit

Théorème 3.3.1 ([99]). *Les bornes sur les valeurs propres de la matrice d'intervalles symétrique $\boldsymbol{A}^s$, sont données par*

$$\lambda_i(\boldsymbol{A}^s) \in [\underline{\lambda_i}(A_c) - \rho(A_\Delta), \overline{\lambda_i}(A_c) + \rho(A_\Delta)], \quad i = 1, \ldots, n.$$

$\rho(A_\Delta)$ est le rayon spectral de la matrice A_Δ.

La méthode de Gerschgorin développée dans [11], calcule les valeurs α_i (les bornes infé-

rieures sur les valeurs propres d'une matrice d'intervalles symétrique réelle) comme suit

$$\alpha_i = \max\left\{0, -\left(\underline{a_{ii}} - \sum_{i \neq j}^{n} \max\left\{|\underline{a_{ij}}|, |\overline{a_{ij}}|\right\} \frac{d_j}{d_i}\right)\right\}, \tag{3.1}$$

avec $d_i > 0, \forall i = 1, \ldots, n$. Le vecteur scaling est $d = \overline{x} - \underline{x}$, ce qui signifie que les variables avec un spectre plus large ont un plus grand impact sur la qualité du sous-estimateur par rapport aux variables avec un spectre plus petit. Pour plus de détails, voir [11].

La méthode de Hladik développée dans [97], est basée sur la propriété d'entrelacement de Cauchy pour les valeurs propres d'une matrice symétrique [103, 104, 105]. Cette méthode implique deux versions d'entrelacement. La première est la version directe qui est basée directement sur le théorème de Cauchy. La seconde est la version indirecte qui est basée sur le théorème de Weyl [104, 105].

La méthode de maximisation diagonale de Hladik (DM) développée dans [97], est identique à la méthode précédente, sauf que dans cette méthode, les auteurs utilisent une matrice d'intervalles symétrique de maximisation diagonale au lieu de la matrice d'intervalles symétrique originale. Pour plus de détails, voir [97].

La méthode de Hertz développée dans [106], est basée sur la construction d'un ensemble de 2^{n-1} matrices, de sorte que les plus petites valeurs propres de cet ensemble de matrices sont les bornes exactes des valeurs propres de la matrice d'intervalles symétrique originale.

3.4 Nouvelle approche et résultats principaux

3.4.1 Nouvelle approche

Dans ce qui suit, nous présentons une proposition générale (la base de notre nouvelle approche) qui prouve des bornes valides sur les valeurs propres pour toute sélection d'une matrice d'intervalles symétrique à partir de la matrice d'intervalles symétrique originale $\mathbf{A}^s$. La sélection d'une telle matrice d'intervalles symétrique est basée sur les éléments diagonaux de la matrice d'intervalles symétrique d'origine $\mathbf{A}^s$.

Définition 3.4.1. *Soit $LV(m) = \{i \in \mathbb{N}\}$, un ensemble d'indices, et m est l'indice de l'ensemble LV. Par exemple, $LV(1) = \{2, 4, 5\}$, $LV(2) = \{1, 2, 4, 6\}$, pour $m = 1, 2$.*

$LV(m)$ est un ensemble qui représente une matrice d'intervalles symétrique, où tous ses éléments diagonaux dont les indices sont dans l'ensemble $LV(m)$, sont fixés à leurs bornes inférieures.

Définition similaire, $UV(m) = \{i \in \mathbb{N}\}$ est un ensemble qui représente une matrice d'intervalles symétrique, où tous ses éléments diagonaux dont les indices sont dans l'ensemble $UV(m)$, sont fixés à leurs bornes supérieures.

On définit une matrice d'intervalles symétrique $\mathbf{A}^s_{LV(m)}$ obtenue de la matrice d'intervalles symétrique originale $\mathbf{A}^s$, par :

$$\mathbf{A}^s_{LV(m)} = \begin{cases} \underline{a_{ij}} & \text{si } i = j = k \quad \text{avec } k \in LV(m) \\ \mathbf{a}_{ij} & \text{sinon} \end{cases}$$

Proposition 3.4.1. *Pour toute sélection d'une matrice d'intervalles symétrique $\boldsymbol{A}^s_{LV(m)}$ obtenue de $\boldsymbol{A}^s$, pour chaque $i \in \{1, \ldots, n\}$, on a :*

$$\underline{\lambda_i}(\boldsymbol{A}^s) \geq \underline{\lambda_i}(\boldsymbol{A}^s_{LV(m)}).$$

Démonstration. Pour chaque $A \in \mathbf{A}^s$, avec $a_{ij} \in [\underline{a_{ij}}, \overline{a_{ij}}]$. La formule de la valeur propre du théorème de Courant-Fischer [103, 104], est donnée par :

$$\lambda_i(A) = \max_{V \subseteq \mathbb{R}^n; \dim\, V = i} \left\{ \min_{x^T x = 1} x^T A x \right\}$$

$$= \max_{V \subseteq \mathbb{R}^n; \dim\, V = i} \left\{ \min_{x^T x = 1} \left(\sum_{i=1}^{n} a_{ii} x_i^2 + \sum_{i \neq j}^{n} a_{ij} x_i x_j \right) \right\}.$$

Maintenant, tous les éléments diagonaux dont les indices sont inclus dans l'ensemble $LV(m)$, sont fixés à leurs bornes inférieures. Par conséquent, l'égalité ci-dessus devient comme suit

$$\max_{V \subseteq \mathbb{R}^n; \dim\, V = i} \left\{ \min_{x^T x = 1} \left(\sum_{i=1}^{n} a_{ii} x_i^2 + \sum_{i \neq j}^{n} a_{ij} x_i x_j \right) \right\}$$

$$\geq \max_{V \subseteq \mathbb{R}^n; \dim\, V = i} \left\{ \min_{x^T x = 1} \left(\sum_{k \in LV(m)} \underline{a_{kk}} x_k^2 + \sum_{i \notin LV(m)}^{n} a_{ii} x_i^2 + \sum_{i \neq j}^{n} a_{ij} x_i x_j \right) \right\}$$

$$= \max_{V \subseteq \mathbb{R}^n; \dim\, V = i} \left\{ \min_{x^T x = 1} x^T A_{LV(m)} x \right\} = \lambda_i(A_{LV(m)}), \quad \forall A_{LV(m)} \in \mathbf{A}^s_{LV(m)}.$$

Donc,

$$\underline{\lambda_i}(\mathbf{A}^s) \geq \underline{\lambda_i}(\mathbf{A}^s_{LV(m)}), \quad \forall i = 1, \ldots, n, \quad \forall m$$

□

De la même manière, on peut prouver que $\overline{\lambda_i}(\mathbf{A}^s) \leq \overline{\lambda_i}(\mathbf{A}^s_{UV(m)}), \quad \forall i = 1, \ldots, n.$

Définition 3.4.2. *De même avec la définition 3.4.1,* $LV1(m), LV2(m), UV1(m)$ *et* $UV2(m)$ *sont des ensembles d'indices.*

$LV1(m) = \{i \in \mathbb{N}\}$ *est un ensemble qui représente une matrice d'intervalles symétrique, où tous ses éléments diagonaux dont les indices sont dans l'ensemble* $LV1(m)$*, sont fixés à leurs bornes inférieures.*

$LV2(m) = \{i \in \mathbb{N}\}$ est un ensemble qui représente une matrice d'intervalles symétrique, où tous ses éléments diagonaux dont les indices sont dans l'ensemble $LV2(m)$, sont des intervalles.

$UV1(m) = \{i \in \mathbb{N}\}$ est un ensemble qui représente une matrice d'intervalles symétrique, où tous ses éléments diagonaux dont les indices sont dans l'ensemble $UV1(m)$, sont fixés à leurs bornes supérieures.

$UV2(m) = \{i \in \mathbb{N}\}$ est un ensemble qui représente une matrice d'intervalles symétrique, où tous ses éléments diagonaux dont les indices sont dans l'ensemble $UV2(m)$, sont des intervalles.

Maintenant, on définit deux types de matrices d'intervalles symétriques pour le calcul des bornes inférieures sur les valeurs propres et deux types de matrices d'intervalles symétriques pour les bornes supérieures sur les valeurs propres de la matrice d'intervalles symétrique $\mathbf{A}^s$. Ces matrices d'intervalles symétriques sont obtenues à partir de $\mathbf{A}^s$ comme suit :

$$\mathbf{A}^s_{LV1(m)} = \begin{cases} \underline{a_{ij}} & \text{si } i = j = k, \quad \text{avec } k \in LV1(m) \\ \mathbf{a}_{ij} & \text{sinon} \end{cases},$$

$$\mathbf{A}^s_{UV1(m)} = \begin{cases} \overline{a_{ij}} & \text{si } i = j = k, \quad \text{avec } k \in UV1(m) \\ \mathbf{a}_{ij} & \text{sinon} \end{cases};$$

$$\mathbf{A}^s_{LV2(m)} = \begin{cases} \underline{a_{ij}} & \text{si } i = j \neq l, \quad \text{avec } l \in LV2(m) \\ \mathbf{a}_{ij} & \text{sinon} \end{cases},$$

$$\mathbf{A}^s_{UV2(m)} = \begin{cases} \overline{a_{ij}} & \text{si } i = j \neq l, \quad \text{avec } l \in UV2(m) \\ \mathbf{a}_{ij} & \text{sinon} \end{cases};$$

avec $\mathbf{a}_{ij} = [\underline{a_{ij}}, \overline{a_{ij}}]$, $\underline{a_{ij}} < \overline{a_{ij}}$ et $\underline{a_{ij}}, \overline{a_{ij}} \in \mathbb{R}$, pour $i, j = 1, \ldots, n$ et $\forall m$.

On utilise $LV1(m)$ et $LV2(m)$ pour le calcul des bornes inférieures sur les valeurs propres des matrices d'intervalles symétriques, et on utilise $UV1(m)$ et $UV2(m)$ pour le calcul des bornes supérieures sur les valeurs propres des matrices d'intervalles symétriques.

La contribution principale de ce travail est basée sur les éléments diagonaux de la matrice d'intervalles symétrique $\mathbf{A}^s$, qui est la sélection de matrices d'intervalles symétriques spécifiques obtenues à partir de la matrice d'intervalles symétrique originale $\mathbf{A}^s$, pour produire des bornes valides sur les valeurs propres pour $\mathbf{A}^s$. Donc, on va utiliser le résultat bien connu donné dans [104, 105], pour identifier une meilleure sélection de matrices d'intervalles symétriques qui, en général, peuvent produire des bornes aussi efficaces que possible sur les valeurs propres de la matrice d'intervalles symétrique originale $\mathbf{A}^s$.

3.4.2 Algorithme

Le résultat utilisé dans [104, 105], est la somme des carrés des entrées d'une matrice normale égale la somme des carrés de ses valeurs propres. Donc, nous présentons un algorithme basé sur ce résultat, pour la sélection des matrices d'intervalles symétriques spécifiques. On remarque dans l'algorithme présenté que $r = \dfrac{n}{2}$ si n est pair ou $r = \dfrac{n-1}{2}$ si n est impair, et ça parce que les ensembles $LV1(m)$ et $LV2(m)$ seront reconstruits à partir de $i = r$ jusqu'à $i = n-2$. En d'autres termes, les matrices d'intervalles symétriques $\mathbf{A}^s_{LV1(m)}$ et $\mathbf{A}^s_{LV2(m)}$ pour $i = r, \ldots, n-2$ sont déjà générées lorsque i varie de 0 à $r-1$. En outre, si $i = n-1$ et $j = n$, alors tous les indices sont ajoutés dans l'ensemble $LV1(m)$, donc tous les éléments de diagonale de $\mathbf{A}^s_{LV1(m)}$ sont fixés à leurs bornes inférieures, et ce travail déjà effectué dans [97]. Dans l'étape 2 de l'algorithme proposé, S est un vecteur de nombres réels et Λ est une liste d'ensembles qui contiennent des indices. Aux étapes 6 et 9, la somme est calculée par

Algorithme 9 Sélection des matrices d'intervalles $O(n^3)$

1. Poser $r = \dfrac{n}{2}$ si n est pair ou $r = \dfrac{n-1}{2}$ si n est impair.
2. Poser $m = 1$, un vecteur S et une liste $\Lambda = \{\emptyset\}$.
3. **Pour** $i = 0, \ldots, r-1$ **faire**
4. **Pour** $j = i+1, \ldots, n$ **faire**
5. Construire l'ensemble $LV1(m)$ et si $i > 0$ ajouter tous les indices $1, \ldots, i$ dans l'ensemble $LV1(m)$.
6. Ajouter l'indice j dans l'ensemble $LV1(m)$ et calculer la somme des carrés des éléments diagonaux de $\mathbf{A}^s_{LV1(m)}$. Stocker le résultat dans $S(m)$.
7. Ajouter l'ensemble $LV1(m)$ dans la liste Λ et poser $m = m + 1$.
8. Construire l'ensemble $LV2(m)$ et si $i > 0$ ajouter tous les indices $1, \ldots, i$ dans l'ensemble $LV2(m)$.
9. Ajouter l'indice j dans l'ensemble $LV2(m)$ et calculer la somme des carrés des éléments diagonaux de $\mathbf{A}^s_{LV2(m)}$. Stocker le résultat dans $S(m)$.
10. Ajouter l'ensemble $LV2(m)$ dans la liste Λ et poser $m = m + 1$.
11. **fin pour, fin pour**
12. Sélectionner de Λ, les n ensembles correspondant aux n valeurs minimales du vecteur S. Sélectionner également les n ensembles correspondant aux n valeurs maximales du vecteur S.

