MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE MOULOUD MAMMERI, TIZI-OUZOU.

FACULTE DES SCIENCES

DEPARTEMENT MATHEMATIQUES

MEMOIRE MASTER

OPTION : RECHERCHE OPERATIONNELLE

Présenté par

M^me^ FERHAOUI KARIMA

Sujet :

Optimisation Globale basée sur l'Analyse d'Intervalle : Applications et Comparaisons

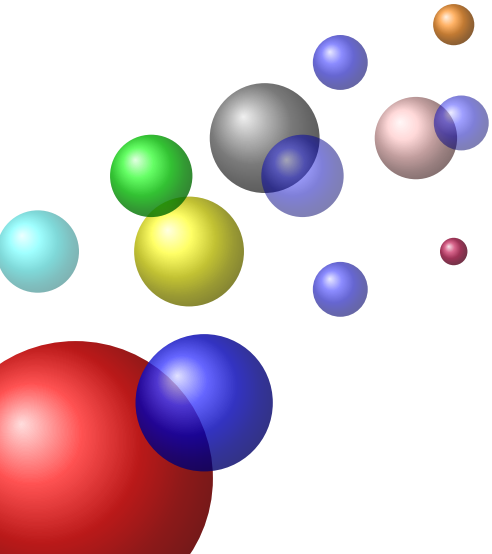
Devant le jury d'examen composé de :

M^r^ Ouanes Mohand	Professeur	UMMTO	Président
M^r^ Amirou Ahmed	Maître de conférence B	UMMTO	Examinateur
M^me^ Goumeziane Lynda	Maître assistante A	UMMTO	Rapporteur

Soutenue le :

Remerciement

J'adresse mes remerciements les plus sincères à Madame Goumeziane qui a bien voulu, avec une volonté exemplaire, diriger le travail que je présente dans ce mémoire, d'avoir eu la patience nécessaire pour m'apprendre beaucoup de choses.
Je tiens à remercier également tout les membres du jury que m'en fait honneur de leurs présences, et pour avoir jugé mon travail et ce de manière objective.
Je n'oublie pas de remercier tous ceux qui m'ont aidé d'une manière ou d'une autre.



Dédicaces

Premièrement Dieu merci de m'avoir permis de mener à terme ce projet, je dédie ce travail :
A celle qui a bercé mes rêves ma très chère mère ;
A celui qui sans son aide je ne serais peut être pas à ce niveau, mon père ;
A mon marie Omar ;
A la mémoire de mon grand-père ; que Dieu te garde dans son vaste Paradies ;
A tout membre de ma grande famille, notamment ; mes deux grands-mères, mon grand-père et ma belle mère ;
A ma chère sœur Yasmina ;
A ma chère petite sœur Khadidja ; que Dieu, le tout puissant, la préserve et la procure santé et longue vie ;
Mes chères enfants Mohammed El Amine et Ahmed El Amine ;
Mon unique et petit frère adoré Mohammed ;
A tout ceux que j'aime et ceux qui m'aiment de loin ou de prés ;
Je remercie tous ceux et celles qui m'ont aidé à réaliser ce travail.

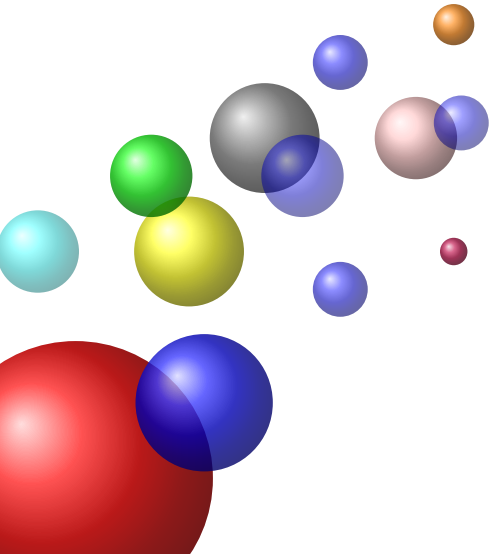


Table des matières

Table des figures

Liste des tableaux

Introduction

Tout les jours durant notre vie, nous cherchons à optimiser notre temps de travail, le trajet que nous aurons à parcourir pour nous rendre quelque part,...etc. Nous cherchons tous une meilleure solution aux problèmes qui jalonnent notre existence. De manière générale, l'optimisation va consiste à trouver cette meilleure solution.

L'optimisation est devenue une discipline incontournable du monde moderne dans lequel nous vivons, car celui-ci est sujet à une compétition internationale excessive et croissante. Dés lors, il devient nécessaire, voire vital pour les entreprises comme pour les gouvernements, de maximiser ou de minimiser toutes sortes de choses ; par exemple, maximiser les profits tout en minimisant les pertes, améliorer si possible de façon optimale certains processus de fabrication ou les fonctionnalités de certains objets ou produits.

Durant les 20 dernières années, le domaine de la recherche sur "l'optimisation global" s'est considérablement enrichi grâce, notamment, à l'accroissement de la puissance de calcul des ordinateurs. Ces progrès ont permis de résoudre des problèmes auparavant insolubles. Il y a deux types de méthodes d'optimisation globale : méthodes stochastiques qui permettent de trouver l'optimum global avec une probabilité proche de 1, et méthodes déterministes qui trouvent, avec certitude, une approximation de l'optimum global.

L'optimisation va consister à rechercher dans le domaine initial une solution qui maximise ou minimise une fonction objectif, pour un domaine continu et discret, on distingue classiquement deux type d'optimisation :

- ***L'optimisation locale :*** *elle recherche une solution qui est la meilleure localement, c'est-à-dire que dans son voisinage aucune solution n'est meilleure qu'elle. Cette solution est appelée un optimum local.*

- ***L'optimisation globale :*** *elle recherche quant à elle la meilleure solution du domaine en entier, c'est-à-dire que dans tout le domaine il n'existe aucune solution qui lui soit meilleure tout en respectant les contraintes. Cette solution est appelée l'optimum global.*

L'optimum globale est aussi une solution locale. En revanche il est bien plus épineux de trouver l'optimum global, et la différence entre la solution globale et une solution locale est bien souvent significative. L'optimum global dans nombreux problèmes est la solution mathématique.

Le premier chapitre de ce mémoire est bibliographique, où on exposera des notions

élémentaires de l'optimisation globale ainsi il propose un état de l'art des différentes méthodes d'optimisation globale. Parmi les méthodes déterministes, les méthodes de recouvrement ont la réputation d'être efficaces en dimension 1, mais la généralisation des mêmes idées au cas dimensionnel crée d'énormes difficultés d'ordre pratique.

Contrairement aux méthodes déterministes, les méthodes stochastiques ont l'avantage d'être facilement imprésentables sur machine. Certaines parmi elles ne requièrent comme celle donnée du problème que les évaluations de la fonction à optimiser. Cependant, ces méthodes ne sont pas efficaces elles sont utilisées lorsqu'il n'est pas possible de résoudre le problème avec une méthode déterministe.

Le deuxième chapitre expliquera les notions de l'analyse d'intervalle et différent notations utilisées dans ce mémoire. Après avoir présenté l'arithmétique d'intervalle ainsi que les diverses notations et définitions utilisées. Cette arithmétique offre de nouvelles méthodes de résolution des problèmes d'optimisation globale avec ou sans contraintes. En optimisation sans contrainte, elle donne des inclusions de la fonction à optimiser grâce à des fonctions construites à l'aide de cette arithmétique appelée fonction d'inclusion. Quand à l'optimisation globale avec contrainte, l'arithmétique d'intervalles permet de résoudre les systèmes d'équations qui résultent des contraintes.

Le troisième est consacré sur l'application de l'analyse d'intervalle sur l'optimisation globale, au début on donne un algorithme d'optimisation globale de type branch and bound. Par la suite on donne les domaines d'utilisations aux méthodes des intervalles qui nous allons illustrer l'application de l'arithmétique des intervalles par quelques exemples. A la fin de ce chapitre, on présente deux méthodes d'amélioration des bornes : La méthode des développements en série de Taylor et la méthode de Baumann qui est une amélioration de la méthode de Taylor.

Pour le chapitre 4 on passe à la programmation des méthodes d'amélioration de la borne inférieure tout en introduisant une extension du logiciel MATLAB pour le calcul par intervalles qui est "INTLAB", et faire la comparaison selon la complexité de la fonction objectif.

Chapitre 1

Les différentes méthodes et approches d'optimisation globale

1.1 Généralités sur l'optimisation globale

L'optimisation est une branche des mathématiques et de l'informatique en tant que disciplines, cherchant à modéliser, à analyser et à résoudre analytiquement ou numériquement les problèmes qui consistent à déterminer quelles sont la ou les solution(s) satisfaisant un objectif quantitatif tout en respectant d'éventuelle contraintes.
***Optimiser** :**(Déf.du Larousse)** rendre optimal, donner à quelque chose les meilleures conditions d'utilisation, de fonctionnement ou de rendement au regard de certaines circonstances.*

Le problème que l'on étudie ici est celui de la recherche du minimum globale d'une fonction réelle f de n variables réelles $x_1, x_2, ..., x_n$. De tels problèmes apparaissent fréquemment dans les applications.

Soit $f : X \to \mathbb{R}$, pour tout $x \in X$,$x = (x_1, x_2, ..., x_n)$

$$\min_{x \in S} f(x) \tag{1.1}$$

où :

- *f : La fonction-objectif*
  (ou fonction coût ou critère ou fonction économique)
- *S :Le domaine réalisable (ou admissible)*

  Le but est de trouver x^* tel que $f(x^*) \leq f(x)$, pour tout $x \in S$.

1.1.1 Modélisation et ingrédients d'une Optimisation

Le processus de formalisation du problème d'identification des objectifs et des variables à optimiser est appelé modélisation qui est une étape très importante ou il faut choisir un modèle pas trop coûteux à évaluer.

Les ingrédients d'une optimisation, issue d'un processus de modélisation sont représentés par [25] :

- ***Les variables ou bien les paramètres d'optimisation*** *: l'optimum est cherché dans un ensemble U plongé dans un espace vectoriel V. Très souvent on pose $V = \mathbb{R}^n$,*
- ***La fonction de coût (objectif)*** *$f : U \to \mathbb{R}$: une mesure quantitative de qualité d'un candidat à l'optimum,*
- ***Les contraintes sur les variables d'optimisation*** *: sont représentés soit par une application $\overrightarrow{c} : V \to \mathbb{R}^m$ dite "vecteur de contraintes", ou par le choix de l'ensemble des solutions admissibles $U \subseteq V$ (feasible region).*

1.1.2 Formulation mathématique

Supposons que $V = \mathbb{R}^n$. Le problème de minimisation dans $\mathbb{R}^n$ avec contraintes s'écrit : trouver $\overrightarrow{u}^* \in \mathbb{R}^n$ tel que :

$$f(u^*) = \min_{u \in \mathbb{R}} f(u) \quad où \quad \begin{cases} c_i(\overrightarrow{u}^*) = 0, & i \in \varepsilon; \\ c_j(\overrightarrow{u}^*) \geq 0, & j \in I. \end{cases} \tag{1.2}$$

Ici, ε et I sont des sous-ensembles d'indices $\{1, 2, ..., m\}$ pour dénoter les contraintes d'égalité, resp. d'inégalité. On peut alors, de manière équivalente, choisir l'ensemble des solutions admissibles.

$$U = \{\overrightarrow{v} \in \mathbb{R}^n : (c_i(\overrightarrow{v}) = 0, \ \forall i \in \varepsilon) \wedge (c_j(\overrightarrow{v}) \geq 0, \ \forall j \in I)\} \tag{1.3}$$

Définition 1.1.1 *Soit $V \subseteq \mathbb{R}^n$, $f : V \to \mathbb{R}$. Le point $x_0 \in V$ s'appelle minimum globale si seulment si :*

$$f(x_0) \leq f(x) \quad \forall x \in V. \tag{1.4}$$

Remarque 1.1.2 *Trouver les minima globaux serait le meilleur résultat qu'on puisse espérer. Malheureusement, pour une fonction coût f générale, c'est une tâche difficile . Très souvent, on n'a pas la vision globale de f, on sait seulement l'évaluer ainsi que ses dérivées sur quelques points x_k, k = 1, 2,.... La plupart des algorithmes trouvent seulement un minimum local.*

Définition 1.1.3 *On dit que f atteint en x_0 un "maximum local" s'il existe un I de la forme $]x_0 - \varepsilon, x_0 + \varepsilon[$ tel que :*

$$f(x) \leq f(x_0), \quad \forall x \in I \tag{1.5}$$

1.1.3 Quelques types de problèmes d'Optimisation

On peut classifier les problèmes d'optimisation selon la forme particuliére de leurs données, comme suit :

1. ***Problèmes de programmation linéaire ou quadratique :***

   On parle des problèmes de programmation linéaire, quand aussi bien $f : V \mapsto \mathbb{R}$ et $C : V \mapsto \mathbb{R}^m$ sont des applications linéaires dans V .

   D'autre part si $f : V \mapsto \mathbb{R}$ est une fonction quadratique de x, alors on parle des problèmes de programmation quadratique.

Exemple 1.1.4 *(Probléme du transport).*
La production de m usines fournit n boutiques à travers le pays. Chaque usine peut produire au maximum a_i, $i = \{1, ..., m\}$ tonnes du produit par semaine et chaque boutique écoule b_j , $j = \{1, ..., n\}$ tonnes du produit par semaine. Comment organiser la livraison du produit des usines aux boutiques de manière la plus rentable, quand on sait que le coût de transport d'une tonne de produit de l'usine i au boutique j coute c_{ij}
On définit x_{ij}, la quantité du produit livré (par semaine) de l'usine i à la boutique j. La facture de livraison hebdomadaire de boutique j est de $\sum_{i=1}^{m} c_{ij}x_{ij}$ et la facture totale est de

$$f(\overrightarrow{u}) = \sum_{j=1}^{n}\sum_{i=1}^{m} c_{ij}x_{ij}$$

Ici, $\overrightarrow{u}$ dénote en fait la matrice x_{ij}. La contrainte de production se formalise par :

$$\sum_{j=1}^{m} x_{ij} \leq a_i, \quad i = \{1, ..., m\}$$

la contrainte d'avoir la marchandise en boutique au moment de la vente donne

$$\sum_{i=1}^{n} x_{ij} \geq b_j, \quad j = \{1, ..., n\}$$

et le flux positif de marchandise des usines aux boutiques donne

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

On a alors un problème typique de la programmation linéaire.

2. ***Optimisation continue et optimisation discrète :***

Dans l'optimisation discrète, en plus des contraintes $\overrightarrow{c}$ on a aussi la contrainte que $\overrightarrow{u}$ soit entière. On peut diviser les problèmes discrets en problèmes combinatoires (résolution de sodoku, par exemple), et en problèmes de programmation en nombres entiers.

En générale, les problèmes d'optimisation continue sont plus faciles à résoudre que les problèmes discrets.

3. ***Optimisation sans et avec contraintes :***

Si l'ensemble admissible U couvre tout l'espace vectoriel, on parle d'optimisation sans contraintes.
En générale, ce type de problème d'optimisation est plus facile à résoudre.
L'optimisation avec contraintes limite l'ensemble admissible U à un sous-ensemble de V. Une possible technique de résolution de ce type de problèmes est en utilisant la technique de pénalisation : on ajoute à la fonction coût $f : V \to \mathbb{R}$ (qu'on veut minimiser) un terme non-négatif, qui devient grand pour $\overrightarrow{u}$ appartient pas à U. Une autre technique consiste dans l'introduction des multiplicateurs de Lagrange.