l'arithmétique d'intervalle comme suit

$$S(m) = \sum_{i=1}^{n} |\mathbf{A}^s_{LV1(m),ii}|^2 \qquad \text{si } m \text{ est impair,}$$

et

$$S(m) = \sum_{i=1}^{n} |\mathbf{A}^s_{LV2(m),ii}|^2 \qquad \text{si } m \text{ est pair,}$$

où $\mathbf{A}^s_{LV1(m),ii}$ est l'élément diagonal de la matrice d'intervalles symétrique $\mathbf{A}^s_{LV1(m)}$ (même chose pour $\mathbf{A}^s_{LV2(m)}$). Considérant seulement les éléments diagonaux parce que toutes les matrices d'intervalles symétriques $\mathbf{A}^s_{LV1(m)}$ ont les mêmes éléments non diagonaux avec la matrice d'intervalles symétrique originale $\mathbf{A}^s$ et ils diffèrent seulement sur les éléments diagonaux. Enfin, à l'étape 12, nous sélectionnons les $2n$ ensembles (ou matrices) correspondant

aux n valeurs minimales et maximales, pour produire des bornes aussi efficaces que possible sur les valeurs propres. Par exemple, pour les valeurs propres de borne inférieure négatives, il est préférable de choisir la valeur minimale de la somme des carrés, sinon, pour les valeurs propres de borne inférieure positives, choisir la valeur maximale de la somme des carrés serait un bon choix. Les sorties de l'algorithme proposé sont $2n$ matrices d'intervalles symétriques, mais pas nécessairement n matrices $\mathbf{A}^s_{LV1(m)}$ et n matrices $\mathbf{A}^s_{LV2(m)}$.

Pour mieux comprendre comment utiliser le vecteur S, les ensembles $LV1(m)$, $LV2(m)$ et la liste Λ dans l'algorithme proposé, nous présentons une exécution simple (sans matrice d'intervalles) de l'algorithme avec $n = 4$. Aux étapes 1 et 2, on a $r = 2$, $m = 1$ et $\Lambda = \{\emptyset\}$. Aux étapes 5 et 6, on construire l'ensemble $LV1(1) = \{1\}$, on calcule la somme (notée s_1) et on la stocke dans le vecteur S, donc $S = (s_1)$. À l'étape 7 on ajoute l'ensemble $LV1(1)$ dans la liste Λ, donc $\Lambda = \{LV1(1)\}$. Ici on remarque que la somme s_1 dans le vecteur S est une identification de l'ensemble $LV1(1)$ dans la liste Λ. De manière similaire pour les étapes 8, 9 et 10, on a $m = 2$, on construit l'ensemble $LV2(2) = \{1\}$, on calcule la somme (notée s_2) et on la stocker dans le vecteur S, donc $S = (s_1, s_2)$. On ajoute l'ensemble $LV2(2)$ dans la liste Λ, donc $\Lambda = \{LV1(1), LV2(2)\}$. À l'étape 12 on aura ce qui suit :

$$S = (s_1, s_2, s_3, \ldots, s_{m-1}),$$

$$\Lambda = \{LV1(1), LV2(2), LV1(3), \ldots, LV2(m-1)\}.$$

Enfin, par exemple, si les n valeurs minimales dans S sont s_1, s_2, s_4 et s_5, alors les ensembles qui doivent être choisis sont $LV1(1)$, $LV2(2)$, $LV2(4)$ et $LV1(5)$. De même pour les n maximales.

Pour la complexité, l'algorithme implique deux boucles qui dépendent de n. À l'intérieur de ces deux boucles, dans les pires cas, on a : $\frac{n}{2}$ instructions à l'étape 5, $2n + 2$ instructions ($2n$ opérations élémentaires) à l'étape 6, 2 instructions à l'étape 7, $\frac{n}{2}$ instructions à l'étape

8, $2n+2$ instructions ($2n$ opérations élémentaires) à l'étape 9 et 2 instructions à l'étape 10. Alors, la complexité totale sans l'étape 12 est $O(n^3)$. Le nombre d'instructions dans l'étape 12 est $m-1$ qui peut être calculé comme suit :

$$m-1=\sum_{i=0}^{r-1}\sum_{j=i+1}^{n}2=-r(-2n+r-1).$$

Si n est impair on a $m-1=\frac{3}{4}(n-1)(n+1)$, et si n est pair on a $m-1=\frac{1}{4}n(3n+2)$, ce qui signifie que la complexité de l'étape 12 est $O(n^2)$. Alors, la complexité de l'algorithme est le terme dominant entre $O(n^3)$ et $O(n^2)$, qui est $O(n^3)$.

3.4.3 Résultats principaux

Dans cette partie de cette section, les résultats de calcul des bornes sur les valeurs propres de la matrice d'intervalles symétrique $\mathbf{A}^s$ sont présentés. Ces résultats sont valides et le nombre d'éléments fixés à leurs bornes inférieures, n'a aucun effet sur l'exactitude de ces résultats puisque nous avons déjà prouvé dans la proposition 3.4.1 que pour toute sélection de matrices d'intervalles symétriques $\mathbf{A}^s_{LV(m)}$, les bornes des valeurs propres produites à partir de ces matrices sont valides. Alors on suppose que les ensembles $LV1(m)$ et $LV2(m)$ contiennent un seul indice, on suppose également que les sorties de l'algorithme proposé sont n matrices d'intervalles symétriques $\mathbf{A}^s_{LV1(m)}$ et n matrices d'intervalles symétriques $\mathbf{A}^s_{LV2(m)}$ avec $m=1,\ldots,n$.

Théorème 3.4.1. *Pour chaque $i\in\{1,\ldots,n\}$, on a :*

$$\underline{\lambda_i}(\boldsymbol{A}^s)\geq\max_{m=1,\ldots,n}\underline{\lambda_i}(\boldsymbol{A}^s_{LV1(m)}).$$

Démonstration. Pour toute $A\in\mathbf{A}^s$, avec $a_{ij}\in[\underline{a_{ij}},\overline{a_{ij}}]$, on peut commencer par le théorème de Courant-Fischer [103, 104]

$$\lambda_i(A) = \max_{V\subseteq\mathbb{R}^n;\dim\ V=i}\left\{\min_{x^Tx=1} x^TAx\right\}$$

$$= \max_{V\subseteq\mathbb{R}^n;\dim\ V=i}\left\{\min_{x^Tx=1}\left(\sum_{i=1}^n a_{ii}x_i^2 + \sum_{i\neq j}^n a_{ij}x_ix_j\right)\right\},$$

maintenant, pour $i=j=k$ avec $k\in LV1(m)$, on suppose que $a_{ij}=\underline{a_{ij}}=\underline{a_{kk}}$, sinon on a $\forall a_{ij}\in[\underline{a_{ij}},\overline{a_{ij}}]$, alors

$$\max_{V\subseteq\mathbb{R}^n;\dim\ V=i}\left\{\min_{x^Tx=1}\left(\sum_{i=1}^n a_{ii}x_i^2 + \sum_{i\neq j}^n a_{ij}x_ix_j\right)\right\}$$

$$\geq \max_{V\subseteq\mathbb{R}^n;\dim\ V=i}\left\{\min_{x^Tx=1}\left(\underline{a_{kk}}x_k^2 + \sum_{i\neq k}^n a_{ii}x_i^2 + \sum_{i\neq j}^n a_{ij}x_ix_j\right)\right\}$$

$$= \max_{V\subseteq\mathbb{R}^n;\dim\ V=i}\left\{\min_{x^Tx=1} x^TA_{LV1(m)}x\right\} = \lambda_i(A_{LV1(m)}),\quad \forall A_{LV1(m)}\in\mathbf{A}^s_{LV1(m)},$$

donc,

$$\underline{\lambda_i}(\mathbf{A}^s)\geq\underline{\lambda_i}(\mathbf{A}^s_{LV1(m)}),\quad \forall i=1,\ldots,n,\quad \forall m=1,\ldots,n.$$

Alors,

$$\underline{\lambda_i}(\mathbf{A}^s)\geq\max_{m=1,\ldots,n}\underline{\lambda_i}(\mathbf{A}^s_{LV1(m)}),\quad \forall i=1,\ldots,n.$$

□

De façon similaire, pour $i=j=k$ avec $k\in UV1(m)$, on suppose que $a_{ij}=\overline{a_{ij}}=\overline{a_{kk}}$,

sinon on a $a_{ij} \in [\underline{a_{ij}}, \overline{a_{ij}}]$, on peut prouver que

$$\overline{\lambda_i}(\mathbf{A}^s) \leq \min_{m=1,\ldots,n} \overline{\lambda_i}(\mathbf{A}^s_{UV1(m)}), \qquad i = 1, \ldots, n.$$

Le théorème suivant est analogue au théorème 3.4.1 puisque la différence entre eux est seulement dans les ensembles $LV1(m)$ et $LV2(m)$.

Théorème 3.4.2. *Pour chaque $i \in \{1, \ldots, n\}$, on a :*

$$\underline{\lambda_i}(\boldsymbol{A}^s) \geq \max_{m=1,\ldots,n} \underline{\lambda_i}(\boldsymbol{A}^s_{LV2(m)}).$$

De façon similaire, pour $i = j \neq l$ avec $l \in UV2(m)$, on suppose que $a_{ij} = \overline{a_{ij}}$, on a :

$$\overline{\lambda_i}(\mathbf{A}^s) \leq \min_{m=1,\ldots,n} \overline{\lambda_i}(\mathbf{A}^s_{UV2(m)}), \qquad i = 1, \ldots, n.$$

À ce point, on peut présenter notre résultat principal en résumant les théorèmes 3.4.1 et 3.4.2 dans le corollaire suivant :

Corollaire 3.4.1. *Soit une matrice d'intervalles symétrique $\boldsymbol{A}^s \in \mathbb{IR}^{n\times n}$, pour $i, m = 1, \ldots, n$, alors on a :*

$$\underline{\lambda_i}(\boldsymbol{A}^s) \geq \max\left\{\max_m \underline{\lambda_i}(\boldsymbol{A}^s_{LV1(m)}), \max_m \underline{\lambda_i}(\boldsymbol{A}^s_{LV2(m)})\right\},$$

$$\overline{\lambda_i}(\boldsymbol{A}^s) \leq \min\left\{\min_m \overline{\lambda_i}(\boldsymbol{A}^s_{UV1(m)}), \min_m \overline{\lambda_i}(\boldsymbol{A}^s_{UV2(m)})\right\},$$

Démonstration. Soit $\mathbf{A}^s \in \mathbb{IR}^{n\times n}$, à partir des théorèmes 3.4.1 et 3.4.2, pour $i, m = 1, \ldots, n$, on a :

$$\underline{\lambda_i}(\mathbf{A}^s) \geq \max_{m=1,\ldots,n} \underline{\lambda_i}(\mathbf{A}^s_{LV1(m)}) \quad \text{and} \quad \underline{\lambda_i}(\mathbf{A}^s) \geq \max_{m=1,\ldots,n} \underline{\lambda_i}(\mathbf{A}^s_{LV2(m)}),$$

alors,

$$\underline{\lambda_i}(\mathbf{A}^s) \geq \max\left\{\max_m \underline{\lambda_i}(\mathbf{A}^s_{LV1(m)}), \max_m \underline{\lambda_i}(\mathbf{A}^s_{LV2(m)})\right\}.$$

□

De la même manière, on peut prouver la deuxième inégalité présentée dans le corollaire 3.4.1.

On désigne respectivement par $A^{(m)}_{l1c}, A^{(m)}_{u1c}, A^{(m)}_{l2c}, A^{(m)}_{u2c}$, les matrices médianes des matrices d'intervalles symétriques $\mathbf{A}^s_{LV1(m)}, \mathbf{A}^s_{UV1(m)}, \mathbf{A}^s_{LV2(m)}, \mathbf{A}^s_{UV2(m)}$. Et $A^{(m)}_{l1\Delta}$, $A^{(m)}_{u1\Delta}, A^{(m)}_{l2\Delta}, A^{(m)}_{u2\Delta}$, les matrices rayons pour les mêmes matrices d'intervalles symétriques pour $m = 1, \ldots, n$.

Le corollaire 3.4.1 présente des bornes inférieures et supérieures valides sur les valeurs propres de la matrice d'intervalles symétrique réelle $\mathbf{A}^s$. De plus, les matrices d'intervalles symétriques obtenues de $\mathbf{A}^s$, peuvent produire des bornes efficaces sur les valeurs propres de la matrice d'intervalles symétrique réelle originale. En appliquant les bornes de Rohn du théorème 3.3.1 sur les matrices d'intervalles symétriques générées, on obtient le corollaire suivant :

Corollaire 3.4.2. *Soit $\boldsymbol{A}^s \in \mathbb{IR}^{n\times n}$. Alors pour $i, m = 1, \ldots, n$ on a :*

$$\underline{\lambda_i}(\boldsymbol{A}^s) \geq \max\left\{\max_m \left(\lambda_i(A^{(m)}_{l1c}) - \rho(A^{(m)}_{l1\Delta})\right), \max_m \left(\lambda_i(A^{(m)}_{l2c}) - \rho(A^{(m)}_{l2\Delta})\right)\right\},$$

$$\overline{\lambda_i}(\boldsymbol{A}^s) \leq \min\left\{\min_m \left(\lambda_i(A^{(m)}_{u1c}) + \rho(A^{(m)}_{u1\Delta})\right), \min_m \left(\lambda_i(A^{(m)}_{u2c}) + \rho(A^{(m)}_{u2\Delta})\right)\right\}.$$

3.4.4 Exemple

Prenons un exemple qui se trouve dans [97] :

$$\mathbf{A}^s = \begin{pmatrix} [2975, 3025] & [-2015, -1985] & 0 & 0 \\ [-2015, -1985] & [4965, 5035] & [-3020, -2980] & 0 \\ 0 & [-3020, -2980] & [6955, 7045] & [-4025, -3975] \\ 0 & 0 & [-4025, -3975] & [8945, 9055] \end{pmatrix}$$

Par la suite, les bornes des valeurs propres de la matrice d'intervalles symétrique $\mathbf{A}^s$ sont calculées. Les significations des éléments $(Q), (R), (D1), (D2), (I1), (I2), (B), (P)$ et (O) sont les suivants :

— (Q) Bornes calculées par la méthode de Leng [96].

— (R) Bornes calculées par la méthode de Rohn [99].

— (D1) Bornes calculées par la méthode 1 avec la règle 1 dans [97].

— (D2) Bornes calculées par la méthode 1 avec la règle 2 dans [97].

— (I1) Bornes calculées par la méthode 2 avec la règle 1 dans [97].

— (I2) Bornes calculées par la méthode 2 avec la règle 2 dans [97].

— (B) Meilleures bornes calculées dans [97].

— (P) Bornes calculées par la méthode proposée.

— (O) Bornes optimales trouvées dans [95].

Les résultats du tableau 3.1 montrent que la méthode proposée produit des bornes plus efficaces et meilleures (P) par rapport aux bornes de Rohn (R) et de Leng (Q). Noté dans [97] que les méthodes développées ne sont pas si efficaces lorsque les intervalles $[\underline{\lambda_i}(\mathbf{A}^s), \overline{\lambda_i}(\mathbf{A}^s)]$ pour $i = 1, \ldots, n$, ne se chevauchent pas, par conséquent, les auteurs choisissent les meilleures

TABLE 3.1 – Comparaison des résultats des bornes de la méthode proposée avec différentes bornes de méthodes sur les valeurs propres pour l'exemple 1.

	$[\underline{\lambda_1}(\mathbf{A}^s), \overline{\lambda_1}(\mathbf{A}^s)]$	$[\underline{\lambda_2}(\mathbf{A}^s), \overline{\lambda_2}(\mathbf{A}^s)]$	$[\underline{\lambda_3}(\mathbf{A}^s), \overline{\lambda_3}(\mathbf{A}^s)]$	$[\underline{\lambda_4}(\mathbf{A}^s), \overline{\lambda_4}(\mathbf{A}^s)]$
(Q)	[12550.5313,12730.531]	[6974.459,7154.459]	[3299.848,3479.848]	[815.161,995.161]
(R)	[12560.629,12720.433]	[6984.557,7144.360]	[3309.946,3469.750]	[825.259,985.063]
(D1)	[8945.000,12720.227]	[4945.000,9055.000]	[2924.504,6281.721]	[825.259,3025.000]
(D2)	[8945.000,12720.227]	[2945.000,9453.444]	[1708.932,6281.721]	[825.259,3025.000]
(I1)	[12560.629,12720.4333]	[6984.557,7144.360]	[3309.946,3469.750]	[825.259,985.063]
(I2)	[12560.629,12720.4333]	[6984.557,7144.360]	[3309.946,3469.750]	[825.259,985.063]
(B)	[12560.629,12720.227]	[6990.761,7138.180]	[3320.286,3459.432]	[837.063,973.199]
(P)	[12560.685,12720.378]	[6994.418,7134.511]	[3329.404,3450.475]	[841.923,968.096]
(O)	[12560.837,12720.227]	[7002.282,7126.828]	[3337.078,3443.312]	[842.925,967.108]

bornes de combinaison (B) de toutes les méthodes. Cependant, la méthode proposée produit toujours des bornes plus efficaces par rapport aux bornes de Rohn et même aux meilleures bornes (B) dans [97] dans ce cas. $\overline{\lambda_i}(\mathbf{A}^s) = 12720.227$ de (B) est obtenue à partir de la proposition 3.2 dans [97], et cette valeur est par hasard la valeur optimale. On peut facilement appliquer cette proposition sur la méthode proposée pour obtenir la même valeur.

3.4.5 Complexité

Les méthodes de calcul des bornes sur les valeurs propres des matrices d'intervalles symétriques réelles peuvent être classées par leurs complexités. La méthode de Hertz dans [106] calcule les bornes exactes sur les valeurs propres des matrices d'intervalles symétriques. Cependant, cette précision des bornes vient avec un coût de complexité de temps. La méthode de Hertz génère un nombre de matrices de sommets égale à 2^{n-1} pour arriver aux bornes exactes sur les valeurs propres, ce coût peut devenir prohibitif dans les calculs des matrices de grande dimension. La méthode de Gerschgorin dans [11] sacrifie la précision des bornes sur les valeurs propres des matrices d'intervalles symétriques, pour la vitesse des calculs. C'est une méthode avec une complexité de $O(n^2)$. La complexité des méthodes de Rohn et Hladik est $O(n^3)$.

La méthode proposée comporte deux procédures. La première procédure est l'algorithme de sélection des matrices d'intervalles, avec une complexité de $O(n^3)$ comme décrit dans la section 3.4.2. La deuxième procédure est le calcul des bornes sur les valeurs propres des matrices d'intervalles obtenues à partir de la première procédure. Nous avons mentionné dans la section 3.4.2, que les sorties de la première procédure sont $2n$ matrices d'intervalles. Puisque nous avons appliqué la méthode de Rohn sur les $2n$ matrices d'intervalles, alors la complexité de la deuxième procédure serait $2n * O(n^3)$, qui est $O(n^4)$. Alors, la complexité de la méthode proposée est le terme dominant parmi les complexités des deux procédures, qui dans notre cas est $O(n^4)$.

La méthode proposée a une structure simple et sa complexité est polynomiale. En outre, la méthode proposée produit, en général, des bornes plus efficaces sur les valeurs propres des matrices d'intervalles symétriques. Par conséquent, ces facteurs peuvent être considérés comme des avantages de mise en œuvre de la méthode proposée par rapport aux méthodes existantes dans la littérature.

3.5 Résultats numériques

L'objectif de notre travail est de produire des bornes inférieures plus efficaces sur les valeurs propres $\underline{\lambda_i}$ pour $i = 1, \ldots, n$, des matrices hessiennes d'intervalles symétriques, puis des sous-estimateurs plus efficaces dans la méthode αBB, pour résoudre les problèmes d'optimisation globale.

Dans l'expérience suivante, nous comparons l'efficacité de l'algorithme αBB en utilisant cinq méthodes différentes pour le calcul de chaque valeur α_i. La première est la méthode de scaled Gerschgorin qui calcule α_i par équation (3.1). La seconde est la méthode de Hladik développée dans [97]. La troisième est la méthode de Hladik maximisation diagonale (DM) développée dans [97]. La quatrième est la méthode proposée et la dernière est la méthode de Hertz [106].

L'algorithme αBB est implémenté dans le programme C++ et exécuté sur un ordinateur Dell avec un processeur Intel(R) Core(TM) i5-4210U avec une vitesse de 3,40 GHz et 8 Go de RAM. Les résultats de calcul de l'algorithme αBB utilisant la méthode proposée sont résumés dans le tableau 3.3. Les résultats de comparaison des performances de la méthode proposée avec les autres méthodes sont présentés dans les tableaux 3.4 et 3.5. Les critères suivants sont pris en considération :

— $N^o iter$ est le nombre d'itérations pour atteindre la valeur optimale ϵ-globale.

— Tcpu est le temps de calcul en secondes. Les valeurs déclarées sont les valeurs moyennes de 1000 fois exécution pour chaque fonction dans le tableau 3.2.

— σ est l'écart-type du temps de calcul de 1000 fois exécution pour chaque fonction dans le tableau 3.2.

— Fopt est la valeur optimale ϵ-globale minimale.

Le tableau 3.3 montre que la solution optimale minimale est obtenue par l'algorithme αBB dans un temps de calcul très court lors de l'utilisation de la méthode proposée. Le tableau 3.4, indique que les résultats diffèrent pour chaque fonction. Pour les fonctions (1,2,3,4 et 7), il n'y avait pas d'amélioration lors de l'utilisation de la maximisation diagonale de Hladik (DM) ou des bornes inférieures proposées sur les valeurs propres des matrices d'intervalles hessiennes des fonctions correspondantes. Cependant, en comparaison avec les autres méthodes d'approche, il y avait une amélioration jusqu'à 156 itérations pour la fonction (4) avec moins de temps de calcul. Pour le reste des fonctions, l'amélioration a été jusqu'à 1886 itérations entre l'utilisation de la méthode proposée et l'utilisation des autres méthodes. En moyenne 14 fonctions, en utilisant la méthode proposée, l'algorithme αBB nécessite 776 itérations pour atteindre la valeur minimale optimale alors qu'il nécessite 1430 itérations avec la méthode Hladik, 1184 itérations avec la méthode de scaled Gerschgorin et 846 itérations avec la méthode de maximisation diagonale Hladik (DM). On peut mesurer en pourcentage, l'amélioration moyenne sur le nombre d'itérations entre la méthode proposée et les autres méthodes d'approche. Alors, pour la méthode Hladik, il y a eu une amélioration d'environ de 45.7%. Pour la méthode de scaled Gerschgorin il y a eu une amélioration de 34.5%. Enfin, pour la méthode de Hladik (DM) il y a eu une amélioration de 8.3%. En utilisant la méthode de Hertz [106] qui calcule les bornes exactes sur les valeurs propres, l'algorithme αBB nécessite en moyenne de 507 itérations pour atteindre la valeur minimale optimale. Les résultats présentés dans le tableau 3.5 présentent l'étude de la vitesse temporelle de l'algorithme αBB avec différentes méthodes de calcul de chaque valeur α_i. Le tableau 3.5 montre que, dans la plupart des cas, l'algorithme αBB fonctionne plus rapidement avec la méthode proposée qu'avec les autres méthodes d'approche. De plus, les faibles écarts-types de plusieurs exé-

TABLE 3.2 – Collection de fonctions multivariables de test

	source	Fonction	Domaine
1	[13]	$sin(x_1 + x_2) + (x_1 - x_2)^2 - 1.5x_1 + 2.5x_2 + 1$	[-1.5,4]×[−3, 3]
2	[13]	$-sin((x_1 - 1)(x_1 - 2)(x_2 + 1))$	[-1,1]×[−2, 0]
3	[13]	$(x_2 - 5x_1^2/(4\pi^2) + 5x_1/\pi - 6)^2 + 10(1 - 1/(8\pi))cos(x_1) + 10$	[-5,10]×[0, 15]
4	[13]	$100(x_2 - x_1^2)^2 + (x_1 - 1)^2$	[-3,3]×[−1.5, 4.5]
5	[13]	$0.5(x_1^2 + x_2^2) - cos(10ln(2x_1))cos(10ln(3x_2)) + 1$	$[0.01,1]^2$
6	[13]	$(1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2))$	$[-2,2]^2$
		$(30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2))$	
7	[94]	$x_1^4 + x_2 - (x_1 + x_2^2)^2$	[1,3]×[−1, 1]
8	[94]	$(x_1 + 10x_2)^2 + 5(x_3 - x_4)^2 + (x_2 - 2x_3)^4 + 10(x_1 - x_4)^4$	$[0,1]^4$
9	[94]	$(2x_1 + x_2 - 3)^2 + (x_1x_2 - 1)^2$	$[0,4]^2$
10	[94]	$100(x_2 - x_1^2) + (1 - x_1)^2 + 90(x_4 - x_3^2)^2 + (1 - x_3)^2$	[1,2]×[−0.5, 0.5] × $[1, 2]^2$
		$+10.1((1 - x_2)^2 + (1 - x_4)^2) + 19.8((1 - x_2) + (1 - x_4))$	
11	[93]	$0.4x_1^{2/3}x_3^{-2/3} + 0.4x_2^{2/3}x_4^{2/3} + 10 - x_1 - x_2$	$[0.1,10]^4$
12	[107]	$\sum_{i=1}^{4} \left[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right]$	$[-3,3]^5$
13	[107]	$\sum_{i=2}^{3} \left[(x_{i-1} + 10x_i)^2 + 5(x_{i+1} - x_{i+2})^2 + (x_i - 2x_{i+1})^4 + 10(x_{i-1} - x_{i+2})^4\right]$	$[-4,5]^5$
14	[107]	$(x_1 - 1)^2 + \sum_{i=2}^{n} \left[i(2x_i^2 - x_{i-1})^2\right]$	$[-10,10]^5$

TABLE 3.3 – Résultats de calcul de l'utilisation de la méthode proposée dans l'algorithme αBB pour résoudre les problèmes tests énumérés dans le tableau 3.2.