4. ***Optimisation globale et locale :***

Les méthodes d'optimisation les plus efficaces procèdent par une amélioration itérative d'un candidat $\overrightarrow{u} \in U$ à l'optimum. Elles cherchent dans le voisinage de $\overrightarrow{u} \in U$ un "meilleur candidat" $\overrightarrow{v} \in U$ pour lequel on a :

$$f(\overrightarrow{v}) \leq f(\overrightarrow{u})$$

Cette technique peut néanmoins assurer seulement la convergence vers un point $\overrightarrow{u}$ qui est localement optimal. Pour avoir la certitude que ce point est un optimum global, soit on doit connaître le comportement global de la fonction coût f, soit on doit évaluer $f(\overrightarrow{u})$ en beaucoup de points de U (ce qui est très coûteux), soit on dispose des propriétés de f qui assurent qu'un minimum local est au même temps un minimum global (par expl. la convexité de $f(\overrightarrow{u})$).

1.2 Les différentes méthodes et approches d'Optimisation Globale

De nos jours, afin de résoudre des problèmes d'optimisation globale avec contraintes, de nombreuses stratégies algorithmiques s'avèrent disponibles. Pour guider le choix de la meilleure stratégie à utiliser, il est nécessaire d'observer : la taille du problème, les propriétés de la fonction objectif et des contraintes (continuité, différentiabilité, linéarité, convexité,...), ainsi que le temps disponible pour résoudre le problème.

Voici une liste non exhaustive des différentes approches :
Il est certainement impossible de citer toutes les méthodes d'optimisation globale, ni de les classer car certaines méthodes peuvent appartenir à plusieurs classes en même temps.

Cependant, on distingue deux grandes classes de méthodes d'optimisation globale :

i/ *Méthodes stochastiques donnant le minimum global de f avec une probabilité proche de 1,*

ii/ *Méthodes déterministes convergeant sûrment vers le minimum global de f.*

1.2.1 Méthodes Stochatiques

Les méthodes probabilistes se sont imposées dans plusieurs branches de la science. Le développement rapide de l'automatisme et de l'informatique a donné naissance à de nouvelles méthodes de la théorie des probabilités appliquées à partir des années 50. Il s'agit de la modélisation stochastique et des algorithmes stochastiques qui constituent un terrain où les problèmes ouverts restent nombreux et d'un grand intérêt. Le développement théorique des algorithmes stochastiques est actuellement bien avancé.

Quoique, généralement, ils ne sont pas encore en mesure de satisfaire les besoins pratiques, leurs applications ne cessent de s'améliorer et de s'étendre. Ils sont utilisés dans les cas suivants :

i/ *La fonction objectif n'est pas connue analytiquement mais connue à travers des évaluations numériques observables à une perturbation aléatoire prés.*

ii/ *La fonction objectif est compliquée on peut l'approcher, en moyenne, par une quantité aléatoire.*

iii/ *La fonction objectif est connue analytiquement mais elle est trop oscillante ou l'ensemble faisable est de grande dimension ($n > 100$).*

Il y a des méthodes stochastiques basées sur la recherche aléatoire pur, c'est à dire sur l'évaluation de la fonction f dans un ensemble fini de points de X tirés de manière aléatoire et indépendante. Ces algorithmes sont surtout utilisés lorsque f n'est pas donnée analytiquement ou lorsqu'aucune information sur f n'est pas disponible à part sa valeur en tout point de X ; c'est ce qu'on appelle "Situation Black Box" (boite noire).

Certaines méthodes stochastiques font intervenir des procédures déterministes à stratégie locale comme la méthode multi-start qui est entièrement basée sur les techniques de recherche d'optima locaux. Elle consiste à faire des descentes locales multiples à partir des points tirés aléatoirement, indépendamment et uniformément de X.

Un autre type d'algorithmes stochastiques connu sous le nom de "algorithme de la ligne aléatoire" consiste à chercher le minimum global à l'aide d'une méthode déterministe appliquée sur des courbes générées aléatoirement. Par conséquent, l'algorithme permet de transformer un problème d'optimisation globale multidimensionnel à un problème unidimensionnel.

D'autres méthodes stochastiques sont basées sur la partition de l'ensemble faisable X en un nombre fini de sous ensembles puis l'application d'un algorithme d'optimisation globale (déterministe ou stochastique) successivement sur ces sous ensembles tirés au hasard selon une loi concentrée sur l'ensemble des éléments de la subdivision.

Citons aussi parmi les techniques basées sur la recherche aléatoire pure les algorithmes markoviens qui représentent une modification de ces techniques.

En effet, on inclut dans ces techniques des éléments adaptifs qui permettent de simuler une loi dépendante du point précèdent x_k et la valeur $f(x_k)$ et qui génèrent x_{k+1}. Un des algorithmes markoviens les plus connus est l'algorithme de "recuit simulé".

Une deuxième classe de méthodes stochastiques est la classe des méthodes basées sur la résolution de l'équation différentielle stochastique. Ces méthodes consistent à chercher les minima globaux d'une fonction f définie sur $\mathbb{R}^n$ et assez régulière en considérant l'équation différentielle d'Itô :

$$d\Phi(t) = -\nabla f(\Phi(t))dt + \varepsilon(t)d\beta(t) \tag{1.6}$$

où $\beta(t)$ est un mouvement brownien standard multidimentionnel et $\varepsilon(t)$ est une fonction qui tend vers 0 lorsque t tend vers ∞.

Pour une fonction $\varepsilon(t^)$ tendant lentement vers 0 lorsque t tend vers ∞, la solution de l'équation d'Itô est une diffusion récurrente dont la loi stationnaire se concentre sur l'ensemble des minimiseurs globaux de f . La méthode tente donc d'obtenir un minimiseur global de f en regardant quand $t \to \infty$ une trajectoire de la diffusion calculée numériquement.

1.2.1.1 Incovénient des méthodes stochastiques

Les algorithmes stochastiques donnent le minimum global avec une probabilité proche de 1 quand le nombre d'évaluations de $f(x)$ est très grand.

Généralement, ils sont simples à implémenter sur machine mais ils ne sont pas efficaces. Cependant, on fait appel à ces méthodes lorsqu'on ne peut pas résoudre les problèmes d'optimisation globale avec une méthode déterministe.

1.2.2 Méthodes d'Optimisation Globale basées sur les techniques de recherche locale

Dans cette classe, les techniques d'optimisation locale sont d'une grande importance dans la stratégie d'optimisation globale.

Ceci est dû à la construction usuelle des stratégies d'optimisation globale constituée de deux étapes :

- *Etape globale*
- *Etape locale*

Dans l'étape globale, la fonction objectif est évaluée en certains points localisés afin d'atteindre un petit voisinage du minimum global. L'étape locale de l'algorithme d'optimisation globale peut être exprimée sous forme implicite ou explicite, comme elle peut être utilisée une ou plusieurs fois.

Une forme explicite de l'étape locale est présente lorsque'une mauvaise approximation de l'optimum global est obtenue et nous voulons la raffiner ou la rendre plus petite que possible.

Il est surtout intéressant de faire intervenir les stratégies locales dans les algorithmes d'optimisation globale lorsque les dérivées de la fonction objectif sont faciles à calculer .

1.2.3 Méthode multistart

Historiquement, la méthode multistart est l'une des premières méthodes d'optimisation globale qui était largement utilisée. Elle consiste à effectuer des recherches multiples (successives ou simultanées) d'extrema locaux à partir de différents points initiaux qui sont fréquemment choisis parmi les éléments d'une grille uniforme de l'espace faisable X.

Si les minima locaux sont éloignés l'un de l'autre, la méthode multistart est modifiée par l'une des deux façons suivantes :

i/ *On applique une procédure de descente au voisinage de chaque minimiseur local. Cette approche n'est possible que si l'on est capable de choisir les voisinages qui sont des domaines d'attraction des minimiseurss locaux correspondants, ceci est trés difficile et même impossible en général.*

ii/ *Méthode des points candidats : Elle consiste à faire des descentes locales simultanées à partir de plusieurs points initiaux, en joignant les points voisins. Le fait de joindre est équivalent à substituer quelques points à l'un d'entre eux dont la valeur de la fonction objectif est la plus petite.*

1.2.4 Algorithme de la méthode des points condidats

Etape 1 *: Générer uniformément s points $x_1, x_2, ..., x_s$ de X.*

Etape 2 *: Procéder à une descente locale à partir de chaque point intial $x_1, x_2, ..., x_s$. Nous obtenons les points $z_1, z_2, ..., z_N$ $(N \leq s)$.*

Etape 3 *: Appliquer la procédure de "plus proche voisin" :*
i/ Chaque point z_i, $i = 1, 2, ..., N$ est supposé appartenir à un voisinage d'un minimiseur local.
ii/ Vérifier s'il existe des points tels que :$d(z_i, z_j)$ ¡ δ (δ un nombre petit) alors z_i,z_j sont confondus et leurs voisinages unifiés.
iii/ Si $N = 1$ ou $\min_{k,l=1,..,N} d(z_i, z_j) > \delta$, aller à l'étape 4.

Etape 4 *: Aprés avoir obtenu m voisinages de minimiseurs locaux ($m \leq 0$), sélectionner m points représentatifs $x_1, ..., x_m$ de ces voisinages (un critère naturel de ce choix est la valeur de la fonction objectif en ces points). Si $m = 1$ ou si le nombre des voisinages reste constant aller à l'étape 5. Sinon, poser $s = m$ et revenir à l'étape 2.*

Etape 5 *Supposer qu'on est dans un voisinage du minimiseur global parmi les voisinages restants.*

1.2.4.1 Inconvénient de la méthode multistart

Pour assurer l'obtention du minimum global, le nombre des points initiaux doit être beaucoup plus grand que le nombre des minimiseurs locaux qui généralement inconnu.

1.2.5 Les algorithmes de Tunneling

Au cours de la dernière décennie, les algorithmes de Tunneling ont été beaucoup exploités. Leurs idée consiste à une recherche successive d'un nouveau point x_0 dans l'ensemble :

$$U(x^*) = \{x \in X \| f(x) \leq f(x^*)\} \backslash \{x^*\} \tag{1.7}$$

où x^* est un point record obtenu précédemment, puis procéder à une descente à partir du point x_0. L'algorithme de Tunneling est constitué de deux étapes utilisées successivement :

- *Etape de minimisation*
- *Etape de Tunneling*

Dans l'étape de minimisation, nous utilisons une procédure de descente locale pour trouver le minimiseur local x^* de la fonction objectif f ; on l'appelle alors "point record".
Dans l'étape de Tunneling, nous construisons une fonction $T(x)$ appelée "fonction de Tunneling", qui a les propriétés suivantes :

- *$T(x)$ atteint un maximum (peut être local) en x^*.*
- *Les premiéres dérivées de $T(x)$ sont continues (sauf peut être en x^*).*
- *$T(x)$ dépend de f, x^* et d'un nombre de paramètres qui sont automatiquement choisis par l'algorithme.*

Après la construction de la fonction auxiliaire $T(x)$, un point x_0 de l'ensemble $U(x^*)$ est recherché dans l'étape de Tunneling grâce à la minimisation de $T(x)$ sur l'ensemble $U(x^*)$. On obtient ainsi$f(x_0) = f(x^*)$.
Ensuite, nous revenons à l'étape de minimisation de f à partir du point x_0. Le minimiseur local de f obtenu maintenant est un nouveau point record. L'itération ci-dessus est répétée jusqu'à ce que le même point $x^* \in U(x^*)$ est obtenu plusieurs fois de suite .

Parmi les fonctions de Tunneling utilisées on cite la fonction suivante :

$$T(x) = T(x, x^*, \alpha) = \frac{f(x) - f(x^*)}{\parallel x - x^* \parallel^{\alpha}} \tag{1.8}$$

avec $\alpha > 0$ un paramètres fixé. Si nous avons plusieus points record $x_1^*, ...x_l^*$ avec la même valeur de la fonction objectif, nous utilisons la fonction de Tunneling suivante :

$$T(x) = \frac{f(x) - f(x^*)}{\prod_{i=1}^{l} \parallel x_i - x_i^* \parallel^{\alpha_i}} \tag{1.9}$$

où $\alpha_i > 0$ des paramètres fixés.
Pour trouver un point x_0 de $U(x^*)$ dans l'étape de Tunneling, nous devons effectuer une minimisation locale de la fonction de Tunneling $T(x)$ à partir d'un point initial très proche de x^* . Si le minimiseur local de $T(x)$ n'appartient pas à l'ensemble $U(x*)$, nous devons changer les paramètres de la fonction auxiliaire. Si, après le changement des paramètres, nous ne réussissons pas à trouver x_0, alors nous devons changer le point initial de la procédure de minimisation de $T(x)$.

Vilkov et collaborateurs(1975) ont montré que si X est un intervalle, f une fonction continûement différentiable ayant un nombre fini de minimiseurs locaux, la fonction de Tunneling à la forme (1.9) et sa minimisation est réalisée pour tout $\alpha > 0$, alors la méthode de Tunneling définie précédemment converge vers un minimiseur global de la fonction objectif. Dans le cas multidimensionne, un résultat analogue est inexistant.

1.2.5.1 Incovénients de la méthode de Tunneling

L'analyse des résultats numériques montre que la méthode de Tunneling ne peut être considérée comme efficace car elle n'arrive pas toujours à trouver le minimiseur global de f. Les principales difficultés dans la réalisation de la méthode de Tunneling sont les suivantes :

i/ *La fonction auxiliaire $T(x)$ dépend de certains paramètres dont le choix n'obéit pas à un raisonnement mathématique rigoureux ; on se contente de prendre ces paramètres aussi petits que possible.*

ii/ *La minimisation de la fonction $T(x)$ déborde souvent du domaine X ; ce qui nécessite une attention particulière pour ne pas ignorer un minimiseur.($T(x)$ n'est pas stable)*

iii/ *Les critères d'arrêt de la recherche ne garantissent pas l'obtention du minimum global avec la précision désirée.*

1.2.6 Méthodes de transition d'un minimiseur local à un autre

Il est connu que la théorie de l'optimisation locale est beaucoup plus développée que celle de l'optimisation globale. Ainsi, les chercheurs ont pensé à modifier les méthodes locales de telle sorte que la trajectoire de recherche passe d'un minimiseur local à un autre jusqu'à l'obtention du minimum global. La plupart de ce type de méthodes est basée sur l'étude des propriétés des solutions de différentes équations différentielles telles que celle de "Pschenichnij et Marchenk (1976)" où la trajectoire de recherche coïncide avec la trajectoire du mouvement d'une balle sur la surface générée par la fonction objectif.

Les trajectoires de recherche pour une grande classe d'algorithmes sont les approximations discrètes de solutions d'équations différentielles ordinaires du second ordre ayant la forme :

$$\mu(t)x^{"}(t) + \gamma(t)x^{'}(t) = -\nabla f(x(t)), t \geq t_0 \tag{1.10}$$

soumises aux conditions inutiles :

$$\begin{cases} x(t_0) = x_0 \\ x'(t_0) = z_0, \quad z_0 \in \mathbb{R}^n, \quad \| z_0 \| = 1 \end{cases}$$

où $\mu(t)$, $\gamma(t)$ sont des fonctions de temps t.