Fonction	ϵ	N$^o iter$	Tcpu.10^{-4}	Fopt
1	1.10^{-8}	41	4.8259	1.913223
2	1.10^{-3}	146	17.3072	-0.999994
3	1.10^{-6}	120	14.3239	0.399250
4	1.10^{-7}	275	29.8433	0
5	1.10^{-4}	3261	948.007	0.000179
6	5.10^{-5}	829	116.375	3.000003
7	5.10^{-5}	752	120.295	-4
8	1.10^{-5}	118	24.8714	0
9	5.10^{-5}	76	7.27851	0
10	1.10^{-7}	32	3.72991	27.8846
11	5.10^{-4}	2068	562.093	-9.2
12	1.10^{-5}	1487	381.041	0
13	1.10^{-4}	510	107.736	0
14	1.10^{-4}	1159	320.712	0

cutions de l'algorithme αBB avec la méthode proposée indiquent que les résultats sont plus consistants par rapport à ceux des autres méthodes d'approche.

TABLE 3.4 – Résultats de la comparaison des performances de l'utilisation de la méthode proposée avec quatre méthodes différentes pour calculer les bornes inférieures sur les valeurs propres dans l'algorithme αBB pour les fonctions tests énumérés dans le tableau 3.2.

	Gerschgorin [11]		Hladik [97]		Hladik (DM) [97]		Proposed		Hertz [106]	
	N$^o iter$	Tcpu.10^{-4}	N$^o iter$	Tcpu.10^{-4}	N$^o iter$	Tcpu.10^{-4}	N$^o iter$	Tcpu.10^{-4}	N$^o iter$	Tcpu.10^{-4}
1	43	4.9736	41	4.9966	41	4.8267	41	4.8259	41	4.8466
2	222	27.2429	150	17.6776	146	17.3216	146	17.3072	126	14.7452
3	120	13.0445	140	16.7857	120	14.334	120	14.3239	106	11.104
4	403	38.096	431	54.825	275	31.7039	275	29.8433	210	17.9807
5	3907	1294.21	4257	1454.67	3506	1082.82	3261	948.007	3131	908.694
6	1174	172.018	1087	168.992	1078	165.23	829	116.375	782	104.63
7	752	96.208	754	128.682	752	125.748	752	120.295	752	114.691
8	682	144.321	155	32.098	144	29.1104	118	24.8714	60	12.6049
9	110	11.3822	79	7.8877	77	7.61806	76	7.27851	58	5.27939
10	157	19.062	73	10.7122	29	3.89415	28	3.72991	28	3.68711
11	1546	292.617	4819	1627.08	2080	581.392	2068	562.093	622	110.437
12	3812	864.817	4287	1623.55	1803	488.402	1487	381.041	593	125.109
13	1360	268.78	710	161.567	585	127.218	510	107.736	172	41.875
14	2292	499.195	3045	1118.77	1216	308.414	1159	320.712	422	86.3169
Avg.	1184.3	267.569	1430.6	459.164	846.6	213.431	776.4	189.889	507.4	111.571

TABLE 3.5 – Résultats de la comparaison du temps de calcul de l'utilisation de la méthode proposée avec quatre méthodes différentes pour calculer les bornes inférieures sur les valeurs propres dans l'algorithme αBB pour les fonctions tests énumérés dans le tableau 3.2.

	Gerschgorin [11]		Hladik [97]		Hladik (DM) [97]		Proposed		Hertz [106]	
	Tcpu.10^{-4}	$\sigma.10^{-5}$	Tcpu.10^{-4}	$\sigma.10^{-5}$	Tcpu.10^{-4}	$\sigma.10^{-5}$	Tcpu.10^{-4}	$\sigma.10^{-5}$	Tcpu.10^{-4}	$\sigma.10^{-5}$
1	4.9736	11.261	4.9966	8.174	4.8267	6.818	4.8259	6.301	4.8466	6.571
2	27.2429	37.149	17.6776	13.512	17.3216	14.747	17.3072	14.880	14.7452	15.226
3	13.0445	23.613	16.7857	17.790	14.334	20.001	14.3239	14.301	11.104	18.012
4	38.096	53.312	54.825	50.653	31.7039	36.673	29.8433	33.762	17.9807	22.269
5	1294.21	231.617	1454.67	251.445	1082.82	217.401	948.007	215.236	908.694	207.908
6	172.018	198.379	168.992	175.478	165.23	117.566	116.375	108.769	104.63	113.165
7	96.208	122.545	128.682	150.892	125.748	124.768	120.295	161.507	114.691	88.316
8	144.321	182.166	32.098	16.499	29.1104	19.490	24.8714	24.966	12.6049	12.525
9	11.3822	29.728	7.8877	10.268	7.61806	9.876	7.27851	10.559	5.27939	8.671
10	19.062	28.666	10.7122	11.122	3.89415	5.258	3.72991	5.273	3.68711	5.746
11	292.617	112.001	1627.08	221.993	581.392	208.696	562.093	186.571	110.437	96.274
12	864.817	180.026	1623.55	216.893	488.402	87.490	381.041	50.127	125.109	24.306
13	268.78	263.734	161.567	39.620	127.218	30.2015	107.736	30.026	41.875	58.625
14	499.195	300.813	1118.77	473.363	308.414	59.508	320.712	115.257	86.3169	20.649
Avg.	267.569	126.786	459.164	118.407	213.431	68.464	189.889	69.824	111.571	49.876

3.6 Conclusion

Dans cet article, nous avons considéré les bornes sur les valeurs propres des matrices d'intervalles symétriques réelles. Comme il est difficile de calculer des bornes exactes ou meilleures sur les valeurs propres des matrices d'intervalles symétriques pour tous les cas, nous avons présenté une méthode améliorante pour calculer des bornes aussi efficaces que possible sur les valeurs propres des matrices d'intervalles symétriques. Nous avons utilisé les bornes inférieures de la méthode proposée sur les valeurs propres pour montrer l'effet d'efficacité de nos bornes inférieures sur les valeurs propres des matrices hessiennes d'intervalles symétriques, dans l'algorithme αBB pour résoudre des problèmes d'optimisation globale.

CHAPITRE 4

OPTIMISATION SEMI-INFINIE

4.1 Introduction

Un problème d'optimisation semi-infinie consiste à optimiser une fonction objectif sur un ensemble infini de contraintes avec un nombre fini de variables [16, 15, 108, 18]. La programmation semi-infinie, ces dernières années, est devenu un domaine de recherche actif dans la programmation mathématique. Les problèmes d'optimisation semi-infinie se rencontrent dans diverses applications d'ingénierie et d'économie, par exemple, lorsqu'une contrainte doit être introduite pour chaque point d'une région géométrique.

4.2 Formulation d'un problème semi-infini [14]

La forme générale d'un problème semi-infini est donnée comme suit :

$$(PSI)\begin{cases} \min f(x) \\ g(x,t) \leq 0, \quad \forall t \in T \\ x \in \mathbb{R}^n \end{cases}$$

où T est un compact de $\mathbb{R}^m$, avec $f : \mathbb{R}^n \mapsto \mathbb{R}$ et $g : \mathbb{R}^n \times \mathbb{R}^m \mapsto \mathbb{R}$. Dans le cas général, l'ensemble des indices T peut dépendre de x, et on écrit $T = T(x)$.

Le problème semi-infini (PSI) est dit linéaire si les fonctions f et g sont linéaires par rapport à x.

Le problème semi-infini (PSI) est dit convexe si la fonction f est convexe, et la fonction g est convexe par rapport à x.

Le problème semi-infini (PSI) est dit non convexe dans tous les autres cas.

Définition 4.2.1. *Les variables de décision sont notées $x \in \mathbb{R}^n$ et les paramètres auxiliaires sont notés $t \in T$.*

4.2.1 Problème pratique d'optimisation semi-infinie

On va présenter un problème pratique modélisé en optimisation semi-infinie.

Conception des systèmes de contrôle MIMO [19]

Nous considérons le problème de conception de systèmes de contrôle à entrées multiples et sorties multiple (MIMO : multiple-input multiple-output, voir figure 4.1) dans le domaine de transformation de Laplace. Le vecteur $r(s)$ désigne la transformation de Laplace de l'entrée du système en boucle fermée. le vecteur $y(s)$ indique la transformation de Laplace de la sortie du système en boucle fermée. le vecteur $u(s)$ indique la transformation de Laplace de l'entrée. La fonction de transfert $P(s)$ est une matrice $(m \times m)$ dont les éléments sont des fonctions de la variable s et il est nécessaire de concevoir la matrice $(m \times m)$ de fonction de transfert de compensation $C(x, s)$ dont les éléments sont des fonctions de la variable s, avec des coefficients qui dépendent du vecteur de conception x de dimension n.

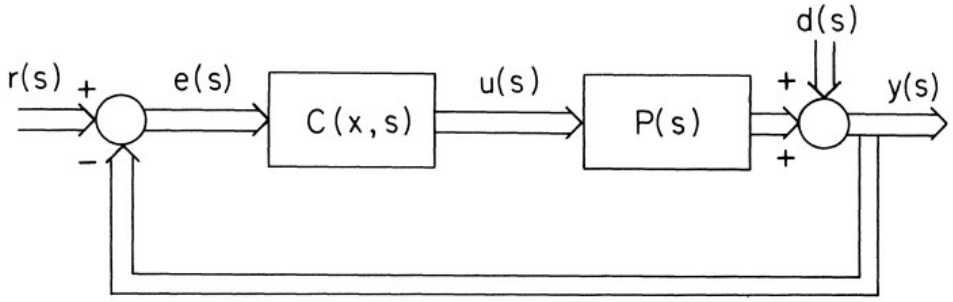

FIGURE 4.1 – Système de contrôle MIMO [19]

Ce système conduit à des équations, non differentiables, semi-infinies suivantes

$$y(s) = [I + P(s)C(x, s)]d(s) = Q(x, s)d(s)$$

$$u(s) = -C(x, s)Q(x, s)d(s) = R(x, s)d(s)$$

Soit $q(x, t)$ la plus grande valeur singulière de $Q(x.it)$, et $s(x, t)$ la plus grande valeur singulière de $R(x.it)$. avec $t \in \mathbb{R}$, et i est un nombre complexe. La variable t indique la fréquence.

La matrice $Q(x.it)$ est complexe. Puisque les plus grandes valeurs singulières sont des normes des matrices, pour rendre la réponse y du système pour une grande classe de perturbations, sans saturer indûment le système à la suite de u devenant grand, Les concepteurs de systèmes de contrôle s'efforcent de garder q et r petits sur les rangs de fréquences appropriées. Cela conduit à la formulation partielle du problème de conception du système de contrôle MIMO, sous forme d'un problème semi-infini suivant :

$$(PSI)\begin{cases} \min f(x) = \max_t q(x,t) \\ s(x,t) \leq b(t), \quad \forall t \in [t^1, t^2] \\ x \in [\underline{x}, \overline{x}] \end{cases}$$

où $b(t)$ est une fonction continue de la fréquence t.

4.3 Problème d'optimisation semi-infinie linéaire [15]

Soit le problème semi-infini linéaire suivant

$$(PSIL)\begin{cases} \min \sum_{i=1}^{n} c_i x_i \\ \sum_{i=1}^{n} a_i(t) x_i \geq b(t), \quad t \in T \\ x \in \mathbb{R}^n \end{cases}$$

où T est un compact de $\mathbb{R}^m$, avec $c_i \in \mathbb{R}$, $a_i : \mathbb{R}^m \mapsto \mathbb{R}$ pour $i = 1, \ldots, n$, et $b : \mathbb{R}^m \mapsto \mathbb{R}$.

4.3.1 Problème dual du problème semi-infini linéaire [15]

Soit le problème (primal) semi-infini linéaire suivant :

$$(PSIL)\begin{cases} \min \sum_{i=1}^{n} c_i x_i \\ \sum_{i=1}^{n} a_i(t) x_i \geq b(t), \quad t \in T \\ x \in \mathbb{R}^n \end{cases}$$

Soit $\delta = \{t_1, t_2, \ldots, t_n\}$ un sous ensemble de T.