En mécanique classique, l'équation (1.10) représente la loi de Newton d'une particule de masse $\mu(t)$ dans un potentiel f soumise à une force dissipative $-\gamma(t)x^{'}(t)$.
Au temps initial $t = t_0$, la particule est au point x_0 ayant une direction de mouvement z_0.

Considérons maintenant une autre méthode basée sur la résolution des équations différentielles développée par Branin (1972). Soit f une fonction deux fois continûment différentiable et supposons que nous pouvons évaluer les valeurs du gradient
$g(x) = \nabla f(x)$ et du hessien $H(x) = \nabla^2 f(x)$ pour tout $x \in X$.

Considérons le système d'équations différentielles simultanées :

$$\frac{\partial[g(x(t))]}{\partial t} = sg(x(t)). \tag{1.11}$$

soumis à la condition initiale : $g(x(0)) = g_0$, où s est une constante égale à $+1$ ou -1.
La solution d'un tel système à la forme :

$$g(x(t)) = g_0 e^{st} \tag{1.12}$$

et nous avons $g(x(t)) \to 0$ si $s = +1$ et $t \to +\infty$ ou si $s = -1$ et $t \to +\infty$.

Ceci veut dire que la trajectoire correspondante à une solution tend vers un point stationnaire de f.

La méthode de Branin consiste à la résolution itérative de l'équation (1.12) en alternant le signe de s afin de passer d'un point stationnaire de f à un autre.

1.2.6.1 Inconvénients des algorithmes basée sur la résolution des équations différentielles

i/ *Ils n'existent pas de résultats généraux à propos de leurs convergences vers un optimum global. Donc, on ne peut trouver le minimiseur global que si on est sûr que tous les minima locaux sont obtenus.*

ii/ *La réalisation et l'emploi de ces algorithmes sont compliqués. Notamment, la nécessité d'évaluer les dérivées de la fonction objectif (parfois la hessien aussi). On fait appel aux différences finies pour approximer les dérivées, chose qui n'est pas évidente.*

Cependant, nous devons noter que certains exemples numériques ont démontré que la méthode de Branin demande peu d'évaluations de la fonction objectif comparée aux algorithmes de Tunneling, mais on est obligé d'évaluer les premiéres et secondes dérivées.

1.2.7 Les méthodes de recouvrement

En pratique, la variation de la fonction objectif est généralement bornée. L'ensemble

$$F = Lip(X, L, d) = \{f : X \to \mathbb{R} \| \mid f(x) - f(y) \mid \leq L.d(x, y), \quad \forall x, y \in X\} \tag{1.13}$$

où d est une métrique définie sur X est bien connue. Si nous choisissons des points $x_1, ..., x_N$ de l'ensemble faisable X de telle sorte que les ensembles :

$$X_i \subset \{x \in X, \quad f(x) \geq f(x_i) - \varepsilon\}, \quad \varepsilon > 0$$

forment un recouvrement de X , c-à-d : $X \subset \bigcup_{i=1}^{N} X_i$; alors notre problème de minimisation : $f^* = \min f(x)$ est résolu avec une précision égale à ε.

En effet, si

$f_N^* = \min_{i=1,...,N} f(x_i), \quad x^* = arg\min_{x \in X} f(x)$ *et* $X \subset \bigcup_{i=1}^{N} X_i$ *alors* $\exists i_0 \in \{1, ..., N\}$ *tel que :* $x^* \in X_{i_0}$.

Autrement dit :

$$f(x^*) = f^* \geq f(x_i) - \varepsilon \geq f_N^* - \varepsilon$$

,

donc :

$$f_N^* - f^* \leq \varepsilon$$

.

Les méthodes de sélection des points $x_1, ..., x_N$ ayant la propriété précédente sont appelées "méthodes de recouvrement". Nous devons construire les ensembles X_i pour $i = \{1, ..., N\}$, ayant un volume maximal et rendant la structure des ensembles $X \bigcup_{i=1}^{k} X_i$, $k = 1, 2, ...$, aussi simple que possible.
Si le choix d'un point x_i de l'ensemble $\{x_1, ..., x_N\}$ ne dépend pas de la valeur de la fonction objectif aux point x_j avec $j < i$ alors l'algorithme de minimisation correspondant est appelé algorithme de ***"recouvrement passif"****.*
Par contre, si le choix de x_i fait intervenir les points $x_1, ..., x_{i-1}$ déja sélectionnés ou

leur image $f(x_1), ..., f(x_{i-1})$, alors l'algorithme de minimisation est appelé algorithme de ***"recouvrement itératif"*** .

1.2.8 Algorithme de recouvrement passif

Un algorithme passif est une méthode de minimisation globale basée sur la construction d'une grille $\{x_1, ..., x_N\}$ à l'avance.

On calcule les valeurs $f(x_i), \quad i = 1, ..., N$, puis on prend le point $x_N^* = \min\limits_{i=1,...,N} f(x_i)$ comme approximation de minimiseur global et la valeur $f_N^* = f(x_N^*)$ comme approximation du minimum global de f. Dans les algorithmes de recouvrement, nous pouvons construire le recouvrement de X à l'aide des boules

$$B(x_i, \delta, d) = \{x \in X, d(x, x_i) \leq \delta\}, i = \{1, ..., N\}. \tag{1.14}$$

où $x_i, \quad i = \{1, ..., N\}$ sont les points de la grille, d'une métrique sur X et δ le rayon des boules. Si les boules $B(x_i, \delta, d)$, $i = \{1, ..., N\}$ recouvrent X alors

$$\exists i_0 \in \{1, ..., N\} \text{ tel que } x^* \in B(x_{i_0}, \delta, d)$$

et puisque f est une fonction lipchitzienne, le problème de minimisation est résolu avec une précision $\varepsilon = L\ \delta$.
En effet,

Posons $f_N^* = \min\{f(x_1), ..., f(x_N)\}$, $f^* = \min\limits_{x \in X} f(x)$, $x_N^* = \min\limits_{i=1,...,N} f(x_i)$ et $x^* = arg \min\limits_{x \in X} f(x)$, on a :
$\exists j \in \{1, ..., N\}$ tel que $x^* \in B(x_j, \varepsilon, d)$ et

$$\begin{aligned} |f_N^* - f^*| = |f(x_N^*) - f(x^*)| &\leq |f(x_j) - f(x*)| \\ &\leq L|x_j - x^*| \\ &\leq L\delta = \varepsilon. \end{aligned} \tag{1.15}$$

Les algorithmes de recouvrement passifs sont assez répondus en optimisation globale. Plusieurs études théoriques leurs sont dédiées. Ceci est dû à la simplicité de leur construction et de leur réalisation sur ordinateur.

Ces algorithmes ont un inconvénient essentiel : ils négligent complètement les informations obtenues durant le processus de recherche.

Cet inconvénient rend l'algorithme de recouvrement passif inefficace ; surtout pour les problèmes de grandes dimensions.

1.2.9 Méthodes de recouvrement itératif

Ces algorithmes sont d'un grand intérêt théorique et pratique, donnons d'abord l'idée principale de ces méthodes. soit à optimiser une fonction f avec précision donnée $\delta > 0$. Supposons que la fonction f est évaluée aux points : $x_1, ..., x_k$. Nous appelons la valeur : $f_k^* = f(x_k^*)$ un point record.

Considérons l'ensemble :

$$Z_k = \{x \in X, f_k^* - \varepsilon \leq f(x)\}.$$

Nous avons évidemment

$$f_k^* - \varepsilon \leq \inf f(x)$$

Donc, nous continuons à chercher l'optimum global dans l'ensemble $X \backslash Z_k$.

Cependant, si :alors notre problème est résolu puisque :

$$X \subset Z_k \tag{1.16}$$

alors notre problème est résolu puisque :

$$X \subset Z_k \Rightarrow \forall x \in X : \quad f_k^* - \varepsilon \leq f(x) \qquad \Rightarrow f_k^* - \varepsilon \leq \min_x \in X f(x) = f^*$$

Donc, nous pouvons prendre le point record x_k^* *comme approximation du minimiseur global* x^**, ainsi, la construction d'une méthode de recouvrement sa ramène à la construction d'une suite de points* $x_1, x_2, ...$*,et d'une suite d'ensembles correspondants* $Z_1, Z_2, ...$*, jusqu'à ce que la condition (1.16) soit réalisée.*

Considérons maintenant la méthode de recouvrement dans le cas où la fonction objectif f *appartient à l'espace fonctionnel* $F = Lip(X, L, d)$*, on peut distinguer deux importantes méthodes :*

i/ Méthode de Piyavskii-Shubert

Dans la classe des méthodes de recouvrement on trouve la méthode proposée par Piyavskii et Shubert qui est la plus connue. Elle sélectionne le point x_{k+1} *de telle sorte que*

$$x_{k+1} \in arg \min_{x \in X} (\max_{j=1,k}(f(x_j) - L.d(x, x_j))) \tag{1.17}$$

L'algorithme (1.17) est aussi appelé : méthode de la ligne polygonale parce que le minorant de la fonction f *à la* $k^{iéme}$ *itération :*

$$f^{(k)}(x) = \max_{j=1}^{k}(f(x_j) - L.d(x, x_j)), \quad pour k \geq 1. \tag{1.18}$$

L'extension de l'algorithme de Piyavskii-Shubert au cas multidimensionnel a été proposé par plusieurs chercheurs mais elle est numériquement inefficace car elle engendre d'énormes difficultés pour la mise en oeuvre de l'algorithme .

ii/ Méthode de Brent

Brent (1973) a proposé un algorithme pour la classe des fonctions à une seule variable deux fois continuement differentiables et dont la dérivée seconde est bornée.
C'est une méthode qui consiste à construire une suite croissante de fonctions (f_k)*,* $1 \leq k \leq N$ *, paraboliques par morceaux convergeante vers* f *.*

L'idée de la méthode repose sur le résultat suivant :
Si f est telle que :

$$| \frac{d^2 f(x)}{dx^2} | \leq 2M$$

pour tout $x \in [a, b]$, alors : $\forall x_1, x_2 \in [a, b]$ tels que $x_1 \leq x_2$, la parabole $\varphi(x)$ définie par :

$$\frac{d^2 f(x)}{dx^2} = M$$

Pour tout $x \in [a, b]$, $\varphi(x_1) = f(x_1)$ et $\varphi(x_2) = f(x_2)$
satisfait l'inégalité :

$$\varphi(x) \leq f(x), \quad \forall x \in [x1, x_2]$$

.
Ainsi, pour toute subdivision : $a = x_1, x_2, ..., x_k = b$ de $[a, b]$, on peut construire une fonction F_k continue et parabolique par morceaux sur l'intervalle $[a, b]$. En effet, on construit à partir de chaque couple (x_j, x_{j+1}), $j = \{1, ..., k-1\}$, formé de deux points successifs de la subdivision , une parabole notée $\varphi_{[x_j, x_{j+1}]}$ définie sur l'intervalle $[x_j, x_{j+1}]$, $j = \{1, ..., k-1\}$.

Ensuite, on définit la fonction F_k par :

$$\begin{array}{rccl} F_k : & [a,b] & \rightarrow & \mathbb{R} \\ & x & \rightarrow & F_k(x) = \varphi_{[x_j, x_{j+1}]}(x), \quad x \in [x_j, x_{j+1}], \ j = \{1, ..., k-1\} \end{array} \tag{1.19}$$

Remarque 1.2.1 *L'algorithme unidimensionnel de Brent ne possède pas de généralisation dans le cas multidimensionnel.*

Cependant, il est plus efficace que la méthode de Piyavskii-Shubert. Alors que l'algorithme de Brent est applicable uniquement à la classe des fonctions deux fois continument dérivables et de dérivées secondes bornées ; les autres algorithmes s'appliquent à une classe plus grande.

1.2.9.1 Inconvénient des méthodes de recouvrement

i/ *Ces méthodes sont très difficiles à programmer dans le cas multidimensionnel.*

ii/ *Leur efficacité dépend considérablement des informations données à l'avance sur la fonction objectif f ; c.-à-d. du choix de la classe fonctionnelle F à laquelle appartient f.*

iii/ *Le nombre d'évaluations de la fonction objectif est très grand dans le cas multidimensionnel.*

Toutefois, les méthodes de recouvrement sont théoriquement construites pour le cas $F = Lip(X, L, d)$ où la valeur de constante de Lipchitz L est connue.

Par contre, en pratique, cette constante est généralement inconnue. Cet inconvénient n'est pas un désavantage puisque la constante de Lipchitz peut être estimée durant la recherche.

1.3 Les méthodes déterministes globales

Dans ce type de méthodes, l'aléatoire n'intervient pas, c'est-à-dire que pour résoudre un problème, l'algorithme se comportera toujours de la même façon et donnera toujours la même réponse. Ces algorithmes peuvent se classer en fonction du type de problèmes pouvant être résolu : les programmes linéaires, les problèmes convexes, quadratiques, polynomiaux ou plus généraux.

Ces techniques ont généralement l'avantage de ne pas exiger de points de départ. Mais avant tout, elles fournissent une réponse déterminante sur la qualité des solutions trouvées : l'optimum est-il local ou global ? Quel est le degré de certitude ?...etc.

Cette précision a une importance significative, car il est souvent beaucoup moins coûteux de trouver une solution que de prouver qu'il s'agit bien de l'optimum global.
Les méthodes déterministes passent par deux techniques : la technique de réduction et la technique de partition.

1.3.1 Réduction de la dimension pour les problèmes multi-extrémaux

Les problèmes d'optimisation multi-extrémaux multidimensionnels sont compliqués en mathématiques appliquées, il est donc naturel de penser à transformer ces problèmes à des problèmes plus simples par exemple en réduisant la dimension de ces problèmes. Il y a plusieurs idées pour réduire un problème d'optimisation globale multi-extémal multidimensionnel à un ou plusieurs problèmes d'optimisation ayant une dimension inferieure ; en particulier de dimension une.

Le schéma théorique le plus simple de la réduction de la dimension est celui appelé : "réduction à plusieurs étape". Il est basé sur la représentation

$$\min_{x\in X} f(x) = \min_{0\leq x\leq 1} ... \min_{0\leq x\leq 1} f(x_1, ..., x_n) \tag{1.20}$$

où f est une fonction continue sur le cube unité : $X = [0, 1]^n$.

Ainsi, nous pouvons utiliser ce schéma pour ramener le problème d'optimisation initial sur un cube à plusieurs problèmes d'optimisation globale unidimensionnels, mais le nombre de ces problèmes est généralement très grand.

1.3.2 Méthodes basées sur les techniques de partition et élimination (Branch and Bound)

Beaucoup de problèmes, en particulier les problèmes d'optimisation, l'ensemble de leurs solutions est fini (en tous les cas, il est dénombrable). Il est donc possible, en principe, d'énumérer toutes ces solutions, et ensuite de prendre celle qui nous arrange. L'inconvénient majeur de cette approche est le nombre prohibitif du nombre de solutions : il

n'est guère évident d'effectuer cette énumération.

La méthode branch and bound (procédure par évaluation et séparation progressive) consiste à énumérer ces solutions d'une manière intelligente en ce sens qui, en utilisant certaines propriétés du problème en question, cette technique arrive à éliminer des solutions partielles qui ne mènent pas à la solution que l'en recherche. De ce fait, on arrive souvent à obtenir la solution recherchée en des temps raisonnables. Bien entendu, dans le pire cas, on retombe toujours sur l'élimination explicite de toutes les solutions du problème.