On peut définir le problème dual du problème $(PSIL)$ comme suit :

$$(DSIL)\begin{cases} \max \sum_{j=1}^{n} b(t_j)y_j \\ \sum_{j=1}^{n} a_i(t_j)y_j = c_i, \quad i = 1, \ldots, n \\ t_j \in T, \quad y \in \mathbb{R}_+^n, \qquad j = 1, \ldots, n. \end{cases}$$

On pose $C = (c_1, \ldots, c_n) \in \mathbb{R}^n$, $a(t_j) = (a_1(t_j), \ldots, a_n(t_j)) \in \mathbb{R}^n$ pour $j = 1, \ldots, n$. Le problème dual $(DSIL)$ peut s'écrire comme suit :

$$(DSIL)\begin{cases} \max \sum_{j=1}^{n} b(t_j)y_j \\ \sum_{j=1}^{n} a(t_j)y_j = C, \quad \delta = (t_1, \ldots, t_n) \\ y \in \mathbb{R}_+^n. \end{cases}$$

Le couple $\{\delta, y\}$ est une solution de base réalisable du problème dual $(DSIL)$, où $\delta \subset T$ et $y \in \mathbb{R}_+^n$.

— **Solution de base :** $\{\delta, y\}$ est dite une solution de base pour le problème dual $(DSIL)$, si les vecteurs $a(t_1), \ldots, a(t_n)$, sont linéairement indépendants.

— **Ensemble de base :** $\delta = \{t_1, \ldots, t_n\} \subset T$ qui correspond à la solution de base $\{\delta, y\}$, est dite ensemble de base.

— **Matrice de base :** soit la matrice $A_{n\times n}$ définit comme suit :

$$A(t_1, \ldots, t_n) = \begin{pmatrix} a_1(t_1) & \cdots & a_1(t_n) \\ \vdots & \ddots & \vdots \\ a_n(t_1) & \cdots & a_n(t_n) \end{pmatrix}$$

On dit que la matrice A est de base avec $rangA = n$, si $\{\delta, y\}$ est une solution de base du problème dual $(DSIL)$.

Le système d'équations $Ay = C$, admet une solution unique donnée par : $y = A^{-1}C$ avec $y \in \mathbb{R}_+^n$.

4.3.2 L'existence d'une solution de base optimale du problème dual [16]

Propriété 4.3.1. *Parmi les vecteurs $a(t)$ avec $t \in T$, il existe un sous ensemble de n vecteurs qui sont linéairement indépendants.*

Propriété 4.3.2. *Supposons que le problème dual $(DSIL)$ est solvable, alors il existe une solution $\{\delta, y\}$ tel que $\delta = \{t_1, \ldots, t_q\}$, $y_j > 0$ et les vecteurs $a(t_1), \ldots, a(t_q)$ sont linéairement indépendants pour $j = 1, \ldots, q$ et $q \leq n$.*

Théorème 4.3.1 (Existence d'une solution de base optimale [16])**.** *Si le problème dual $(DSIL)$ est solvable, alors il existe toujours, parmi les solutions de base, une solution de base optimale.*

Théorème 4.3.2 ([16])**.** *$\{\delta, y\}$ et x sont des solutions optimales pour les problème dual $(DSIL)$ et primal $(PSIL)$ respectivement si et seulement si les équations suivantes sont satisfaites*

$$\sum_{i=1}^{n} a_i(t)x_i \geq b(t), \quad t \in T \tag{4.1}$$

$$\sum_{j=1}^{n} a_i(t_j)y_j = c_i, \quad i = 1, \ldots, n \tag{4.2}$$

$$\begin{cases} x_i \left(\sum_{j=1}^{n} a_i(t_j)y_j - c_i \right) = 0, & i = 1, \ldots, n \\ y_j \left(\sum_{i=1}^{n} a_i(t_j)x_i - b(t_j) \right) = 0, & j = 1, \ldots, n \end{cases} \tag{4.3}$$

Les équations (4.3) sont appelées les relations des écarts complémentaires, c'est à dire, Si $\{\delta, y\}$ est une solution du problème dual $(DSIL)$, alors, en utilisant les équations (4.3), on peut déterminer la solution x du problème primal $(PSIL)$, et vice-versa.

4.3.3 Résolution d'un problème semi-infini linéaire par la discrétisation

La discrétisation [108, 18, 109, 110] consiste à utiliser des règles pour trouver un sous-ensemble fini de points à partir d'un ensemble infini de points, puis à utiliser des méthodes connues pour résoudre le problème (par exemple la méthode du simplexe pour le cas d'un problème linéaire).

Formule d'un problème semi-infini linéaire discrétisé

Soit le problème semi-infini linéaire suivant :

$$(PSIL)\begin{cases} \min \sum_{i=1}^{n} c_i x_i \\ \sum_{i=1}^{n} a_i(t) x_i \geq b(t), \quad t \in T \\ x \in \mathbb{R}^n \end{cases}$$

Le problème semi-infini linéaire discrétisé est défini comme suit :

$$(PSIL_m)\begin{cases} \min \sum_{i=1}^{n} c_i x_i \\ \sum_{i=1}^{n} a_i(t_j) x_i \geq b(t_j), \quad t_j \in T_m \\ x \in \mathbb{R}^n, T_m = \{t_1, \ldots, t_m\} \subset T \end{cases}$$

Principe de la discrétisation

Il s'agit de résoudre le problème discrétisé $(PSIL_m)$ en utilisant, à chaque itération, la méthode du simplexe.

Soit $x^m = (x_1^m, \ldots, x_n^m)$ une solution du problème discrétisé $(PSIL_m)$, et on pose $D = \{x \in \mathbb{R}^n, \sum_{i=1}^{n} a_i(t) x_i \leq b(t), t \in T\}$, et $D_m = \left\{x \in \mathbb{R}^n, \sum_{i=1}^{n} a_i(t_j) x_i \leq b(t_j), t_j \in T_m\right\}$. C'est clair que $D \subset D_m$ puisque $T_m \subset T$.

Test d'optimalité

en utilisant la solution x^m du problème discrétisé $(PSIL_m)$, on résout le problème suivant :

$$\min_{t \in T} \left(\sum_{i=1}^{n} a_i(t) x_i^m - b(t) \right),$$

on a deux cas :

premier cas : si

$$\min_{t\in T}\left(\sum_{i=1}^{n} a_i(t)x_i^m - b(t)\right) \geq 0.$$

Alors $x^m \in D$. On note par $x^* = (x_1^*, \ldots, x_n^*)$ la solution optimale du problème $(PSIL)$, on a alors :

$$\sum_{i=1}^{n} c_i x_i^* \leq \sum_{i=1}^{n} c_i x_i^m. \tag{4.4}$$

On a aussi $D \subset D_m$, alors :

$$\sum_{i=1}^{n} c_i x_i^m \leq \sum_{i=1}^{n} c_i x_i^*. \tag{4.5}$$

D'après les équations (4.4) et (4.5), x^m est la solution optimale du problème semi-infini discrétisé $(PSIL_m)$.

deuxième cas : si

$$\min_{t\in T}\left(\sum_{i=1}^{n} a_i(t)x_i^m - b(t)\right) < 0,$$

alors $x^m \notin D$. Dans ce cas, il faut trouver une autre solution réalisable et optimale du problème semi-infini discrétisé.

Trouver $t' \in T$ tel que :

$$\min_{t\in T}\left(\sum_{i=1}^{n} a_i(t)x_i^m - b(t)\right) = \left(\sum_{i=1}^{n} a_i(t')x_i^m - b(t')\right),$$

où $T_{m+1} = T_m \cup \{t'\}$, avec $t' \notin T_m$. On passe à l'itération suivante $(m+1)$, on aura le problème discrétisé suivant :

$$(PSIL_{m+1}) \begin{cases} \min \sum_{i=1}^{n} c_i x_i \\ \sum_{i=1}^{n} a_i(t_j)x_i \geq b(t_j), \quad t_j \in T_{m+1} \\ x \in \mathbb{R}^n, T_{m+1} = \{t_1, \ldots, t_{m+1}\} \subset T \end{cases}$$

Soit $D_{m+1} = \left\{x \in \mathbb{R}^n, \sum_{i=1}^{n} a_i(tj)x_i \geq b(t_j), t_j \in T_{m+1}\right\}$, tel que $D \subset D_{m+1}$.

Soit $x^{m+1} = (x_1^{m+1}, \ldots, x_n^{m+1})$, une solution du problème discrétisé $(PSIL_{m+1})$.

On répète la procédure de vérification d'optimalité de la solution x^{m+1} pour le problème discrétisé jusqu'à arriver à la solution optimale x^* du problème semi-infini linéaire $(PSIL)$.

Théorème 4.3.3 ([111]). *Soit x^m une suite des solutions des problèmes discrétisés $(PSIL_m)$, avec $m = 1, \ldots$ et $T_1 \subset T_2 \subset \cdots \subset T$. Alors, la suite x^m converge vers la solution optimale x^* du problème semi-infini linéaire $(PSIL)$.*

$$\lim_{m \to \infty} x^m = x^*.$$

Algorithme de discrétisation pour le problème semi-infini linéaire

Algorithme 10 Algorithme de discrétisation

1. **Initialisation :** Poser $m = 1$, et choisir $T_m \subset T$.
2. Résoudre le problème discrétisé suivant :

$$(PSIL_m) \begin{cases} \min \sum_{i=1}^{n} c_i x_i \\ \sum_{i=1}^{n} a_i(t_j) x_i \geq b(t_j), \quad t_j \in T_m \\ x \in \mathbb{R}^n, T_m = \{t_1, \ldots, t_m\} \subset T \end{cases}$$

   x^m est la solution du problème $(PSIL_m)$.
3. Calculer la valeur suivante :

$$L = \min_{t \in T} \left(\sum_{i=1}^{n} a_i(t) x_i^m - b(t) \right).$$

4. **Si** $L \geq 0$, alors **STOP**. x^m est la solution optimale du problème semi-infini linéaire $(PSIL)$. **Sinon** poser $T_{m+1} = T_m \cup \{t'\}$, avec $t' \in T$ et $t' \notin T_m$. Poser $m = m + 1$ et aller à l'étape 2.

4.3.4 La méthode du simplexe pour la programmation semi-infinie linéaire [17]

Par la suite, nous présentons le schéma de la méthode du simplexe pour la résolution de problème semi-infini linéaire ($PSIL$). Soit l'ensemble de base $\delta = \{t_1, \ldots, t_n\} \in T$. En suivant les étapes suivantes, on cherche à déterminer l'ensemble $\delta' \subset T$. Les étape de la méthode sont données comme suit :

Etape 1

Résoudre le système d'équations linéaires suivant pour trouver y :

$A(t_1, \ldots, t_n)y = C$.

Etape 2

Déterminer la solution unique x du système suivant :

$A^T(t_1, \ldots, t_n)x = b(t_1, \ldots, t_n)$.

On a deux cas :

premier cas Si

$$\sum_{i=1}^{n} a_i(t)x_i \geq b(t), \quad \forall t \in T,$$

alors x et $\{\delta, y\}$ sont les solution optimales du problèmes primal ($PSIL$) et dual ($DSIL$) respectivement, et on arrête.

deuxième cas S'il existe au moins une contrainte du problème primal ($PSIL$), qui n'est pas vérifié pour la solution x. Alors x et $\{\delta, y\}$, ne sont pas des solution optimales pour les problèmes ($PSIL$) et ($DSIL$) respectivement. Dans ce cas, on cherche une autre solution x' et $\{\delta', y'\}$ qui sera meilleure que x et $\{\delta, y\}$. La nouvelle solution sera déterminé de tel sort que

$$\sum_{j=1}^{n} b(t_j)y_j < \sum_{j=1}^{n} b(t'_j)y'_j,$$

l'ensemble $\delta = \{t_1, \ldots, t_n\}$ est remplacé par l'ensemble $\delta' = \{t_1, \ldots, t_{i-1}, t', t_{i+1}, \ldots, t_n\}$. La question qui se pose c'est quel élément t_i de δ qui va être remplacé par l'élément t' ?

Etape 3

Chercher t' tel que

$$\sum_{i=1}^{n} a_i(t')x_i \geq b(t').$$

Etape 4

Résoudre le système des équations linéaires suivant :

$$A(t_1, \ldots, t_n)d = a(t'),$$

où $a(t_1, \ldots, t_n)$ est la combinaison linéaire des vecteurs $a(t_j)$ avec $t_j \in \delta$. d est la solution unique du système précédent.

Etape 5

On a deux cas :

premier cas Si $(d_j \leq 0, j = 1, \ldots, n)$, alors les problèmes $(PSIL)$ et $(DSIL)$ n'admettent pas de solutions, et on s'arrête.

deuxième cas S'il existe $d_j > 0, j = 1, \ldots, n$, alors il faut choisir un indice $i \in \{1, \ldots, n\}$ tel que :

$$i = arg\min_j \left\{ \frac{y_j}{d_j}, d_j > 0 \right\}.$$

Donc, l'élément t_i doit être remplacé par l'élément t', dans l'ensemble δ' et on écrit

$$\delta' = \left(\delta \cup \left\{t'\right\}\right) - \left\{t_i\right\}.$$

Il reste à montrer que l'ensemble δ' est un ensemble de base, c'est à dire les vecteurs $a(t_1), \ldots, a(t'), \ldots, a(t_n)$ sont linéairement indépendants.