Pour ce faire, cette méthode se dote d'une fonction qui permet de mettre une borne sur certaines solutions pour soit les exclure soit les maintenir comme des solutions potentielles. Bien entendu, la performance de la méthode branch and bound dépend, entre autres, de la qualité de cette fonction.

L'algorithme général : *Par convenance, on représente l'exécution la méthode branch-and-bound à travers une arborescence. La racine de cette arborescence représente l'ensemble de toutes les solutions du problème considéré. Dans ce qui suit, nous résumons la méthode branch-and-bound sur des problèmes de minimisation.*

Pour appliquer la méthode branch-and-bound, nous devons être en possession :

- *d'un moyen de calcul d'une borne inférieure d'une solution partielle,*
- *d'une stratégie de subdiviser l'espace de recherche pour créer des espace de recherche de plus en plus petits,*
- *d'un moyen de calcul d'une borne supérieure pour au moins une solution.*

La méthode commence par considérer le problème de départ avec son ensemble de solutions, appelé la racine. Des procédures de bornes inférieures et supérieures sont appliquées à la racine. Si ces deux bornes sont égales, alors une solution optimale est trouvée, et on arrête là. Sinon, l'ensemble des solutions est divisée en deux ou plusieurs sous-problèmes, devenant ainsi des enfants de la racine.

La méthode est ensuite appliquée récursivement à ces sous-problèmes, engendrant ainsi une arborescence. Si une solution optimale est trouvée pour un sous-problème, elle est réalisable, mais pas nécessairement optimale, pour le problème départ. Comme elle est réalisable, elle peut être utilisée pour éliminer toute sa descendance : si la borne inférieure d'un noeud dépasse la valeur d'une solution déjà connue, alors on peut affirmer que la solution optimale globale ne peut être contenue dans le sous-ensemble de solution représenté par ce nœud. La recherche continue jusqu'à ce que tous les nœuds sont soit explorés ou éliminés.

1.4 Conclusion

Nous avons présenté dans ce premier chapitre les différentes méthodes d'optimisation globale. Parmi les méthodes déterministes, les méthodes de recouvrement qui ont la réputation d'être efficaces en dimension 1. Contrairement aux méthodes déterministes, les méthodes stochastiques ont l'avantage d'être facilement exécutables sur machine. Cependant, ces méthodes ne sont pas efficaces mais elles sont utilisées lorsqu'il n'est pas possible de résoudre le problème avec une méthode déterministe.

Chapitre 2

Analyse des intervalles

L'analyse des intervalles est née dans les années 1960[20] avec l'apparition des premiers ordinateurs. Beaucoup de chercheurs ont commencé à s'intéresser aux différentes méthodes pour prendre en compte dans les calculs l'erreur d'arrondi liée à la précision machine. Le but était de se rapprocher au maximum des mathématiques exactes pour avoir une assurance sur les résultats numériques pour pouvoir les valider (cf exemple 2.0.1).

En effet tous les nombres réels ne sont pas représentables sur ordinateur et à l'époque les normes de codage des nombres n'étaient pas encore instaurées ainsi les résultats pouvaient différer suivant la machine ou le langage utilisé. Les seuls nombres exacts sont ceux pouvant être codé en binaire sur 32 bits pour la simple précision et 64 bits pour la double précision, de nos jours, tous les autres sont arrondis au nombre codable le plus proche suivant la norme IEEE(Institute of Electrical and Electronics Engineers) [9].

Comme on peut le constater, le vrai résultat n'a aucun rapport avec la valeur retournée par l'ordinateur, même le signe n'est pas correct.

Le premier livre sur ce domaine est celui de Moore (1966) [20]. Ce n'est pas historiquement le premier travail sur ce sujet, mais c'est à la suite de ce livre que plus de 1000 article furent écrits, d'où on peut le qualifier du livre sacré de l'analyse d'intrevalle.

L'idée de cette arithmétique est de représenter tous les nombres réels par deux nombres flottants qui l'encadrent.

Exemple 2.0.1 *[24]*
Considérons l'équation suivante :

$$f(x,y) = 33.75y^6 + x^2(11x^2y^2 - y^6 - 121y^4 - 2) + 5.5y^8 + \frac{x}{2y}$$

L'évaluation au point $x = 77617$ *et* $y = 33096$*, en Fortran double précision, donne :*
$f((77617, 33096)) = 1.1726039400531...$
Alors que le vrai resultat est : $f((77617, 33096)) = -\frac{54767}{66192} = -0.8273960599$.

Exemple 2.0.2 *Pour représenter* $\frac{1}{3}$ *sur machine, on utilise un intervalle contenant la valeur réelle :*
$\frac{1}{3} \longrightarrow$ *[0.33333331,0.33333335] avec un codage simple précision,*
$\frac{1}{3} \longrightarrow$ *[0.33333333333333331,0.33333333333333338] en double précision .*

2.1 Arithmétique d'Intervalles

La présentation des nombres en arithmétique d'intervalle, qui approche plus ou moins fidèlement la valeur désirée, est un intervalle la contenant. Par exemple, on peut tenir compte d'une erreur de mesure en remplaçant une valeur mesurée α avec une incertitude ε par l'intervalle $[\alpha - \varepsilon, \alpha + \varepsilon]$.

On peut également remplacer une valeur non exactement représentable, telle que $\frac{1}{3}$, par un intervalle la contenant ; si l'on dispose d'un ordinateur représentant les nombres en base 10 avec 3 chiffres, $\frac{1}{3}$ sera remplacé par $[0.33,\ 0.34]$. Enfin, si l'on désire obtenir un résultat valide pour tout un ensemble de valeurs, on utilise un intervalle contenant ces valeurs.

En effet, l'objectif de l'arithmétique d'intervalles est de transmettre des résultats qui contiennent avec certitude la valeur ou l'ensemble cherché, on parle alors de résultats garantis ou validés, ou encore certifiés.

Rappelons que les intervalles sont des sous-ensembles fermés connexes de $\mathbb{R}$. On notera $\mathbb{I}$ l'ensemble des intervalles de $\mathbb{R}$ ($\mathbb{I} \subset \mathcal{P}(\mathbb{R})$).

On peut les généraliser en plusieurs dimensions : un vecteur intervalle $x \in \mathbb{I}^n$ est un vecteur dont les n composantes sont des intervalles et une matrice intervalle $A \in \mathbb{I}^{m \times n}$ est une matrice dont les composantes sont des intervalles.

Une représentation graphique d'un vecteur de $\mathbb{I}^2$ et $\mathbb{I}^3$ est donnée dans la FIGURE 2.1. Elle illustre le fait qu'un vecteur intervalle est un ensemble parallélépipédique de vecteurs aux côtés parallèles aux axes du repère, cela justifie que par la suite on utilisera indifféremment les termes de vecteur intervalle, de pavé ou de boite ou même d'intervalle.

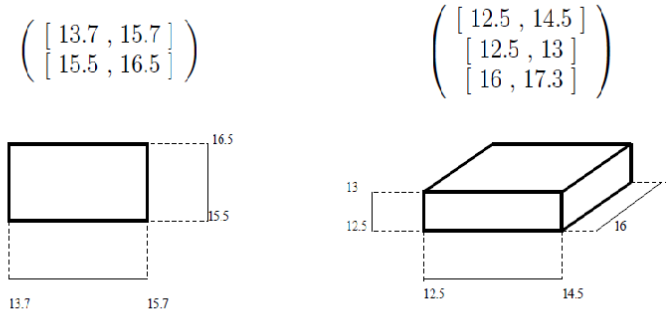


FIGURE *2.1 – Exemple des vecteurs de $\mathbb{I}^2$ et $\mathbb{I}^3$*

2.1.1 Notations et Définitions

Soit $\mathbb{R}$ l'ensemble des nombres réels et soit $\mathbb{I} = \{[a, b] \setminus (a, b) \in \mathbb{R}^2\}$, l'ensemble des intervalles compacts réels .

Un intervalle A appartenant à $\mathbb{I}$ est caractérisé par sa borne inférieure a^L et sa borne supérieure a^U : $A = [a^L, a^U]$.

Tout nombre réel x sera confondu avec l'intervalle $[x, x]$ correspondant.
Les intervalles auront pour notation des caractères majuscules et les caractères miniscules désigneront des nombres réels.

On définit un vecteur d'intervalles dans $\mathbb{I}^n$ comme un n uplets d'intervalles et une matrice d'intervalle dans $\mathbb{M}_{m\times n}(\mathbb{I})$ comme une matrice de taille $m\times n$ dont les composantes sont des intervalles.

Définition 2.1.1 ***(Milieu d'intervalle)***
On appelle milieu d'intervalle, noté mid(A), toute fonction définie de $\mathbb{I}$ dans $\mathbb{R}$, ou dans le cas de fonction vectorielles de $\mathbb{I}^n$ dans $\mathbb{R}^n$, tel que :

$$mid(A) = \frac{a^L + a^U}{2},\ A \in \mathbb{I}.$$

Par extension aux formes vectoreielles, nous obtenons :

$$mid(X) = (mid(X_1)..., mid(X_n)),\ X = (X_1, ..., X_n) \in \mathbb{I}^n.$$

Définition 2.1.2 ***(Largeur d'un untervalle)***
On appelle largeur d'un intervalle, noté $w(A)$, toute fonction de $\mathbb{I}$ dans $\mathbb{R}$, ou dans le cas de fonctions vectorielles de $\mathbb{I}^n$ dans $\mathbb{R}^n$, tel que :

$$w(A) = a^U - a^L,\ a \in \mathbb{I};$$

Par extension aux formes vectorielles nous obtenons :

$$w(X) = (w(X_1), ..., w(X_n)),\ X = (X_1, ..., X_n) \in \mathbb{I}^n.$$

Remarque 2.1.3
$X \in \mathbb{I}^n$ est un vecteur dont chacune de ses composantes est un intervalle. On désignera souvent par les termes "pavé" ou boite un élément de $\mathbb{I}^n$.

Définition 2.1.4
Soit f : $\mathbb{R}^n \to \mathbb{R}$, f possède une expression explicite si l'expression analytique de f est connue de façon explicite,c'est à dire qu'elle peut s'écrire en n'utilisant que des variables,des fonctions et des opérateurs élémentaires tels que $+$, $-$, $\times$, log, exp, cos, sin, arccos, */ On dit que f est une fonction explicite.*

Définition 2.1.5
Un problème explicite est un problème dans lequel toutes les fonctions possèdent des expressions explicites et les contraintes n'utilisent que les relations standards ($\leq$,$<$,$\geq$,$>$,$=$).

2.1.2 Opérations élémentaires

Les opérations sur l'arithmétique des intervalles sont définies de la façon suivante :

$$\textit{Pour tout } (A,B) \in \mathbb{I}^2, \quad A \star B = \{x_a \star x_b \setminus x_a \in A,\ x_b \in B\}$$

Le symbole $\star$ *étant l'une des opérations usuelles :* $+$, $-$, $\times$, $\div$, *comme suit :*

$$[a,b] + [c,d] = [a+c, b+d]$$
$$[a,b] - [c,d] = [a-d, b-c]$$
$$[a,b] \times [c,d] = [\min\{a\times c, a\times d, b\times c, b\times d\}, max\{a\times c, a\times d, b\times c, b\times d\}]$$
$$[a,b] \div [c,d] = [a,b] \times [\frac{1}{d}, \frac{1}{c}] \textit{ si 0 n'appartient pas } [c,d]$$

Remarque 2.1.6

En encadrant le couple de réels (x_a, x_b) *par un couple d'intervalles* (A,B), *on a toujours :*

$$x_a \star x_b \in A \star B, \ \textit{pour tout } (x_a, x_b) \in A \times B.$$

2.1.3 Fonctions Usuelles

Soit A *un intervalle de* $\mathbb{I}$, *tel que* $A = [a^L,\ a^U]$, *nous pouvons définir aussi :*

1. ***Puissance entiére :***

$$A^n = \begin{cases} [1,1], & \textit{si } n = 0; \\ [(a^L)^n, (a^U)^n], & \textit{si } a^L \geq 0 \textit{ ou } n \textit{ est impair}; \\ [(a^U)^n, (a^L)^n], & \textit{si } a^U \leq 0 \textit{ et } n \textit{ est pair}; \\ [0, max\{(a^U)^n, (a^L)^n\}], & \textit{si } 0 \in A \textit{ et } n \textit{ est pair}. \end{cases}$$

2. ***Logarithme :***

   Le logarithme est une fonction strictement croissante sur l'intervalle $]0,+\infty[$, *d'où :*

$$\log(A) = [\log(a^L), \log(a^U)], \ \textit{avec } A \subset]0, +\infty[.$$

3. ***Racine carrée :***

   La racine carrée étant une fonction strictement croissante sur l'intervalle $[0,+\infty[$, *nous obtenons :*

$$\sqrt{A} = [\sqrt{a^L}, \sqrt{a^U}], \ \textit{avec } A \subset [0, +\infty[.$$

4. ***Valeur absolue :***

$$|\ A\ | = \begin{cases} [a^L, a^U] & , \textit{ si } a^L \geq 0; \\ [|\ a^U\ |, |\ a^L\ |] & , \textit{ si } a^U \leq 0; \\ [0, max\{|\ a^U\ |, |\ a^L\ |\}] & , \textit{ si } 0 \in A. \end{cases}$$

5. ***Cosinus :***

   Pour calculer le cosinus d'un intervalle A *de* $\mathbb{I}$ *on doit ajouter* 2π *à* A *jusqu'à ce que* $a^L \in [0, 2\pi]$, *ensuite :*

$$\text{si } a^L \leq \pi, \text{ alors : } \begin{cases} a^U \leq \pi & \Rightarrow \cos A = [\cos a^U, \cos a^L], \\ a^U \in [\pi, 2\pi[& \Rightarrow \cos A = [-1, \max(\cos a^L, \cos a^U)], \\ a^U > 2\pi & \Rightarrow \cos A = [-1, +1], \end{cases}$$

$$\text{et si } a^L \in]\pi, 2\pi], \text{ alors : } \begin{cases} a^U \leq 2\pi & \Rightarrow \cos A = [\cos a^L, \cos a^U], \\ a^U \in]2\pi, 3\pi] & \Rightarrow \cos A = [\min(\cos a^L, \cos a^U), 1], \\ a^U > 3\pi & \Rightarrow \cos A = [-1, +1]. \end{cases}$$

6. ***Sinus :***

   $\forall A \in \mathbb{I}$, *le calcul de* $sin(A)$ *se fait de la même façon que cosinus ; on ajoute* 2π *ou on enlève* 2π *à* A, *jusqu'à ce que* $a^L \in [\frac{-\pi}{2}, \frac{3\pi}{2}]$, *ensuite :*

$$\text{si } a^L \leq \tfrac{-\pi}{2}, \text{ alors : } \begin{cases} a^U \leq \frac{-\pi}{2} & \Rightarrow \sin A = [\sin a^L, \sin a^U], \\ a^U \in]\frac{-\pi}{2}, \frac{3\pi}{2}] & \Rightarrow \sin A = [\min(\sin a^L, \sin a^U), 1], \\ a^U > \frac{3\pi}{2} & \Rightarrow \sin A = [-1, +1], \end{cases}$$

$$\text{et Si } a^L \in]\tfrac{\pi}{2}, \tfrac{3\pi}{2}], \text{ alors : } \begin{cases} a^U \in]\frac{\pi}{2}, \frac{3\pi}{2}] & \Rightarrow \sin A = [\sin a^U, \sin a^L], \\ a^U \in]\frac{3\pi}{2}, \frac{5\pi}{2}] & \Rightarrow \sin A = [-1, \max(\sin a^L, \sin a^U)], \\ a^U > \frac{5\pi}{2} & \Rightarrow \sin A = [-1, +1]. \end{cases}$$

2.1.4 Propriétés Elémentaires

Propriété 2.1.7 ***(Principe d'inclusion)***
Soit $(A, B, C) \in \mathbb{I}^3$. *Si* $A \subset C$ *et* $B \subset D$ *alors* $A \star B \subset C \star D$.