Supposons que $i = 1$ alors $\delta' = \left\{t', t_2, \ldots, t_n\right\}$. On sait déjà que les vecteurs $a(t_2), \ldots, a(t_n)$

sont linéairement dépendants, donc

$$\exists \alpha_j \in \mathbb{R}, \quad a(t') = \sum_{j=2}^{n} a(t_j)\alpha_j.$$

D'autre coté on a

$$a(t') = \sum_{j=1}^{n} a(t_j)d_j.$$

En comparant les deux dernières formules de $a(t')$, on aura

$$\begin{cases} d_1 = 0, \\ d_j = \alpha_j, \quad j = 2, \ldots, n. \end{cases}$$

Ce qui donne une contradiction avec $d_i > 0$. Alors les vecteurs $a(t'), a(t_2), \ldots, a(t_n)$ sont linéairement indépendants et $\delta' = \{t', t_2, \ldots, t_n\}$ est un ensemble de base.

4.4 Problème d'optimisation semi-infinie non linéaire

Nous avons vu précédemment le problème d'optimisation semi-infinie linéaire et la technique de résolution de ce genre de problème. Dans cette section, nous allons traiter le problème d'optimisation semi-infinie non linéaire [108, 18, 109, 110].

On considère le problème suivant :

$$(PSINL) \begin{cases} \min f(x) \\ g(x,t) \leq 0, \quad \forall t \in T \\ x \in \mathbb{R}^n \end{cases}$$

où $f : \mathbb{R}^n \mapsto \mathbb{R}$ est une fonction C^2-différentiable. T est le produit cartésien de N intervalles de $\mathbb{R}$. $g : \mathbb{R}^n \times T \mapsto \mathbb{R}$ est une fonction C^2-différentiable.

On note par $\nabla f_j(x) = \dfrac{\partial f(x)}{\partial x_j}$, la dérivée partielle de la fonction f par rapport à x_j pour $j = 1, \ldots, n$.

On note par $\nabla g_j(x,t) = \dfrac{\partial g(x,t)}{\partial x_j}$, la dérivée partielle de la fonction g par rapport à x_j pour $j = 1, \ldots, n$.

Définition 4.4.1. *Soit x^0 une solution réalisable du problème semi-inifini non linéaire $(PSINL)$. Le point x^0 est un point stationnaire si :*

1. *Il existe $t_i^0 \in T, \quad i = 1, \ldots, q$, tel que :*

$$g(x^0, t_i^0) = 0.$$

2. *Il existe $\lambda_i^0 \in \mathbb{R}_+, \quad i = 1, \ldots, q$, tel que :*

$$\nabla f_j(x^0) + \sum_{i=1}^{q} \lambda_i^0 \nabla g_j(x^0, t_i^0) = 0, \quad j = 1, \ldots, n.$$

Définition 4.4.2. *Soit x^0 une solution réalisable du problème semi-infini non linéaire $(PSINL)$. Soient les variables t_i^0 pour $i = 1, \ldots, q$ tel que $g(x^0, t_i^0) = 0$. Les variables t_i^0 s'appellent les points stationnaires de la fonction $g(x,t)$ au voisinage du point x^0.*

Définition 4.4.3. *On note par $E(x)$ l'ensemble des points stationnaires (maximum) de la fonction g dans T, où : $g(x,t) \geq -\epsilon$ avec $\epsilon > 0$.*

$$E(x) = \left\{ t_j \in T : g(x,t_j) = 0, \ \text{ou } \epsilon > 0, g(x,t_j) \geq -\epsilon \right\}.$$

Définition 4.4.4. *Soit $x \in \mathbb{R}^n$ et $t = \{t_1, \ldots, t_N\} \in E(x) \subset \mathbb{R}^N$. On a :*

$\delta = (\delta_1, \ldots, \delta_l)$ sont des composantes de t sur la frontière de T avec $l \leq N$.

$$\nabla_1 g(x,t) = \left(\frac{\partial g(x,t)}{\partial t_1}, \ldots, \frac{\partial g(x,t)}{\partial t_l} \right).$$

$$\nabla_2 g(x,t) = \left(\frac{\partial g(x,t)}{\partial t_{l+1}}, \ldots, \frac{\partial g(x,t)}{\partial t_N} \right).$$

La matrice carrée d'ordre $(N-l)$, $\nabla_2^2 g(x,t)$ *est donnée comme suit :*

$$\nabla_2^2 g(x,t) = \begin{pmatrix} \dfrac{\partial g(x,t)}{\partial^2 t_{l+1}} & \cdots & \dfrac{\partial g(x,t)}{\partial t_{l+1}\partial t_N} \\ \vdots & \ddots & \vdots \\ \dfrac{\partial g(x,t)}{\partial t_N \partial t_{l+1}} & \cdots & \dfrac{\partial g(x,t)}{\partial^2 t_N} \end{pmatrix}.$$

Définition 4.4.5. *Soit* S *un ensemble non vide de* $\mathbb{R}^n$*, où*

1. *Il existe* $t \in T$ *tel que* $\nabla_2 g(x,t) = 0$*.*
2. *A chaque point* t *vérifiant* $\nabla_2 g(x,t) = 0$*, correspond au maximum local de la fonction* g*, d'où la matrice* $\nabla_2^2 g(x,t)$ *est définie negative.*

4.4.1 Caractérisation des solutions du problème semi-infini non linéaire

Avant de discuter la méthode de résolution d'un problème semi-infini non linéaire, une définition de ce qui constitue précisément une solution est nécessaire. Comme le problème fini non linéaire (PNL), les solutions sont caractérisées comme des points où les conditions de Karush-Kuhn-Tucker (KKT) sont vérifiées. Les conditions nécessaires d'optimalité du première ordre (KKT) pour le problème semi-infini non linéaire $(PSINL)$, sont données dans le théorème suivant.

Théorème 4.4.1 (Condition nécessaire d'optimalité du première ordre (KKT) [112])**.** *Soit* $x^* \in S$ *un minimum local du problème semi-infini* $(PSINL)$*. Alors, pour* $q \leq N$ *il existe* q *maximums globaux* t_i^* *de la fonction* $g(x^*,t)$*, et un vecteur des multiplicateurs de lagrange* $\lambda^* = (\lambda_1^*, \ldots, \lambda_q^*)$*, tel que :*

$$\left\{ \begin{array}{l} \nabla f(x^*) + \sum_{i=1}^{q} \lambda_i^* \nabla_x g(x^*, t_i^*) = 0 \\ \qquad g(x^*, t_i^*) = 0, \ \textit{et} \ \lambda_i^* \geq 0, \qquad \forall i = 1, \ldots, q. \end{array} \right\}.$$

La forme de la condition nécessaire d'optimalité de première ordre, est importante car seulement un nombre fini de points dans T sont requis. Par conséquent, de nombreuses méthodes de résolution basées sur le remplacement de la contrainte semi-infinie par un ensemble fini de contraintes ordinaires.

On définit la fonction de lagrange comme suit :

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

avec $h_i = g(x, t_i)$ pour $i = 1, \ldots, q$.

Théorème 4.4.2 (Condition nécessaire d'optimalité du deuxième ordre [112]). *Soit $x^* \in S$ un minimum local du problème semi-infini $(PSINL)$. Soit un vecteur des multiplicateurs de lagrange λ^*, tel que la condition de (KKT) est vérifiée, alors on a :*

$$\left\{ \begin{array}{ll} d' \nabla^2 L(x^*, \lambda^*) d \geq 0, & d \neq 0 \\ d' \nabla h_i(x^*) = 0, & i = 1, \ldots, q. \end{array} \right\}.$$

4.4.2 Méthodes de résolution de problème semi-infini non linéaire [18]

a) Méthode localement convergente

Pour le problème non linéaire fini (PNL), une méthode localement convergente est obtenue en appliquant la méthode de Newton aux conditions d'optimalité de premier ordre (KKT). Une approche similaire pour le problème semi-infini non linéaire $(PSINL)$ peut être adoptée, bien que le système d'équations auquel la méthode de Newton doit être appliquée soit beaucoup plus grand : il se compose des conditions d'optimalité de premier ordre (KKT) pour le problème semi-infini $(PSINL)$ ainsi que des conditions d'optimalité de premier ordre (KKT) pour chaque maximum global t_i^*. Si on suppose que T est défini par un

nombre fini des contraintes continues différentiables suivantes :

$$b_j(t) \leq 0, \quad j = 1, \ldots, k_b. \tag{4.6}$$

Comme il faudra trouver des maximum de la fonction g par rapport à t, on suppose que les contraintes (4.6) satisfont la condition de qualification des contraintes. Alors le système à résoudre devient comme suit :

$$\left\{ \begin{array}{cc} \nabla f(x) + \sum_{i=1}^{q} \lambda_i \nabla_x g(x, t_i) = 0 & \\ g(x, t_i) = 0 & i = 1, \ldots, q \\ \nabla_t g(x, t_i) + \sum_{j \in B_T(i)} \eta_{ij} \nabla b_j(t_i) & i = 1, \ldots, q \\ b_j(t_i) = 0 & i = 1, \ldots, q, \quad \forall j \in B_T(i) \end{array} \right\} \tag{4.7}$$

où $B_T(i)$ est l'ensemble des indices des contraintes de T sur lesquelles se trouve t_i^*. les variables η_{ij} sont les multiplicateurs de lagrange pour les contraintes $b_j(t)$ qui sont actives au maximum global t_i^*. On suppose une stricte complémentarité pour le problème $(PSINL)$ au point x^* et pour le problème de maximisation locale à chaque t_i. Les conditions suffisantes de second ordre sont supposées êtres vérifiées pour x^*, et aussi pour chaque maximum global t_i^*. Ces conditions garantissent que l'étape de Newton pour le système d'équations (4.7) est bien définie pour x,λ,η_{ij} et t_i suffisamment proche de leurs valeurs optimales.

Malheureusement, en général, cette méthode ne converge que localement. En raison des excellentes propriétés de convergence asymptotique des méthodes localement convergentes, de nombreuses méthodes globales ont été créées en modifiant ces méthodes locales.

b) Méthode de discrétisation [17, 18, 110, 113]

Comme on a vu dans le cas linéaire, l'idée principale d'une méthode de discrétisation est de remplacer l'ensemble infini T par un sous ensemble fini T_0 de T. Avec cet modification le problème semi-infini non linéaire $(PSINL)$ est transformé à un problème fini non linéaire (PNL) qui peut être résolu par des méthodes existantes dans la programmation non linéaire.

Bien sûr la solution obtenue ne sera pas forcément une solution du problème $(PSINL)$. En pratique une succession des problèmes (PNL) sont résolus. Chaque problème (PNL) est différent à un autre dans le choix de l'ensemble fini T_0. En général, cette procédure s'arrête à certain t dans $T - T_0$. Sous certaines conditions, les solutions de cette séquence des (PNL), convergent vers une solution du problème semi-infini non linéaire $(PSINL)$.

Hittich et Gramlich [17, 113] ont développé des algorithmes de discrétisation pour les problèmes semi-infinis linéaires, et pour les problème semi-infinis quadratiques convexes respectivement. Ces algorithmes sont facilement généralisés aux problèmes semi-infinis non linéaires. Ces algorithmes résolvent un problème semi-infini non linéaire en résolvant une succession des problèmes non linéaires (PNL). Le $i$ème (PNL) dans la séquence consiste à minimiser la fonction objectif sous la contrainte $g(x,t)$ sur l'ensemble fini $\{T_i\}$, où $T_i \subseteq T$. Typiquement, chaque ensemble $\{T_i\}$ est choisi comme des intersections d'une grille. Comme les grilles de discrétisation deviennent plus fines, le nombre de contraintes dans chaque (PNL) peut devenir très grand. Pour éviter les calculs excessifs, certaines des contraintes obtenues de la discrétisation sont omises du (PNL). Le (PNL) est alors résolu avec un ensemble de contraintes réduites. Si la solution obtenue est satisfait toutes les contraintes omises, alors le (PNL) original a été résolu. Si au moins une contrainte omise n'est pas satisfaite, alors au moins une des contraintes omises est replacée dans l'ensemble des contraintes du (PNL), et le processus est répété.

La construction de grille pour l'ensemble fini $\{T_i\}$, se fait en deux étapes principales :

discrétisation par force brute : une règle arithmétique explicite est utilisée pour définir la grille, par exemple pour $T = [0,1]$, on a :

$$T_i = \left\{ \frac{1}{n} : n = 1, \ldots, i \right\}, \quad i \geq 1.$$

La génération de la grille de façon adaptative. Beaucoup de ces règles (adaptative) ont été développées dans la littérature de la programmation semi-infinie [114, 115]. Par la suite nous

allons décrire l'algorithme de discrétisation et la règle de définition de grille énoncée dans [114], les règles dans les autres références sont de nature similaire. Blankenship et Falk dans [114] proposent un algorithme composé de quatre étapes :

Algorithme 11 Algorithme général de discrétisation

1. **Initialisation :** Poser $i = 1$, et choisir $T_i \subset T$.
2. Résoudre le $i$ème problème (PNL) suivant :
$$\begin{cases} \min f(x) \\ g(x,t) \leq 0, \quad \forall t \in T_i \end{cases}$$
x^i est la solution de ce problème.
3. Résoudre par rapport à t, le problème suivant :
$$\begin{cases} \max g(x^i,t) \\ \forall t \in T. \end{cases}$$
t^i est la solution de ce problème.
4. **la règle de définition de grille : Si** $g(x^i,t^i) \leq 0$, alors **STOP** (la solution globale est trouvée). **Sinon** mettre à jours la grille en posant $T_{i+1} = T_i \cup t^i$, et aller à l'étape 2.

4.5 Conclusion

Dans ce chapitre nous avons traité les aspects principaux de l'optimisation semi-infinie. Nous avons présenté un problème pratique (système de contrôle MIMO) de l'optimisation semi-infinie. Nous avons présenté quelques méthodes de résolution pour les problèmes d'optimisation semi-infinie linéaire et pour les problèmes d'optimisation semi-infinie non linéaire.

CONCLUSION GÉNÉRALE ET PERSPECTIVES

Au cours des dernières décennies, l'essor de l'optimisation a été favorisé par le développement des moyens de calcul sur machine et leurs nombreuses applications dans divers domaines de la vie.

La plupart des problèmes d'optimisation globale sont NP-difficiles. La détection du minimum global parmi plusieurs minima locaux est un grand défi. Les chercheurs ont rencontré des difficultés considérables dans le traitement des problèmes multidimensionnels et non convexes, définis par des fonctions non régulières. Bien qu'il existe plusieurs résultats théoriques sur la caractérisation des minimums globaux, leur implémentation numérique reste dans la plupart des cas une question très complexe.

La convexité est l'outil principal et important dans l'optimisation globale. L'optimisation convexe est une sous discipline de l'optimisation, dans laquelle la fonction objectif à minimiser est convexe sur l'ensemble admissible convexe. L'optimisation convexe est facilitée par l'existence d'une grande variété de théorèmes et d'algorithmes. Par conséquence, les problèmes d'optimisation convexe sont plus facile à résoudre que les problèmes d'optimisation non convexe, bien qu'il puissent être (l'optimisation convexe) NP-difficile en général.

Plusieurs chercheurs ont développé des méthodes visant à atteindre l'optimum global. Cependant chacune possède des défaux. Certaines méthodes sont relativement efficaces, mais seulement dans des cas très précises.

Dans cette thèse on s'est intéressé aux méthodes d'optimisation mathématique, plus précisément, les méthodes d'optimisation globale basées sur la construction de sous-estimateurs convexes, par exemple la méthode αBB. L'efficacité de ces méthodes a une forte et directe relation avec les valeurs propres des matrices hessiennes des fonctions qui définissent le problème à résoudre. Cependant, les matrices associées aux fonctions C^2-différentiables sur l'intervalle domaine de variables, sont des matrices d'intervalles symétriques réelles. Par conséquence, le calcul des valeurs propres des matrices d'intervalles est, en générale, un

problème NP-difficile.

On a proposé une nouvelle méthode pour le calcul des bornes inférieures et supérieures sur les valeurs propres des matrices d'intervalles symétriques réelles. La méthode proposée est simple, avec une complexité polynomiale et facile dans l'implémentation sur la machine. La méthode proposée peut produire des bornes, en général, améliorantes par rapport aux bornes produites par les méthodes existantes dans la littérature.

On a utilisé notre nouvelle méthode dans l'algorithme αBB pour produire des bornes inférieures sur les valeurs propres des matrices hessiennes d'intervalles symétriques, afin de construire des sous-estimateurs plus efficaces dans la résolution des problèmes d'optimisation globale ainsi que les problèmes d'optimisation semi-infinie.

En perspective, beaucoup de questions peuvent être envisagées :

— En optimisation globale, le temps de calcul nécessaire pour atteindre le minimum global représente le critère d'efficacité le plus commun des méthodes. Ce temps dans la plupart des cas, est lié au nombre d'itérations de l'algorithme de résolution, par exemple l'algorithme αBB, qui dépend de l'efficacité du sous-estimateur utilisé. De ce fait, il est naturel de construire une nouvelle fonction vérifiant les conditions d'un sous-estimateur et qu'elle soit plus efficace par rapport au sous-estimateur développé dans la méthode αBB.

— Amélioration de notre contribution publiée pour les matrices hessiennes d'intervalles non symétriques (dans le cas des fonctions non déffirentiables).

BIBLIOGRAPHIE

[1] Hoang Tuy, Tuy Hoang, Tuy Hoang, Viêt-nam Mathématicien, Tuy Hoang, and Vietnam Mathematician. *Convex analysis and global optimization*. Springer, 1998.

[2] Richard J Hillestad and Stephen E Jacobsen. Reverse convex programming. *Applied Mathematics and Optimization*, 6 :63–78, 1980.

[3] Reiner Horst, Thai Q Phong, and Nguyen V Thoai. On solving general reverse convex programming problems by a sequence of linear programs and line searches. *Annals of Operations Research*, 25(1) :1–17, 1990.

[4] Reiner Horst and Hoang Tuy. *Global optimization : Deterministic approaches*. Springer Science & Business Media, 2013.

[5] Hiroshi Konno, Phan Thien Thach, and Hoang Tuy. *Optimization on low rank nonconvex structures*, volume 15. Springer Science & Business Media, 2013.

[6] Hoang Tuy. Convex programs with an additional reverse convex constraint. *Journal of Optimization Theory and Applications*, 52(3) :463–486, 1987.

[7] Brian Borchers and John E Mitchell. An improved branch and bound algorithm for mixed integer nonlinear programs. *Computers & Operations Research*, 21(4) :359–367, 1994.

[8] Christoph H Lampert, Matthew B Blaschko, and Thomas Hofmann. Efficient subwindow search : A branch and bound framework for object localization. *IEEE transactions on pattern analysis and machine intelligence*, 31(12) :2129–2142, 2009.

[9] Frédéric Messine. *Méthodes d'optimisation globale basées sur l'analyse d'intervalle pour la résolution de problèmes avec contraintes*. PhD thesis, Toulouse, INPT, 1997.

[10] Ioannis P Androulakis, Costas D Maranas, and Christodoulos A Floudas. αbb : A global optimization method for general constrained nonconvex problems. *Journal of Global Optimization*, 7(4) :337–363, 1995.

[11] Claire S Adjiman, Stefan Dallwig, Christodoulos A Floudas, and Arnold Neumaier. A global optimization method, αbb, for general twice-differentiable constrained nlps—i. theoretical advances. *Computers & Chemical Engineering*, 22(9) :1137–1158, 1998.

[12] Claire S Adjiman, Ioannis P Androulakis, and Christodoulos A Floudas. A global optimization method, αbb, for general twice-differentiable constrained nlps—ii. implementation and computational results. *Computers & chemical engineering*, 22(9) :1159–1179, 1998.

[13] Mohand Ouanes, Hoai An Le Thi, Trong Phuc Nguyen, and Ahmed Zidna. New quadratic lower bound for multivariate functions in global optimization. *Mathematics and Computers in Simulation*, 109 :197–211, 2015.

[14] Jonathan M Borwein. Semi-infinite programming duality : how special is it ? In *Semi-Infinite Programming and Applications : An International Symposium, Austin, Texas, September 8–10, 1981*, pages 10–36. Springer, 1983.

[15] Rembert Reemtsen and Jan-J Rückmann. *Semi-infinite programming*, volume 25. Springer Science & Business Media, 1998.

[16] Klaus Glashoff and S-A Gustafson. *Linear optimization and approximation : an introduction to the theoretical analysis and numerical treatment of semi-infinite programs*, volume 45. Springer Science & Business Media, 2012.

[17] Rainer Hettich. An implementation of a discretization method for semi-infinite programming. *Mathematical Programming*, 34 :354–361, 1986.

[18] Christopher John Price. Non-linear semi-infinite programming. 1992.

[19] Anthony V Fiacco and Kenneth O Kortanek. *Semi-Infinite Programming and Applications : An International Symposium Austin, Texas, September 8–10, 1981*, volume 215. Springer Science & Business Media, 2012.

[20] Michel Bierlaire. *Introduction à l'optimisation différentiable*. PPUR presses polytechniques, 2006.

[21] Michel Minoux. *Programmation mathématique. Théorie et algorithmes*. Lavoisier, 2008.

[22] Alain Billionnet. *Optimisation Discrète, de la modélisation à la résolution par des logiciels de programmation mathématique*. 2007.

[23] Jean-Christophe Culioli. *Introduction à l'optimisation*. Ellipses, 2012.

[24] Mohamed Aidene, IL Vorob'ev, and Brahim Oukacha. Algorithm for solving a linear optimal control problem with minimax performance index. *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki*, 45(10) :1756–1765, 2005.

[25] Abdelkrim Keraghel. *Étude adaptative et comparative des principales variantes dans l'algorithme de Karmarkar*. PhD thesis, Institut National Polytechnique de Grenoble-INPG, 1989.

[26] Kahina Louadj. *Résolution de problèmes paramétrés de contrôle optimal*. PhD thesis, Universite Mouloud MAMMERI Tizi-Ouzou, 2012.

[27] Scott Kirkpatrick. Dg jr., and mp vecchi. *Optimization by simmulated annealing. science*, 220(4598) :671–680, 1983.

[28] DE Goldberg and JH Holland. Genetic algorithms and machine learning. 3 (2) : 95-99, 1988.

[29] Costas D Maranas and Christodoulos A Floudas. A deterministic global optimization approach for molecular structure determination. *The Journal of chemical physics*, 100(2) :1247–1261, 1994.

[30] Hoai An Le Thi and Mohand Ouanes. Convex quadratic underestimation and branch and bound for univariate global optimization with one nonconvex constraint. *Rairo-Operations Research*, 40(3) :285–302, 2006.

[31] Reiner Horst and Panos M Pardalos. *Handbook of global optimization*, volume 2. Springer Science & Business Media, 2013.

[32] Andreas Antoniou and Wu-Sheng Lu. *Practical optimization : algorithms and engineering applications*, volume 19. Springer, 2007.

[33] Jacques Teghem. *Programmation linéaire*. Paris, 2003.

[34] EML Beale. On quadratic proramming. *Naval Research Logistics Quarterly*, 6(3) :227–243, 1959.

[35] Laurent Schwartz and Khelifa Zizi. Théorie des ensembles et topologie. *Hermann*, 1991.

[36] Stephen P Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[37] Juan Enrique Martínez-Legaz. On weierstrass extreme value theorem. *Optimization letters*, 8(1) :391–393, 2014.

[38] Djaouida Guettal. *Efficacité et fiabilité des méthodes utilisant l'approche branch-and-bound 'pour l'optimisation globale non convexe*. PhD thesis, 2018.

[39] Philip Wolfe. The simplex method for quadratic programming. *Econometrica : Journal of the Econometric Society*, pages 382–398, 1959.

[40] Edwin KP Chong, Wu-Sheng Lu, and Stanislaw H Żak. *An introduction to optimization*. John Wiley & Sons, 2023.

[41] Cornelis Roos, Tamás Terlaky, and J-Ph Vial. Interior point methods for linear optimization. 2005.

[42] Stephen J Wright. *Primal-dual interior-point methods*. SIAM, 1997.

[43] Ilan Adler, Mauricio GC Resende, Geraldo Veiga, and Narendra Karmarkar. An implementation of karmarkar's algorithm for linear programming. *Mathematical programming*, 44 :297–335, 1989.

[44] II Dikin. Iterative solution of problems of linear and quadratic programming. In *Doklady Akademii Nauk*, volume 174, pages 747–748. Russian Academy of Sciences, 1967.

[45] Renato DC Monteiro, Ilan Adler, and Mauricio GC Resende. A polynomial-time primal-dual affine scaling algorithm for linear and convex quadratic programming and its power series extension. *Mathematics of Operations Research*, 15(2) :191–214, 1990.

[46] Philip Wolfe. A duality theorem for non-linear programming. *Quarterly of applied mathematics*, 19(3) :239–244, 1961.

[47] Mokhtar S Bazaraa, Hanif D Sherali, and Chitharanjan M Shetty. *Nonlinear programming : theory and algorithms*. John wiley & sons, 2013.

[48] Mira Al Kharboutly. *Résolution d'un problème quadratique non convexe avec contraintes mixtes par les techniques de l'optimisation DC*. PhD thesis, Normandie, 2018.

[49] Anthony V Fiacco and Garth P McCormick. *Nonlinear programming : sequential unconstrained minimization techniques*. SIAM, 1990.

[50] Michael JD Powell. Algorithms for nonlinear constraints that use lagrangian functions. *Mathematical programming*, 14 :224–248, 1978.

[51] James M Ortega and Werner C Rheinboldt. *Iterative solution of nonlinear equations in several variables*. SIAM, 2000.

[52] Zhibin Zhu, Weidong Zhang, and Zhenjie Geng. A feasible sqp method for nonlinear programming. *Applied mathematics and computation*, 215(11) :3956–3969, 2010.

[53] Philip E Gill, Walter Murray, and Margaret H Wright. *Practical optimization*. SIAM, 2019.