Propriété 2.1.8
Les opérations arithmétique sur les intervalles ne vérifient pas les propriétés algébriques de leurs analogues scalaires. En particulier la soustraction n'est pas la réciproque de l'addition, et la division n'est past la réciproque de la multiplication.

Exemple 2.1.9
On remarque que $[-2, 3] + [5, 7] = [3, 10]$
D'autre part, le résultat de cette opération moins la seconde opérande vaut :
$[3, 10] - [5, 7] = [-4, 5]$ *qui n'est pas égal à* $[-2, 3]$, *mais il le contient.*

Propriété 2.1.10 ***(Sous-distributivité)***
L'arithmétique d'intervalles n'a pas toutes les propriétés de l'arithmétique classique, par exemple elle n'est pas distributive :

$$A \times (B + C) \subseteq A \times B + A \times C, \quad Pour (A, B, C) \in \mathbb{I}^3$$

On dit qu'elle est sous-distributive .

Propriété 2.1.11
Les nombres réels sur machine ne sont pas représentables, donc le résultat exacte de l'opération $a \star b$ *ne pourra être connu, nous encadrons donc celui-ci par l'intervalle* $A \star B$.

$A \star B$ *sera le plus petit intervalle connu contenant* $a \star b$. *L'extensions de ces définitions aux vecteurs d'intervalle de* $\mathbb{I}^n$ *sont sans aucune difficulté.*

Propriété 2.1.12
Le triplet $(\mathbb{I}, +, 0)$ *est donc un monoïde commutatif, ainsi que* $(\mathbb{I}, \times, 1)$.

Propriété 2.1.13

La propriété de sous-distributivité implique que les ensembles $\mathbb{I}^n$ ne sont pas des espaces vectoriels sur $\mathbb{R}$. Cependant, cela n'interdit pas de définir sur ces ensembles les opérations analogues aux opérations sur $\mathbb{R}^n$ et le calcul matriciel à l'aide des matrices d'intervalles.

2.1.5 Aritrhmétique des intervalle étendue

Cette arithmétique a été introduite par Hansan[8] et par Kahan [11]en 1968. Elle permet d'étendre l'arithmétique d'intervalles standard aux divers cas générant l'infini ; notamment la division par un intervalle contenant 0.

Soient $A = [a^L, a^U], \quad B = [b^L, b^U] \in \mathbb{I}$ *et* $c \in \mathbb{R}$, *alors on obtient les cas suivants de* $A \div B$:

$$\begin{aligned}
[\frac{a^U}{b^L}, +\infty], &\ si\ a^U \leq 0\ et\ b^L = 0,\\
[-\infty, \frac{a^U}{b^U}] \cup [\frac{a^U}{b^L}, +\infty], &\ si\ a^U \leq 0\ et\ 0 \in [b^L, b^U],\\
[-\infty, \frac{a^U}{b^U}], &\ si\ a^U \leq 0\ et\ b^L = 0,\\
[-\infty, +\infty], &\ si\ 0 \in a^L, a^U,\\
[-\infty, \frac{a^L}{b^L}], &\ si\ a^L \geq 0\ et\ b^U = 0,\\
[-\infty, \frac{a^L}{b^L}] \cup [\frac{a^L}{b^U}, +\infty], &\ si\ a^L \geq 0\ et\ 0 \in [b^L, b^U],\\
[\frac{a^L}{b^U}, +\infty], &\ si\ a^L \geq 0\ et\ b^L = 0.
\end{aligned}$$

Les régles pour l'addition et la soustraction sont :

$$\begin{aligned}
A + [-\infty, c] &= [-\infty, a^U + c],\\
A + [c, +\infty] &= [a^L + c, +\infty]\\
A + [-\infty, +\infty] &= [-\infty, +\infty]\\
A + [-\infty, c] &= [a^L - c, +\infty]\\
A - [c, +\infty] &= [\infty, a^U - c]
\end{aligned}$$

2.1.6 Arithmétique d'intervalle arrondie

Nous avons vu que l'arithmétique d'intervalles nous permettait de calculer des bornes précises de l'encadrement, c'est à dire une borne inferieure et une borne supérieure, des opérations élémentaires : $+, -, \times, \div$.

Concernant l'implémentation de l'arithmétique d'intervalles sur machine, il est bien entendu impossible d'utiliser des nombres réels, une version n'utilisant que des nombres flottants est donc nécessaires.

Cette arithmétique s'appelle l'arithmétique d'intervalles arrondie, elle consiste à approximer les nombres réels par le nombre flottant le plus proche de telle sorte que l'intervalle réel soit inclus dans l'intervalle arrondi [30].

Exemple 2.1.14

Supposant que le calcul de la borne inferieur et la borne supérieur soit d'une grande précision, sachant que même au niveau des opérations élémentaires le calcul peut manquer de précision.
On a : [0.1339,0.6496] + [0.2724,0.9161] = [0.4063,1.5657].
Mais sur MatLab avec une simple précision on aura l'interval [0.4062,1.5658].
Ce qui fait ce résultat peut être arrondi par [0.407,0.565] avec une précision de 3 décimales.
Mais nous aurions pu espérer avoir l'encadrement [0.405,1.566] car le calcul de la somme est vraiment inclus dans cet intervalle.

Pour cette raison, nous utiliserons directement cette façon d'arrondir en définissant deux fonctions :

i/ ***Arrondit supérieur :*** *On arrondit x^U par le plus petit nombre représentable par la machine supèrieur à x^U.*

ii/ ***Arrondi inférieur :*** *On arrondit x^L par le plus grand nombre représentable par la machine inférieur à x^L.*

Une étude pour créer le standard IEEE-P1788 définissant cette arithmétique est actuellement en cours [22]. Mais, il existe déjà de nombreuses implémentations performantes pour cette arithmétique, voici une liste non exhaustive dans différents langages de programmation :

- *INTLIB sur Fortran 77 et Fortran 90[12]*
- *INTLAB sur Matlab [29],*
- *PROFIL/BIAS sur C [13],*
- *une librairie directement incluse dans le compilateur SUN f90 [10].*

Ces différentes librairies permettent d'utiliser le type de variable interval et de manipuler comme tous les autres types de données. Les opérateurs et les fonctions usuelles sont simplement redéfinis par surcharge d'opérateur [19].
Pour plus de détaille de cette arithmétique voir[20].

2.2 La théorie de l'analyse des intervalles en optimisation globale

L'analyse des intervalles offre de nouvelles méthodes de résolution des problèmes d'optimisation global avec ou sans contraintes. En effet, elle assurera, grâce à l'information globale ainsi fournie, que l'algorithme d'optimisation détermine bien l'optimum globale et qu'il n'est pas piégé par un optimum local.

Dans cette partie, nous allons tout d'abord présenter l'optimisation globale sans contraintes qui permet de fournir des bornes inférieures et supérieures de la fonction objectif, une fois ces bornes trouvées, la résolution d'un problème d'optimisation global sans contrainte par un algorithme de type branch and bound est alors possible. Puis nous allons voir que l'analyse des intervalles permet de résoudre les systèmes d'équations qui découlent des contraintes en optimisation global sous contraintes.

2.2.1 Optimisation globale sans contraintes

La résolution des problèmes d'optimisation globale sans contraintes par les méthodes d'intervalles se fait par des algorithmes de type branch and bound. qui sont classés comme algorithmes d'optimisation globale qui produisent avec certitude l'optimum globale d'une fonction continue ainsi que tous ses optimiseurs même si l'optimum est sur l'une des frontières de pavé initial caractérisant le domaine de recherche.

L'idée de ces algorithmes est de décomposer le pavé initial (région faisable) en des sous pavés puis éliminer tout sous pavé qui ne peut pas contenir un minimiseur global tout en gardant les sous pavés restants dans une liste. La nomination "branch and bound" provient du fait qu'ils sont basés sur deux étapes importantes qui génèrent l'itération de l'algorithme [3] :

i/ ***Etape de partionnement (branching)*** *:*

Dans cette étape, la région faisable X est décomposée en sous-ensembles afin d'obtenir une partition de celui là : $X = \bigcup X_i$.
A la première itération, la partition ne contient qu'un seul ensemble qui est la région faisable de X.

ii/ ***Etape de bornétude (bounding)*** *:*

Dans cette étape, on détermine des minorants l_i de la fonction objectif f sur les sous-ensembles X_i : $l_i \leq \min f(x)$.
L'algorithme continue à raffiner la partition de X jusqu'à l'obtention du minimum global ainsi que tous les minimiseurs.
Dans les algorithmes de branch and bound classiques, le calcul d'un minorant de f sur chaque sous pavé X_i issu de la décomposition de la région faisable X est effectué par plusieurs procédures. La plus connue parmi ces procédures, est celle basée sur l'évaluation de la fonction f en certains point de X_i choisis suivant un critère donné.

2.2.1.1 Fonctions d'Inclusion

Soit $D \subseteq \mathbb{R}$ et $f : D \to \mathbb{R}$.

Définition 2.2.1 ***(Image directe d'une application)***
Soit $f(Y) = \{f(y) \backslash y \in Y\}$, pour tout Y inclus dans $\mathbb{I}(D)$, où $\mathbb{I}(D)$ est un intervalle compact inclus dans D.
$f(Y)$ est applée image directe de f sur Y.

Définition 2.2.2 ***(Fonction d'inclusion)***
Une fonction $F : \mathbb{I}(D) \to \mathbb{I}$ est appelée fonction d'inclusion pour f si et seulement si $f(Y) \subseteq F(Y)$, pour tout $Y \in \mathbb{I}(D)$, où $f(Y)$ représente l'image directe de f sur Y.

Le but de l'arithmétique d'intervalles est de permettre la construction de fonctions d'inclusion. Il en existe plusieurs, certaines fournissant des encadrements bien meilleurs que celle citée ci-dessous. Cependant la fonction d'inclusion étudiée ici a pour intérêt principal : sa simplicité et sa grande souplesse de programmation, utilisation de la surcharge d'opérateurs et de la généricité des langages tels que ADA, C^{++}, Fortran90.

2.2.2 Extension naturelle d'une expression de la fonction aux intervalles

Théorème 2.2.3 ***(Construction d'une fonction d'inclusion)***[30]

En supposant f littéralement connue, et en considérant une expression de $f(y)$ définie par des variables y_i, pour tout y appartenant à Y , ne dépendant que de la variable y , des constantes et des coefficients réels , des quartes opérations arithmétiques : $+$, $-$, $\times$ et $\div$, des fonctions pré-déclarées exp, sin, cos, log,...et des symboles auxiliaires, comme les parenthèses. Alors, l'extension *de cette expression aux intervalles (c.-à-d. remplacement de chaque occurrence de y par le pavé Y, les fonctions pré-déclarées par des fonctions d'inclusion correspondantes et les opérations arithmétiques par des opérations correspondantes sur les intervalles) est une* fonction d'inclusion.

Exemple 2.2.4 *Si $f : \mathbb{R} \to \mathbb{R}$ est définie par $x \mapsto x^2 - 2x + 1$ et si $X = [-1, 3]$.*
En remplaçant x par X dans l'expression de f on obtient

$$X^2 - 2X + 1 = [-5, 12].$$

Si on utilise plutôt l'expression "équivalente en arithmétique réelle"
$f(x) = x(x - 2) + 1$, on obtient

$$X(X - 2) + 1 = [-8, 4].$$

Et enfin en utilisant l'écriture factorisée$f(x) = (x - 1)^2$, on obtient

$$(X - 1)^2 = [0, 4].$$

Cet exemple illustre clairement le fait que des expressions équivalentes en arithmétique réelle ne le sont plus en arithmétique par intervalles, même si chacune donne lieu à un sur-encadrement de l'image de X par f.

Propriété 2.2.5

*Pour une fonction quelconque, l'évaluation optimale de cette fonction sur un intervalle est un but inaccessible : en effet, le problème a priori plus simple de l'évaluation à ε prés d'une fonction polynomiale à plusieurs variables et à coefficients rationnels est NP-difficile [**?**].*

Remarque 2.2.6

- *Un problème est dit **NP-difficile** si il est au moins aussi difficile que tout problème de **NP***

- *Toutes les extensions naturelles aux intervalles de différentes expressions d'une même fonction ne sont pas équivalentes.*
- *Si la la fonction est monotone sur un intervalle X de $\mathbb{I}$, alors nous pouvons calculer directement l'image directe de f*

$$f(X) = [f(x^L), f(x^U)], \text{ si } f \text{ est croissante ;}$$
$$f(X) = [f(x^U), f(x^L)], \text{ si } f \text{ est décroissante ;}$$

Exemple 2.2.7 *Soit f une fonction telle que $\forall x \in X, f(x) = xx + 3x + 1$ où $X = [0.1]$, alors*

$$F(X) = XX + 3X + 1 = [0.1] + [0.3] + [1.1] = [1.5],$$

donc $f(x^*) = \min f(x) \in [1.5]$.

2.2.3 Optimisation globale avec contrainte

Le problème d'optimisation globale avec contraintes est un problème d'optimisation globale d'une fonction réelle f sur un ensemble borné S de $\mathbb{R}^n$ et dont la solution vérifie une ou plusieurs contraintes sous forme d'égalités ou d'inégalités ou bien deux ensembles. Il est de la forme :

$$\begin{cases} \min f(x) \\ c_i(x) = 0 \quad i=1,..,n\,; \\ g_i(x) \leq 0, \quad j=1,..,n. \end{cases} \tag{2.1}$$

Cette vérification ce fait de la manière suivante :
Soit S_i un sous pavé issu de la décomposition de la région faisableS.

i/ ***Cas d'une contrainte égalité :***

Considérons la contrainte égalité $h(x) = 0$. Soit H une fonction d'inclusion de la fonction $h(x)$. Si 0 n'appartient pas $H(S_i)$ alors le sous-pavé S_i est rejeté de domaine de recherche du minimiseurs globale de f.

ii/ ***Cas d'une contrainte d'inégalité :***

Soit la contraite d'inégalité $g(x) \leq 0$, et soit G une fonction d'inclusion de la fonction g. Si $G^L(S_i) > 0$ alors le sous-pavé S_i est rejté ((G^L) est la borne inférieure de l'intervalle $G(X_i)$).

Dans le cas où le problème d'optimisation est soumis à plusieurs contraintes (égalités, inégalités ou les deux en même temps)la vérification se fait pour chaque contrainte.

Les fonctions d'inclusion H et G données ci dessus sont applées "contraintes d'intervalles".