[54] HC Wu. Duality theory for optimization problems with interval-valued objective functions. *Journal of Optimization Theory and Applications*, 144 :615–628, 2010.

[55] B Scheurer. Existence et approximation de points selles pour certains problèmes non linéaires. *RAIRO. Analyse numérique*, 11(4) :369–400, 1977.

[56] Khosrow Moshirvaziri and Mahyar A Amouzegar. A cutting plane algorithm for linear reverse convex programs. *Annals of Operations Research*, 105 :201–212, 2001.

[57] Thuan Nguyen Quang. *Approches locales et globales basées sur la programmation DC et DCA pour des problèmes combinatoires en variables mixtes 0-1 : applications à la planification opérationnelle.(Local and global approaches based on DC programming and DCA for mixed 0-1 combinatorial problems : applications to operational planning)*. PhD thesis, University of Metz, France, 2010.

[58] Viet Nga Pham. *Programmation DC et DCA pour l'optimisation non convexe/optimisation globale en variables mixtes entières : Codes et Applications*. PhD thesis, Rouen, INSA, 2013.

[59] Elliot Ward Cheney and Allen A Goldstein. Newton's method for convex programming and tchebycheff approximation. *Numerische Mathematik*, 1(1) :253–268, 1959.

[60] Karla Leigh Hoffman. A method for globally minimizing concave functions over convex sets. *mathematical Programming*, 20 :22–32, 1981.

[61] Hoang Tuy. *On outer approximation methods for solving concave minimization problems*. Universität Bremen. Fachbereiche Mathematik/Informatik, Elektrotechnik ..., 1983.

[62] David Q Mayne and Elijah Polak. Outer approximation algorithm for nondifferentiable optimization problems. *Journal of Optimization Theory and Applications*, 42(1) :19–30, 1984.

[63] EM Landis. On functions representable as the difference of two convex functions. In *Doklady Akad. Nauk SSSR (NS)*, volume 80, pages 9–11, 1951.

[64] Philip Hartman. On functions representable as a difference of convex functions. 1959.

[65] Pham Dinh Tao and LT Hoai An. Convex analysis approach to dc programming : theory, algorithms and applications. *Acta mathematica vietnamica*, 22(1) :289–355, 1997.

[66] Mahdi Moeini. *La programmation DC et DCA pour l'optimisation de portefeuille*. PhD thesis, Metz, 2008.

[67] HA Le Thi. Contribution à l'optimisation non convexe et l'optimisation globale : : Théorie. algorithmes et applications. habilitation à diriger des recherches. *National Institute for Applied Sciences, Rouen*, 1997.

[68] Le Thi Hoai An and Pham Dinh Tao. Solving a class of linearly constrained indefinite quadratic problems by dc algorithms. *Journal of global optimization*, 11 :253–285, 1997.

[69] JB Hiriart Urruty. Conditions nécessaires et suffisantes d'optimalité globale en optimisation de différences de deux fonctions convexes. *CR Acad. Sci. Paris Ser. I Math*, 309 :459–462, 1989.

[70] John F Toland. Duality in nonconvex optimization. *Journal of Mathematical Analysis and Applications*, 66(2) :399–415, 1978.

[71] T Pham Dinh. Calcul du maximum d'une forme quadratique définie positive sur la boule unité de la norme du max. Technical report, Technical Report, Grenoble, 1976.

[72] Pham Dinh Tao et al. Algorithms for solving a class of nonconvex optimization problems. methods of subgradients. In *North-Holland Mathematics Studies*, volume 129, pages 249–271. Elsevier, 1986.

[73] Helmut Ratschek and Jon Rokne. *New computer methods for global optimization*. Halsted Press, 1988.

[74] Ioannis G Akrotirianakis and Christodoulos A Floudas. A new class of improved convex underestimators for twice continuously differentiable constrained nlps. *Journal of Global Optimization*, 30 :367–390, 2004.

[75] Ioannis G Akrotirianakis and Christodoulos A Floudas. Computational experience with a new class of convex underestimators : Box-constrained nlp problems. *Journal of Global Optimization*, 29 :249–264, 2004.

[76] Abhishek Kumar, Guohua Wu, Mostafa Z Ali, Rammohan Mallipeddi, Ponnuthurai Nagaratnam Suganthan, and Swagatam Das. A test-suite of non-convex constrained optimization problems from the real-world and some baseline results. *Swarm and Evolutionary Computation*, 56 :100693, 2020.

[77] CA Floudas and AR Ciric. Strategies for overcoming uncertainties in heat exchanger network synthesis. *Computers & chemical engineering*, 13(10) :1133–1152, 1989.

[78] Christodoulos A Floudas and Panos M Pardalos. *A collection of test problems for constrained global optimization algorithms*. Springer, 1990.

[79] Hong S Ryoo and Nikolaos V Sahinidis. Global optimization of nonconvex nlps and minlps with applications in process design. *Computers & Chemical Engineering*, 19(5) :551–566, 1995.

[80] Ashok D Belegundu and Jasbir S Arora. A study of mathematical programmingmethods for structural optimization. part ii : Numerical results. *International journal for numerical methods in engineering*, 21(9) :1601–1623, 1985.

[81] E Sandgren. Nonlinear integer and discrete programming in mechanical design optimization. 1990.

[82] Horst Nowacki. Optimization in pre-contract ship design. 1973.

[83] Maria L Bergamini, Nicolás J Scenna, and Pio A Aguirre. Global optimal structures of heat exchanger networks by piecewise relaxation. *Industrial & engineering chemistry research*, 46(6) :1752–1763, 2007.

[84] Co A Haverly. Studies of the behavior of recursion for the pooling problem. *Acm sigmap bulletin*, (25) :19–28, 1978.

[85] VASILIOS MANOUSIOUTHAKIS and DENNIS SOURLAS. A global optimization approach to rationally constrained rational programming. *Chemical Engineering Communications*, 115(1) :127–147, 1992.

[86] William M Kahan. A more complete interval arithmetic. *Lecture notes for an engineering summer course in numerical analysis, University of Michigan*, 4 :31, 1968.

[87] Richard J Hanson. Interval arithmetic as a closed arithmetic system on a computer. *Jet Propulsion Laboratory Report*, 197, 1968.

[88] Eldon Hansen and G William Walster. *Global optimization using interval analysis : revised and expanded*, volume 264. CRC Press, 2003.

[89] Djamel Zerrouki and Mohand Ouanes. Bounds on eigenvalues of real symmetric interval matrices for αbb method in global optimization. *Yugoslav Journal of Operations Research*, 34(1) :73–92, 2023.

[90] Arkadii Nemirovskii. Several np-hard problems arising in robust stability analysis. *Mathematics of Control, Signals and Systems*, 6 :99–105, 1993.

[91] Jiří Rohn. Checking positive definiteness or stability of symmetric interval matrices is np-hard. *Commentationes Mathematicae Universitatis Carolinae*, 35(4) :795–797, 1994.

[92] Vladik Kreinovich, Anatoly V Lakeyev, Jiří Rohn, and PT Kahl. *Computational complexity and feasibility of data processing and interval computations*, volume 10. Springer Science & Business Media, 2013.

[93] Chrysanthos E Gounaris and Christodoulos A Floudas. Tight convex underestimators for c^2-continuous problems : Ii. multivariate functions. *Journal of Global Optimization*, 42(1) :69–89, 2008.

[94] Milan Hladík. On the efficient gerschgorin inclusion usage in the global optimization αBB method. *Journal of Global Optimization*, 61(2) :235–253, 2015.

[95] Quan Yuan, Zhiqing He, and Huinan Leng. An evolution strategy method for computing eigenvalue bounds of interval matrices. *Applied mathematics and computation*, 196(1) :257–265, 2008.

[96] Huinan Leng and Zhiqing He. Computing eigenvalue bounds of structures with uncertain-but-non-random parameters by a method based on perturbation theory. *Communications in Numerical Methods in Engineering*, 23(11) :973–982, 2007.

[97] Milan Hladík, David Daney, and Elias Tsigaridas. Bounds on real eigenvalues and singular values of interval matrices. *SIAM Journal on Matrix Analysis and Applications*, 31(4) :2116–2129, 2010.

[98] Huinan Leng, Zhiqing He, and Quan Yuan. Computing bounds to real eigenvalues of real-interval matrices. *International journal for numerical methods in engineering*, 74(4) :523–530, 2008.

[99] Jirı Rohn. A handbook of results on interval linear problems. *Internet text available at http ://www. cs. cas. cz/rohn/handbook*, 2005.

[100] Lubomir Kolev and Simona Petrakieva. Assessing the stability of linear time-invariant continuous interval dynamic systems. *IEEE Transactions on Automatic Control*, 50(3) :393–397, 2005.

[101] Jiri Rohn. Stability of interval matrices : the real eigenvalue case. *IEEE transactions on automatic control*, 37(10) :1604–1605, 1992.

[102] Lubomir Kolev. Eigenvalue range determination for interval and parametric matrices. *International Journal of Circuit Theory and Applications*, 38(10) :1027–1061, 2010.

[103] Gene H Gloub and Charles F Van Loan. Matrix computations. *Johns Hopkins Universtiy Press, 3rd edtion*, 1996.

[104] Roger A Horn and Charles R Johnson. *Matrix analysis*. Cambridge university press, 2012.

[105] James Hardy Wilkinson. *The algebraic eigenvalue problem*. Oxford University Press, Inc., 1988.

[106] David Hertz. The extreme eigenvalues and stability of real symmetric interval matrices. *IEEE Transactions on automatic control*, 37(4) :532–535, 1992.

[107] Momin Jamil and Xin-She Yang. A literature survey of benchmark functions for global optimisation problems. *International Journal of Mathematical Modelling and Numerical Optimisation*, 4(2) :150–194, 2013.

[108] Panayiotis Lemonidis. *Global optimization algorithms for semi-infinite and generalized semi-infinite programs*. PhD thesis, Massachusetts Institute of Technology, 2008.

[109] Rainer Hettich and Kenneth O Kortanek. Semi-infinite programming : theory, methods, and applications. *SIAM review*, 35(3) :380–429, 1993.

[110] Alexander Shapiro. Semi-infinite programming, duality, discretization and optimality conditions. *Optimization*, 58(2) :133–161, 2009.

[111] Elijah Polak and Limin He. Rate-preserving discretization strategies for semi-infinite programming and optimal control. *SIAM journal on control and optimization*, 30(3) :548–572, 1992.

[112] Marco López and Georg Still. Semi-infinite programming. *European journal of operational research*, 180(2) :491–518, 2007.

[113] Rainer Hettich and Guenther Gramlich. A note on an implementation of a method for quadratic semi-infinite programming. *Mathematical programming*, 46 :249–254, 1990.

[114] Jerry W Blankenship and James E Falk. Infinitely constrained optimization problems. *Journal of Optimization Theory and Applications*, 19 :261–281, 1976.

[115] C Gonzaga, Elijah Polak, and R Trahan. An improved algorithm for optimization problems with functional inequality constraints. *IEEE Transactions on Automatic Control*, 25(1) :49–54, 1980.

ملخص

لقد أصبح التحسين الشامل نظاما هاما في العالم الحديث الذي نعيش فيه، يواجه منافسة دولية مفرطة ومتنامية. في عملنا، درسنا المفاهيم والجوانب الأساسية المتعلقة بالتحسين الشامل (التحدب، عدم التحدب، طرق الاستبانة العددية، الفرع والتقييد والطريقة αBB إلخ). لقد قدمنا وطبقنا طريقة جديدة لحساب الحدود على القيم الذاتية لمصفوفات الفاصل المتماثلة للدوال الغير محدبة

كلمات مفتاحية : التحسين الشامل، التحدب، عدم التحدب، طريقة الفرع و التقييد، مصفوفة هيسيان، مضفوفة الفواصل، حدود على القيم الذاتية، التحسين الشبه لا نهائي.

Abstract

Global optimization has become an important discipline in the modern world in which we live, faced with excessive and growing international competition. In our work, we studied the fundamental concepts and aspects concerning global optimization (convexity, non convexity, numerical resolution methods, Branch and Bound, αBB method ...). We have presented and applied a new approach to compute bounds on eigenvalues of hessian symmetric interval matrices for nonconvex functions.

Key words : Global optimization, convexity, nonconvexity, Branch and Bound, hessian matrix, interval matrix, bounds on eigenvalues, semi-infinite optimization.

Résumé

L'optimisation globale est devenue une discipline importante dans le monde moderne dans lequel nous vivons, confronté à une concurrence internationale excessive et croissante. Dans notre travail, nous avons étudié les concepts et les aspects fondamentaux concernant l'optimisation globale (convexité, non convexité, méthodes de résolutions numériques, Branch and Bound, méthode αBB ...). Nous avons présenté et appliqué une nouvelle approche pour calculer des bornes sur les valeurs propres des matrices hessiennes d'intervalles symétriques pour les fonctions non convexes.

Mots clés : Optimisation globale, convexité, non convexité, Branch and Bound, matrice hessienne, matrice d'intervalles, bornes sur les valeurs propres, optimisation semi-infinie.