La résolution des problèmes d'optimisation avec contraintes se ramène à la résolution des systèmes d'équations comme suit :
Soit $f : S \subseteq \mathbb{R}^n \to \mathbb{R}^n$ une fonction continûment différentiable. La résolution du problème :

$$\begin{cases} \min f(x) \\ h_i(x) = 0, \quad i = \{1, .., m\}. \end{cases} \tag{2.2}$$

où les fonctions h_i sont continûment différentiables sur $\mathbb{R}^n$, revient à optimiser le lagrangien L défini par :

$$L(x, \lambda_1, \lambda_m) = f(x) + \sum_{i=1,m} \lambda_i h_i(x) = 0, \quad \lambda_i \in \mathbb{R}, \quad \forall i \in 1, ..., m \tag{2.3}$$

et à rèsoudre le système d'équations :

$$\begin{cases} \nabla f(x_i) + \sum_{i=1,m} \lambda_i \nabla h_i(x) = 0 \\ h_i(x_i) = 0, \quad i \in 1, .., m \end{cases}$$

Rapplons qu'on cherche les minima de L aussi bien que ses maxima car un maximum de L peut être un minimum de f.

La résolution du problème :

$$\begin{cases} \min f(x) \\ g_i(x) \leq 0, i = \{1, .., p\} \end{cases} \tag{2.4}$$

où les fonctions $g_i(x)$ sont différentiables sur $\mathbb{R}^n$, revient à résoudre le système :

$$\begin{cases} \nabla f(\xi) + \sum_{j=1,..,p} \mu_j \nabla g_i(x) = 0 \\ \mu_j \geq 0, \quad \forall j = 1, ..., p \\ \mu_j g_j(x) = 0, \quad \forall j = 1, ..., p \end{cases}$$

La résolution du problème d'optimisation globale :

$$\begin{cases} \min f(x) \\ h_i(x) = 0, \quad i = 1, ..., m \\ g_i(x) \leq 0, \quad j = 1, ..., p \end{cases} \tag{2.5}$$

où les fonctions h_i, g_i et f sont continûment différentiables sur $\mathbb{R}^n$, revient à résoudre le système d'équations :

$$\begin{cases} \nabla f(\xi) + \sum_{i=1,p} \mu_j \nabla g_i(x) + \sum_{j=1,m} \lambda_i \nabla h_i(x) = 0 \\ \mu_j \geq 0, \quad \forall j = 1, ...p \\ \mu_j g_j(x) = 0, \quad \forall j \in 1, ..., p \end{cases}$$

Un élément de A_{ij} d'une matrice d'intervalles A peut être un réel α car on peut considérer ce réel comme intervalle en posant : $\alpha = [\alpha, \alpha]$.
Du même pour les éléments du vecteur d'intervalle B.

Définition 2.2.8 *On appelle une H matrice d'intervalles toute matrice A vérifiant :*
$\exists v = (v_1, ..., v_n) \in \mathbb{R}^n$ tel que $v_i > 0, \quad \forall i = 1, ..., n$ et $\langle A \rangle v > 0$.
où $\langle A \rangle$ est la matrice définie par :

$$\langle A_i \rangle = \begin{cases} \langle A_i \rangle, & si\ i = j; \\ - \mid \langle A_i \rangle \mid, & si\ i \neq j. \end{cases}$$

Définition 2.2.9 *Soit $A \in M_{m \times n}(\mathbf{II}(\mathbb{R}))$ une matrice d'intervalles. On matrice centre de A la matrice réelle notée A_c définie par :*
$A_c = ((A_{ij})_c)$ où $(A_{ij})_c$ est le centre de l'intervalle A_{ij}.

Remarque 2.2.10 *Les définitions introduites dans cette section sont propres aux intervalles.*

2.2.4 Conclusion

L'arithmétique d'intervalles constitue une bonne approche pour répondre à l'exigence de fiabilité des calculs. En effet, elle repose sur le principe que chaque calcul retourne un encadrement garanti de son résultat.
De plus, Le premier intérêt de l'arithmétique d'intervalle est de pouvoir prendre en compte les incertitudes de mesure dans le cas ou les données sont des valeurs expérimentales.
Le second intérêt est de permettre de manipuler sur ordinateur, en utilisant le calcul flottant, des quantités qui ne sont pas exactement représentables : par exemple, le nombre π pourra être représenté par $[3.14, 3.15]$ sur une machine qui calcule en base 10 avec 3 chiffres de mantisse, cet intervalle contient de façon garantie la valeur de π et tout calcul utilisant cet intervalle contiendra le résultat de calcul avec π.
Cette propriété de résultats constitue l'avantage essentiel de l'arithmétique par intervalles.

Chapitre 3

Application de l'analyse d'intervalle sur l'optimisation globale

*L'analyse d'intervalle continue à se développer mais avec des objectifs différents depuis le début des années (*1990*). Son atout majeur, est de permettre de calculer sur des ensembles, par exemple le c'est seul outil fiable permettant de déterminer l'optimum global d'une fonction continue, et de déterminer tous les zéros d'une fonction et de prouver en même temps leur existence et leur éventuelle unicité ou encore de déterminer l'image directe ou inverse d'un ensemble par une fonction.*

Dans ce chapitre nous allons présenter trois méthode pour l'application de l'analyse d'intervalle sur l'optimisation globale qui consiste à trouver le minimum exacte d'une fonction avec ou sans contraintes. Nous allons donner d'abord un algorithme de type Branch and Bound basé sur l'analyse des intervalles qu'est l'une des rares méthodes dans ce domaine, ensuite nous présenterons l'extension naturelle d'une fonction par la méthode de Taylor et la méthode de Baumann.

3.1 Algorithme branche and bound par intervalles

Les algorithmes branche and bound ont pris une place non négligeable dans le monde de l'optimisation globale. Les raisons de cet engouement sont simples, c'est un principe à la fois basique et général sur lequel peut venir se greffer de nombreuses idées pour améliorer la convergence. Dans notre étude nous intéresserons qu'à la méthode branch and bound basé sur l'analyse d'intervalle.

Les premiers algorithmes branch and bound par intervalle sont nées dans les années 70, c'est une méthode pour résoudre une classe de problèmes d'optimisation globale, il s'agit de la recherche par décomposition du domaine en sous-domaine. Cette technique repose sur l'utilisation de l'analyse des intervalles puisqu'à chaque étape, le problème courant est traité de manière globale, c'est à dire que le domaine (intervalle ou pavé) est considéré comme élément du calcul pour l'évaluation.

3.1.1 Principe de l'algorithme de Branch and Bound par intervalle

L'algorithme commence par calculer $F(X) = [F^L(X), F^U(X)]$ *(choix d'une fonction d'inclusion) et définir le record :*

$$z = f(x^*) \text{ où : } x^* = mid(F(X)).$$

Ensuite,il procède à la décomposition du pavé initial X *en sous-pavés* X_i *puis l'encadrement de* f *sur chaque sous-pavé grâce à l'arithmétique d'intervalles. Cet encadrement est noté :*$[F^L(X_i), F^U(X_i)]$ *(c'est un intervalle de* $\mathbb{R}$*).*

Les bornes $F^L(X_i)$, $F^U(X_i)$ *étant successivement un minorant et un majorant de* $F(X_i)$*, ils sont donc de même pour* $f(X_i)$ *telle que* F *est une fonction d'inclusion de* f*.*

De là, on peut déduire que tout sous-pavé X_i *ayant une borne inférieure* $F^L(X_i)$ *supérieure au record* z *ne peut pas contenir un minimiseur global de* f *; il sera donc rejeté. Les sous-pavés restants seront gardés avec leurs bornes inférieures sous forme de couples* $(X_i, F^L(X_i))$ *dans une liste.*

Le record z *va être évidemment amélioré à chaque itération.*

La décomposition d'un pavé $X = (X_1, X_2, ..., X_n)$ *se fait généralement dans la direction de la coordonnée pour laquelle l'intervalle auquel elle appartient est de taille maximale.*

3.1.2 Ordonnancement de la liste

L'ordonnancement de la liste des pavés issus de la décomposition du pavé initial peut se faire que l'on désire converger le plus rapidement possible vers une solution,ou que l'on désire mieux répartir les recherches sur tous les pavés :

i/ *Classer les pavés de la liste par ordre croissant par rapport à la borne inférieure de la fonction d'inclusion.*

ii/ *Classer les pavés de la liste par ordre décroissant par rapport à la taille des pavés (les plus larges seront décomposés en premier).*

iii/ *Classer les pavés de la liste par ordre décroissant par rapport à l'âge des pavés (les premiers entrés seront les premiers sortis).*

iv/ *Classer les pavés de la liste par ordre croissant par rapport à la valeur de la fonction au milieu du pavé. La rapidité de la convergence de l'algorithme dépend du choix de l'ordonnancement de la liste des sous-pavés.*

3.1.3 Algorithme Branch and Bound par intervalle

Algorithm 3.1 Algorithme Branch and Bound par l'analyse d'intervalle

Etape 1 : Poser $Y = X$, $y = F^L(X)$; Initialisation de la liste $L = ((Y, y))$ et poser : $z = F^U(X)$

Etape 2 : Choisir une direction k suivant une coordonnée : $k \in 1, ..., n$

Etape 3 : Déviser Y dans la direction $k : Y = V_1 \cup V_2$

Etape 4 : Calculer $F(V_1)$ et $F(V_2)$ et poser $v_i = F^L(V_i)$, $i = 1, 2$ et $z = min(z, F^U(V_1), F^U(V_2))$

Etape 5 : Supprimer (Y, y) de la liste L.

Etape 6 : Pour $i = 1, 2$: Si $v_i > z$ alors supprimer le couple (V_i, v_i) de la liste L.

Etape 7 : Ajouter le (les) couple(s) restant(s) à la liste L. Si la liste est vide alors arrêter (pas de points réalisables)

Etape 8 : Noter le couple dont la seconde composante est minimale par (Y, y).

Etape 9 : Si : $z - F^L(Y) < \varepsilon$ alors afficher $F(Y)$ et $Y : F^L(Y) \leq f^* \leq z$ sinon aller à l'étape 2.

3.1.4 Extension Naturel(EN)

L'extension naturelle aux intervalles d'une expression est simplement la réécriture de l'expression analytique d'une fonction en arithmétique d'intervalle. La forme de la fonction ne change pas, seul le calcul change (cf Exemple 3.1.1). Elle est notée $EN_f(X)$.

Exemple 3.1.1 *Voici un exemple d'application de l'extension naturelle [23] :*
Les occurrences de x_1 *sont remplacer par* $[1; 2]$ *et les occurrences de* x_2 *par* $[3; 4]$. *Les calculs sont ensuite effectués en utilisant les formules de l'arithmétique d'intervalles.* $\forall x \in X = [1; 2] \times [3; 4]$, $f(x) = x_1^2 + x_2^2 - x_1 x_2$
$EN_f(X) = [1; 2]^2 + [3; 4]^2 - [1; 2] \times [3; 4] = [2; 17]$
$EN_f(X) = [2; 17]$.
*Ainsi,*2 *est un minorant et* 17 *est un majorant de* f *sur* $[1; 2] \times [3; 4]$.

Ce qui fait le résultat du calcul par arithmétique d'intervalles retourne un intervalle contenant toutes les valeurs possibles de la fonction pour l'intervalle de départ choisi (Figure3.1),c'est-à-dire :

$$\forall X \in \mathbb{I}, \quad \forall x \in X, \quad f(x) \in EN_f(X) \tag{3.1}$$

Il est donc facile d'obtenir une borne inférieure et une borne supérieure d'une fonction sur un intervalle.
Néanmoins toutes les propriétés de l'arithmétique classique ne sont pas respectées. En effet l'arithmétique d'intervalle n'est que sous distributive (cf équation3.2) et la forme de l'expression a un impact sur les bornes calculées ; c'est à dire que deux expression sémantiquement équivalente ne retourne pas nécessairement le même résultat en arithmétique d'intervalle.(cf Exemple3.1.2) et (cf Exemple 3.1.3)

$$\forall (x, y, z) \in \mathbb{I}^3, x \times (y + z) \subseteq x \times y + x \times z \tag{3.2}$$

Ce problème est lié en partie au répétition des variables, car en arithmétique d'intervalle, lorsque l'on calcule $X - X$, on considère n'importe quel élément de X moins n'importe quel autre élément de X, donc le résultat n'est pas zéro (cf Exemple 3.1.2).

Exemple 3.1.2 *Soit* $X = [0; 1]$
$1 \times X - 1 \times X = 1 \times [0; 1] - 1 \times [0; 1] = [-1; 1]$
$X(1 - 1) = [0; 1] \times (1 - 1)) = 0$

Ceci implique donc que l'intervalle calculé par l'extension naturelle est rarement le plus petit intervalle contenant toutes les valeurs de $f(x)$. C'est pourquoi, il est très important de factoriser les expressions analytiques au maximum.

Exemple 3.1.3 $\forall x \in [1; 2] \times [3; 4]$,

$$\begin{aligned} f(x) &= x_1^2 + x_2^2 - x_1 x_2 \\ &= x_1(x_1 - x_2) + x_2^2 \end{aligned}$$

En arithmétique d'intervalle :

$$[1; 2]^2 + [3; 4]^2 - [1; 2] \times [3; 4] = [2; 17]$$
$$[1; 2] \times ([1; 2] - [3; 4]) + [3; 4]^2 = [3; 14]$$

l'image directe correspondant aux bornes exactes : $\forall x \in [1; 2] \times [3; 4], \quad f(x) \in [7; 13]$.

3.2 Domaines d'utilisation aux méthodes des intervalles

Nous allons illustrer l'application de l'arithmétique des intervalles par quelques exemples [3].

a/ *Résolution de système linéaires :*

Résoudre un système linéaire en arithmétique des intervalles, se donne une matrice A de taille $n \times n$ et un vecteur $b \in \mathbb{R}^n$ et déterminer un vecteur $x \in \mathbb{R}^n$ telle que :

$$A.X = b \tag{3.3}$$

l'arithmétique des intervalles consiste à déterminer l'ensemble des vecteurs d'intervalles (chacune des composantes et un intervalle)qui bornent l'ensemble des vecteurs de $\mathbb{R}^n$ solution de l'équation (3.3) ; par exemple en appliquant les versions par intervalles de la méthode de Gauss ou de celle de Gauss-Seidel.
Ces versions diffèrent significativement des algorithmes ponctuels correspondants. une détaillée de ces versions peut ce trouve dans [8].

b/ *Résolution de système non linéaires et optimisation :*

L'arithmétique des intervalles réussit à résoudre les systèmes non linéaires rencontrés on optimisation globale a cause de sa capacité de borner les fonctions.
Ces systèmes sont de la forme suivante :

$$F(x) = (f_1(x), f_2(x), ..., f_n(x)) = 0 \tag{3.4}$$

où $F : X \subseteq \mathbb{R}^n \to \mathbb{R}^n$ est une fonction non linéaire.

Si On suppose que $f_i(X)$ est un intervalle de $\mathbb{R}$, par exemple on calcule $f_i(X)$ par l'arithmétique des intervalle et on trouve que 0 n'appartient pas à $f_i(x)$ alors l'équation ne peut pas avoir une solution. Le même raisonnement intervient dans un problème d'optimisation globale avec contrainte de la forme :

$$\begin{cases} min\Phi(x) \\ c_i(x) = 0, \quad i = 1, ..., m; \\ g_i(x) \leq 0, \quad j = 1, ..., r \end{cases} \tag{3.5}$$

Si pour un sous ensemble Y de X on trouve que 0 n'appartient pas à $c_i(Y)$ pour un $i \in \{1, ..., m\}$ ou bien un $j \in \{m+1, ..., r\}$, alors Y est supprimé du domaine de recherche ; c'est qu'on appelle : "violation d'une contrainte".

c/ *Quadrature (approximation des intégrales)*

Les méthodes des intervalles ont permis, vu la forme du terme de l'erreur dans les approximations des intégrales, de produire des inclusions significatives et des bornes garanties de ces intégrales. Les formules de quadrature pour une fonction f de classe C^n sur l'intervalle $[a, b]$ sont de la forme :

$$L(f) = Q(f) + R(f) \tag{3.6}$$

où $L(f) = \in$ et $Q(x)$ est l'approximation de $L(f)$ et $R(f)$ est l'erreur de la quadrature.

d/ *Equations aux dérivées partielles (problèmes aux limites)*

Les équations aux dérivées partielles ou encore les systèmes d'équations aux dérivées partielles peuvent être convertis, grâce à la discrétisation, à des systèmes d'équations différentielles ordinaires ou à des systèmes algébrique linéaires ou non linéaires.
L'arithmétique des intervalles dans ce cas permet de borner rigoureusement l'erreur de discrétisation pour un problème de dimension infinie. D'autre techniques d'intervalles pour les équations différentielles ordinaires et partielles peuvent être trouvées dans[3].

3.3 Amélioration de calcul des bornes

Combinées à l'analyse d'intervalles qui offre une méthode plus simple pour trouver des minorants et des majorants de f grâce aux fonctions d'inclusion de f et c'est ce qu'on appelle "encadrement de". D'autre méthodes ont été dévloppées,en exploitant notamment des informations sur le gradioent (cf Figure3.1) ; le bute étant d'améliorer aux mieux le calcul de la borne inférieure et supérieure :

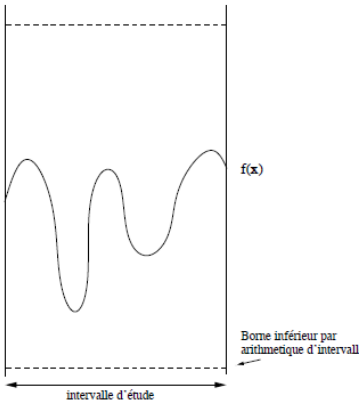


FIGURE *3.1 – En arithmétique d'intervalle, la fonction est encadrée par une boite rectangulaire*

3.3.1 Mèthode de Taylor

En appliquant la méthode des développements en séries de Taylor, il est possible d'obtenir une meilleure approximation de la borne inférieure. En effet, le calcul du gradient de f nous retourne une borne inférieure et supérieure du gradient c'est à dire un encadrement de la plus grande et la plus petite pente de f. Ainsi en effectuant le développement au point $(m, f(m))$ avec m le milieu de l'intervalle, on obtient la formule A. Il ne reste plus qu'à l'appliquer en a et b pour obtenir une borne inférieure de la fonction sur l'intervalle $[a, b]$ [18].

$$T(X) = f(m)+G(X)(m-X) = \begin{cases} \overline{G(X)}(m-a)+f(m), \underline{G(X)}(m-b)+f(m); \\ \text{ ou suivant la forme de } \overline{G(X)} \\ \underline{G(X)}(m-b)+f(m), \overline{G(X)}(m-a)+f(m). \end{cases} \tag{3.7}$$

où X est l'intervalle d'étude, G le gradient de f et m le mélieu de X .

3.3.2 Méthode de Baumann

Boumann en (1988) *[1], proposa une amélioration de la méthode de Taylor, en choisissant plus astucieusement le point de développement, c'est à dire qu'il ne considère plus le milieu m mais le centre optimal qui maximise la borne inférieure. Autrement, pour améliorer ce minorant Baumann propose un choix astucieux du point x^* du développement de Taylor pour maximiser le minorant obtenu avec T_1.*

Baumann démontre que ce point noté x^B s'obtient lorsque les points A et B de la figure 3.1 sont à la même hauteur. Il peut se calculer facilement grâce à la forme générale suivante :

Soit $x^B = (x_1^B, x_2^B, ..., x_n^B)$ le centre de Baumann de f sur $X = (x_1, x_2, ..., x_n)$.

Soit f la fonction définie sur $X = [a, b] \subset \mathbb{R}$ telle que $L \leq f'(X) \geq U$ et La valeur moyenne forme une relation concave obtenue par la fonction liée à la borne inférieure linéaire. Donc la borne inférieure de la fonction est atteinte à un point de X extrémal :

$$z_c^- = \min f(c) + (a-c)U, f(c) + (b-c)L. \tag{3.8}$$

Le centre optimal de Baumann est obtenue lorsque $f(c) + (a - c)U = f(c) + (b - c)L$ *et donne l'optimum, on peut voir dans le (Figure3.2) que les deux points* $M = (a; f(a))$ *et* $N = (b; f(b))$ *sont au même niveau, tel que :*

$$c_b^- = \frac{aU - bL}{U - L} \tag{3.9}$$

et la borne inférieure correspondant est :

$$z_b^- = f(\frac{aU - bL}{U - L} + \frac{(b - a)LU}{U - L}. \tag{3.10}$$

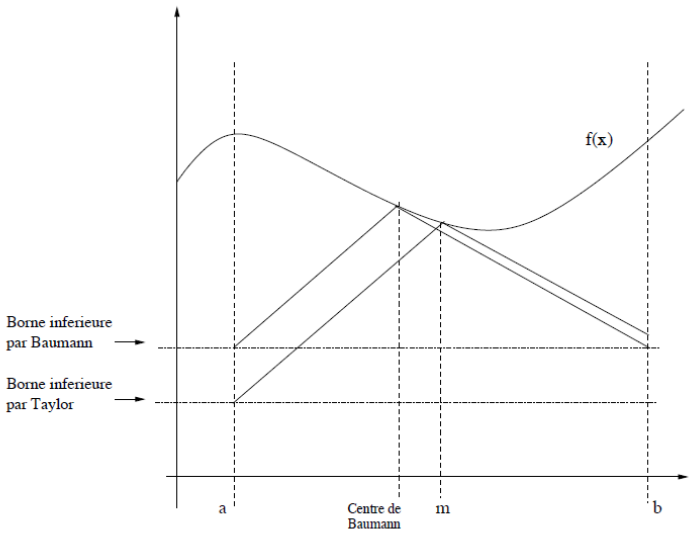


FIG. 2 – Schéma de la méthode de Taylor et Baumann

FIGURE *3.2 – Schéma de la méthode de Taylor et Baumann*

3.4 Conclusion

Nous avons présenté un algorithme déterministe d'optimisation globale pour résoudre des problèmes sans contraintes. Cet algorithme est basé sur l'arithmétique d'intervalle et sur les techniques branch and bound qui permettent de déterminer un encadrement précis de l'optimum global. Cependant il est quelques fois difficile d'obtenir rapidement la solution absolue en utilisant directement l'arithmétique d'intervalle, c'est pourquoi des techniques sont développées et ajoutées au problème général comme l'amélioration de la borne inférieure.

Chapitre 4

Implimentation

4.1 Introduction

L'analyse des intervalles a été introduit par R.E.Moore [20]. Elle se repose sur la représentation des nombres réels ou entiers par des intervalles qui les contiennent. Les intervalles peuvent être utilisés dans la prise en compte d'une incertitude sur une valeur mesurée, qui sera encadrée par des bornes inférieure et supérieure connues.

L'analyse des intervalles est utilisée dans de nombreux domaines, par exemple pour la commande robuste des systèmes, ainsi que pour l'estimation de paramètres, ou encore pour la détection de défaut.

Classiquement, les intervalles sont notés par [.], par exemple l'intervalle $[x]$ est noté par $[x_{inf}\ x_{sup}]$ où x_{inf} est la borne inférieure de $[x]$ et x_{sup} sa borne supérieure. De plus les opérations arithmétiques classiques telles que l'addition, la soustraction, la multiplication et la division peuvent être étendues aux intervalles en prenant quelques précautions(voir chapitre2).
Dans ce chapitre, nous allons présenter une introduction d'un logiciel efficace pour les besoins de notre implémentation qui est MatLab dont une boite à outil a été injecter pour faciliter le calcul des opérations arithmétiques d'intervalles, qui a pour appélation IntLab.

4.2 Introduction à Matlab

Matlab pour « MATtrix LABoratory », est un logiciel qui a été conçu pour fournir un environnement de calcul numérique de haut niveau. Il est particulièrement performant pour le calcul matriciel car sa structure de données interne est basée sur les matrices. Il dispose également de grandes capacités graphiques pour la visualisation d'objets mathématiques complexes.

Son fonctionnement repose sur un langage de programmation interprété qui permet un développement très rapide. Pour des applications nécessitant un temps de calcul plus élevé, un langage compilé comme le C++ ou le Fortran est mieux adapté.

4.3 Lancement de Matlab

Au lancement de MATLAB, une fenêtre s'affiche (fenêtre de commande) et le prompt matlab (») indique que matlab attend des instructions (variable ou valeur numérique). Les variables sont définies au fur et à mesure que l'on donne leurs noms (identificateur) et leurs valeurs numériques ou leurs expressions mathématiques. matlab ne nécessite pas de déclaration de type ou de dimension pour une variable. Les variables sont stockées dans l'espace de travail (ou workspace) et peuvent être utilisées dans les calculs subséquents. Chaque ligne écrite devant le prompt est apelée ligne d'instruction.

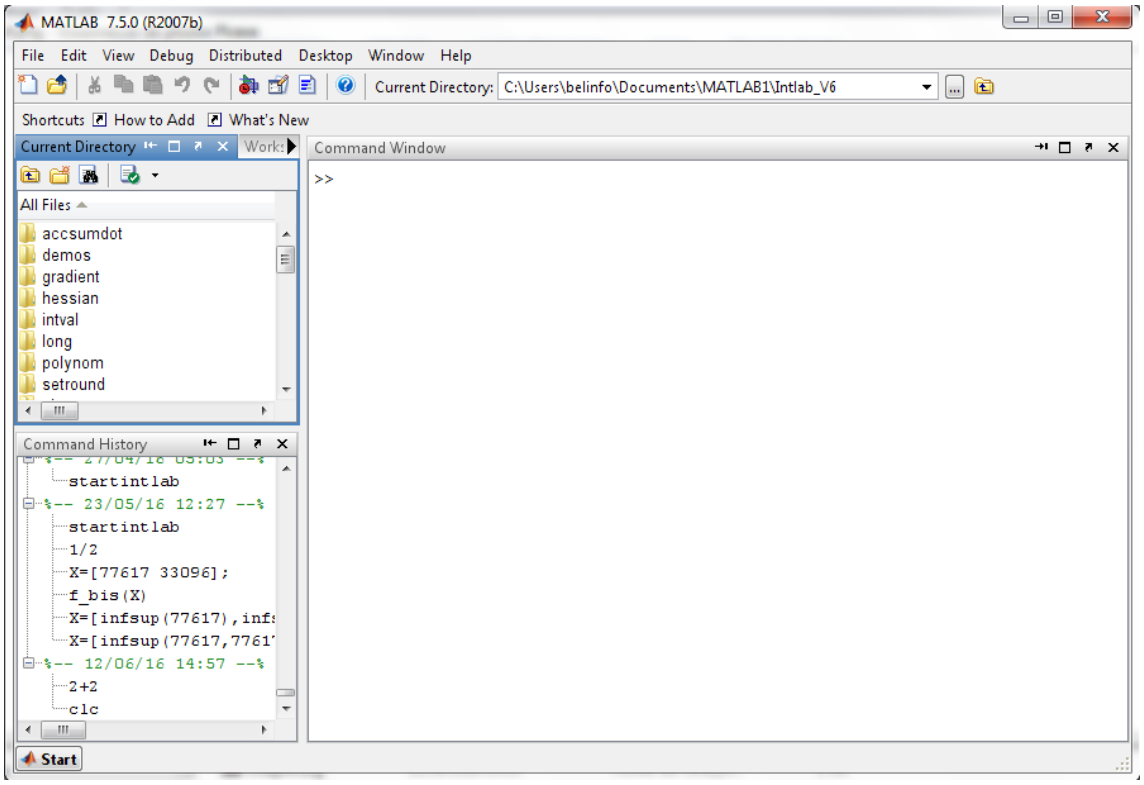


FIGURE 4.1 – L'interface de MatLab

4.3.1 Commandes et calculs de base

Matlab fonctionne de manière similaire à un shell Linux ou DOS. L'utilisateur rentre des commandes et Matlab les exécute.

Les prompts (≫) indique à l'utilisateur où il faut rentrer la commande. On ne peut pas « revenir en arrière », c'est-à-dire, il ne faut pas essayer de placer le curseur sur une ligne au-dessus du dernier (≫). Pour taper une autre commande on le fait à la suite. Matlab possede de nombreuses fonctions predefinies utiles en mathematiques. Pour avoir plus des information sur Matlab [26].
Dans ce chapitre nous allons se concentrer sur une boite à outil de Matlab ou une librairie de MatLab.
L'introduction d'un logiciel rapide et efficace pour l'arithmétique d'intervalle,comme Intlab la boite à outil Matlab, a abouti à la pluparité croissante de l'utilisation de l'analyse

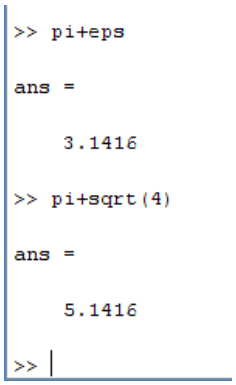


FIGURE 4.2 – Déroulement de calcul sur MatLab

d'ontervalle. Le concepte de l'analyse d'intervalle est de calculer de nombre réels en place des nombre réels.Alors que l'arithmétique floattante est afféctée par des erreurs d'arrondie

4.3.2 Arrondi vers l'exterieur

L'intervalle $x = [\underline{x}; \overline{x}]$ ne peut pas être représenté sur une machine si $\underline{x}$ et $\overline{x}$ ne sont pas les numéros de machines. Sur une machine $\underline{x}$ et $\overline{x}$ doivent être arrondis et la valeur par défaut est généralement arrondi à le nombre représentable le plus proche. Un intervalle arrondie de cette manière,$\tilde{x}$,peuvent pas lié l'intervalle x originale.
Pour que $x \subseteq \tilde{x}$,$\underline{x}$ doit être arrondi vers le bas et $\overline{x}$ doivent être arrondi vers le haut, qui est appelé arrondi vers l'esterieur.La norme IEEE omniprésente pour virgule flottante arithmétique dispose de quatre modes d'arrondi, plus proche, arrondir vers le bas, arrondir et retourna vers zéro, rendant ainsi possible l'arithmétique d'intervalle sur pratiquement tous les ordinateurs actuels.
INTLAB la boite à outil de MatLab utilise le "stround" routine pour changer l'accroissement, le mode du processeur entre le plus proche, et arrondir vers le bas. Cette routine à été utilisé pour créer des fonctions d'entrée, de sortie de l'arithmétique d'intervalle.

4.4 Introduction à INTLAB

Le logiciel INTLAB (extension de MATLAB) . INTLAB constitue une extension du logiciel MATLAB pour les calculs par intervalles. il existe plusieurs version de IntLab, dans notre travaille on a utilisé IntLab 6 [29].

Pour démarrer le calcul sur IntLab, on doit appeler toutes les définitions vu le chapitre 2, biensûr en selectionant le chemin courant de IntLab, en introduisant le commande >> stratintlab (voir FIGURE 4.3)

4.4.1 Intervalle entrée et sortie

Les intervalles réels dans INTLAB sont stockés par la borne inférieure et la borne supérieure, tandis que les intervalles complexes sont mémorisés par le point central ***(mid)*** *et le rayon* ***(rad)****(voir FIGURE 4.4). Toutefois, cela ne se voit pas par l'utilisateur. Les intervalles peuvent être entrés par deux façon :*

1. *soit en utilisant la borne supérieure et la borne inférieure de l'intervalle.*
2. *soit en utilisant le point central et rayon de l'intervalle (représentation complexe)*

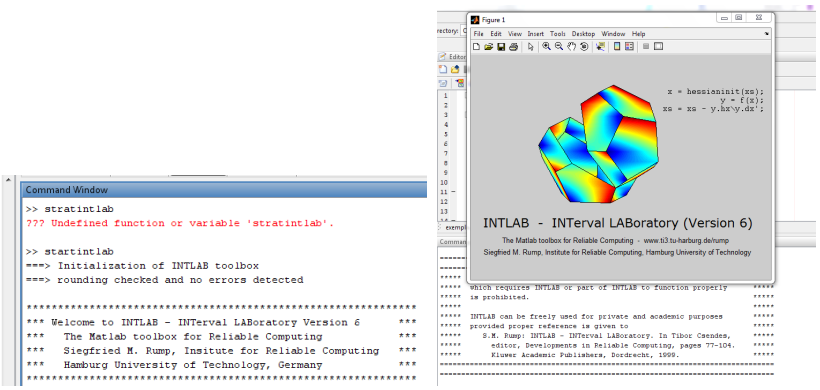


FIGURE 4.3 – Démarer IntLab

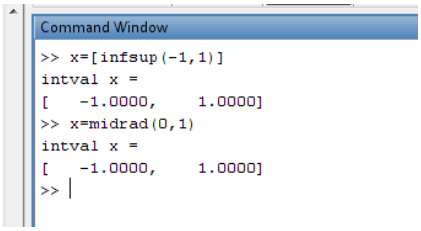


FIGURE 4.4 – Différentes représentation d'intervalle.

Par contre un intervalle complexe est stocké en tant que région circulaire en utilisant son point médian et son rayon, il est plus exact que cet intervalle soit entré de cette façon. Par exemple, la région circulaire qui a pour milieu 1+i et rayon 1 est entrée à l'aide
$>> \quad y = midrad(1 + i; 1);$

Si une région rectangulaire est entrée en utilisant la borne supérieure et la borne inférieure puis la région est stockée avec une surestimation comme la plus petite région circulaire de telle sorte la borne inférieure est entrée comme le point bas à gauche de la région et la borne supérieure est le point haut à droite.

Les vecteurs et les matrices d'intervalle sont entrés d'une manière similaire avec des arguments étant vecteurs ou matrices de la taille requise.
La fonction ***intval*** *fournit une autre façon d'entrer une variable d'intervalle ou convertir une variable en type intervalle. Cette fonction donne un intervalle avec des bornes d'une valeur réelle ou complexe. Il est bien connu que la valeur 0.1 ne peut pas être exprimée en binaire arithmétique en virgule flottante, ce qui fait* ***intval*** *peut être utilisé pour donner une borne rigoureuse, en exécutant* $intval(0.1)$*, où 0.1 est converti au format binaire avant d'être un intervalle de longueur 0.*
Ce qui fait pour avoir l'intervalle qui représentera 0.1, ***intval*** *utilise un argument de type chaîne de caractère (voir FIGURE 4.5) :*

La borne inférieure, la borne supérieure, le milieu et la langueur (rayon) de l'intervalle X peuvent être obtenus avec les commandes inf(x), sup(x), mid(x), rad(x) respectivement. La sortie par défaut est d'afficher les incertitudes de signe "-". Cela signifie que l'identité d'intervalle est donnée par le point milieu et le rayon (voir FIGURE 4.6).

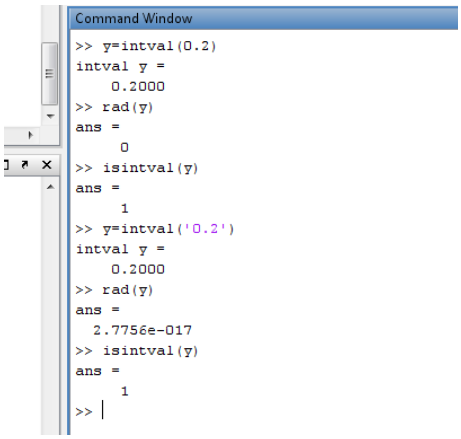


FIGURE 4.5 – Conversion en variable type intervalle.

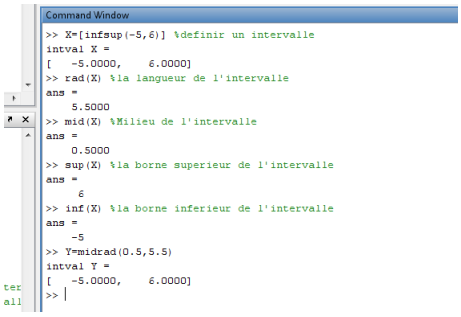


FIGURE 4.6 – La pièce identité d'un intervalle sur IntLab

4.4.2 INTLAB Arithmétique

INTLAB peut générer toute opération de base qui à pour variables vecteurs ou matrices, définis par des scalaires de type intervalle avec des bornes réelles ou complexes. Ces opérations sont entrés de la même façon que l'arithmétique réelle ou complexe, si la matrice $\boldsymbol{A}$ *est entrée alors* $\boldsymbol{A}^2$ *effectue des operations à base d'arithmétique d'intervalle avec* $\boldsymbol{A} * \boldsymbol{A}$*. Les fonctions standard telle que les fonctions trigonométriques et expentielles sont disponibles et utilisés de la manière habituelle sur MATLAB.*

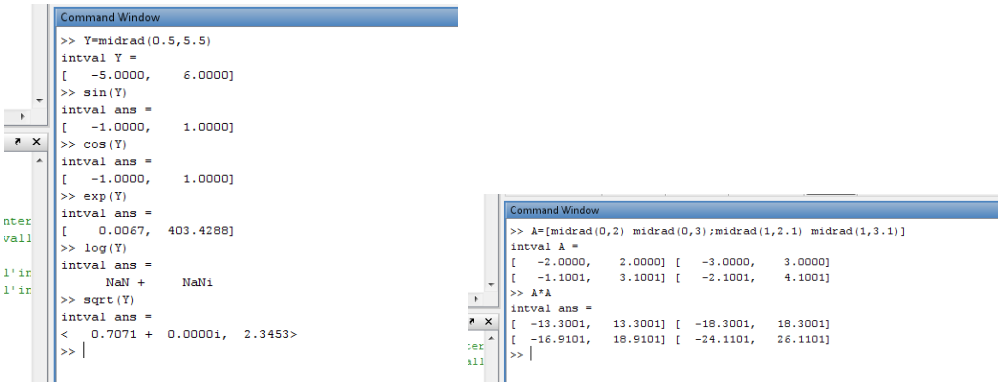


FIGURE 4.7 – Opérations Usuelles sur IntLab

INTLAB fournit aussi des fonctions pour un grand nombre des propriétés des intervalles comme étant des ensembles algébriques. Comme l'intersection et l'union de deux intervalles, aussi on peut vérifier si un intervalle est inclus dans un autre,

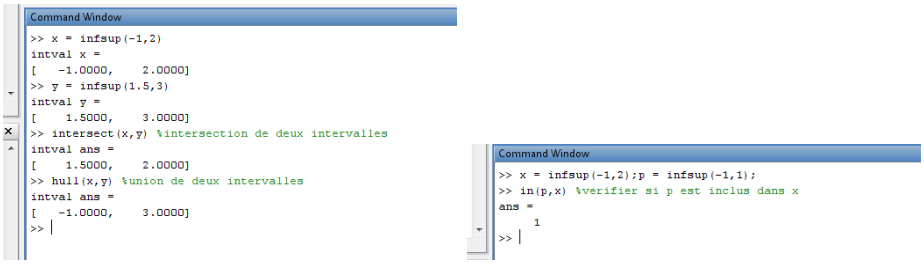


FIGURE 4.8 – Opérations algébrique sur deux intervalles

4.4.3 Mode Arrondi

La fonction la plus importante sur INTLAB est la fonction qui permet d'arrondir les résultats effectués qui est " setround " (mode d'arrondi). Cette instruction permet grâce à l'arithmétique d'intervalles de canaliser les erreurs lors de l'exécution d'un algorithme.
$>> y = getround$
y =
0
Le résultat $y = 0$ signifie que le mode d'arrondi est actuellement défini comme le plus proche. Ceci peut être changé en effectuant un arrondi vers le bas par
$>> setround(-1)$
ou un arrondi vers le haut par
$>> setround(1)$
A la fin de l'exécution il est important de redéfinir par défaut le mode d'arrondi par :
$>> setround(y)$

4.4.4 Gradient

INTLAB contient un paquet spécialement pour le calcul du gradient qui utilise la différentiation automatique. Un exemple à initialiser une variable à être utilisé avec le paquet de gradient est donnée par :
$>> u = gradientinit(2)$
gradient value u.x =
2
gradient derivative(s) u.dx =
1
Des bornes sur la dérivée d'une fonction peuvent être obtenues par un intervalle en utilisant le paquet du gradient.
$>> intvalinit('displayinfsup')$;
$\Rightarrow$ *Default display of intervals by infimum/supremum*
$>> x = gradientinit(infsup(0.999, 1.001))$
intval gradient value x.x =
$[0.9989, 1.0010]$
intval

gradient derivative(s) x.dx =
$[1.0000, 1.0000]$

4.5 Application Numérique

Plusieurs études comparatives ont été déja effectué sur les méthodes d'optimisation globale. Le critère d'efficacité le plus commun consiste à comparer le temps de calcul nécessaire pour déterminer la solution avec une précision donnée.
Parfois, le nombre d'évaluations de la fonction objectif est utilisé au lieu du temps. La fiabilité, et la simplicité de programmation et l'occupation en mémoire sont aussi des critères d'efficacité importants d'un algorithme.
Nous allons appliquer notre algorithme à certaines fonctions en comparant les résultats obtenus à ceux donnés au moyen d'autres méthodes.

4.5.1 Présentation des fonctions

Les fonction qui nous avons pris en considération sont les fonctions polynomiales à 2,3,4 variables.
Dans le premier tableau, On donne pour chaque problème polunomiale le domaine initial de la recherche.

Fonctions	*domaine initial de la recherche*
$f_1(x)$=$1 + x_1^2 + 2)x_2 + x_1x_2$	$X = [1, 2] \times [-10, 10]$
$f_2(x)$=$2x_1^2$-1.05x_1^4+x_2^2-x_1x_2+$\frac{1}{6}x_2^6$	X=[-2,4]×[-2,4]
$f_3(x)$=$(x_1$-$2x_2$-7$)^2$+$(2x_1$+x_2-5$)^2$	X=[-25,3.5]×[-1.5,4.5]
$f_4(x)$=$(1$+$(x_1$+x_2+1$)^2$(15-14x_1+$3x_1^2$-14x_2)+$(6x_1x_2$+ $3x_2^2)(30$+$(2x_1$-$3x_2)^2$(18-32x_1+$12x_1^2$+48x_2-36x_1x_2+ $27x_2^2)^2$	X=[-2,2]×[-2,2]
$f_5(x)$=$4x_1^2$-(2.1)x_1^4+$\frac{1}{3}$ x_1^6+x_1 x_2-$4x_2^2$+$4x_2^4$	X=[-100,100]×[-100,100]

TABLE *4.1 – Fonctions Testes*

Dans les tableaux de comparaison des différentes méthodes, on a utilisé les notations suivantes :
***CPU** :le temps en seconde sur la machine*
***its** : nombre d'itérations*
| L | : nombre de décomposition restantes.
***itsG** : nombre de fois que le gradient est calculer*
Le symbole - signifie que l'algorithme n'a pas donné de résultats.

Pbs	*EN*			*T1*				*TB*			
f	*CPU*	*Its*	$\|L\|$	*CPU*	*Its*	*ItsG*	$\|L\|$	*CPU*	*Its*	*ItsG*	$\|L\|$
f_1 1	*150.2446*	*15492*	*3629*	*0.5928*	*33*	*66*	*1*	*0.5304*	*28*	*56*	*1*
f_2 1	*116.4391*	*11672*	*4565*	*0.7332*	*25*	*50*	*2*	*0.7332*	*25*	*50*	*2*
$f_3$1	*12.2149*	*2837*	*614*	*0.4680*	*27*	*54*	*1*	*0.4212*	*24*	*48*	*1*
f_4 1	-	-	-	*328.4757*	*5280*	*10560*	*34*	*233.5023*	*3721*	*7442*	*20*
f_5 1	-	-	-	*24.2582*	*647*	*1294*	*16*	*19.0165*	*483*	*966*	*14*

TABLE *4.2 – Teste sans monotonie*

Pbs	*EN*			*T1*				*TB*			
f	*CPU*	*Its*	$\|L\|$	*CPU*	*Its*	*ItsG*	$\|L\|$	*CPU*	*Its*	*ItsG*	$\|L\|$
f_1 1	*150.2446*	*15492*	*3629*	*0.5928*	*33*	*66*	*1*	*0.5304*	*28*	*56*	*1*
f_2 1	*116.4391*	*11672*	*4565*	*0.7332*	*25*	*50*	*2*	*0.7332*	*25*	*50*	*2*
$f_3$1	*12.2149*	*2837*	*614*	*0.4680*	*27*	*54*	*1*	*0.4212*	*24*	*48*	*1*
f_4 1	-	-	-	*328.4757*	*5280*	*10560*	*34*	*233.5023*	*3721*	*7442*	*20*
f_5 1	-	-	-	*24.2582*	*647*	*1294*	*16*	*19.0165*	*483*	*966*	*14*

TABLE *4.3 – Teste avec monotonie*

4.5.2 Commentaires

Pour le premier tableau(Teste sans monotonie), en remarque que les fonctions d'inclusions f_1, f_2, f_3, f_4, f_5 ,l'algorithmes utilisant l'extension naturelle, la méthode de Taylor d'ordre1 ensuite la méthode de Baumann a été inefficace ceci revient à la complexité de l'expression des fonctions d'inclusion qui nécessite beaucoup de calculs.
Pour le deuxième tableau (Teste avec monotonie) nous remarquons une amélioration sensible de la rapidité de l'algorithme. Cette amélioration provient à le teste de monotonie qui permis d'améliorer sensiblement le temps d'exécution.

4.6 Conclusion

Les algorithmes Branch and Bound basés sur l'analyse des intervalles possèdent des techniques plus efficaces pour extraire les bornes inférieures de la fonction objectif. D'après la les testes qu'on a fait, en est ressorti que la méthode de Baumann donne toujours un résultat supérieur ou égale à celle de Taylor d'ordre 1, par contre il arrive parfois que l'extension naturelle fait mieux que la méthode de Baumann, ceci est revient de la complexité de l'expression de la fonction f qui nécessite beaucoup de calculs, comme on peut constater avec teste sans monotonie. Par contre dans le deuxièmes tableau on remarque que le teste de monotonie a permis d'améliorer sensiblement le temps d'exécution. Donc il est d'utiliser simultanément l'extension naturelle et la méthode de Baumann et de garder la meilleure des bornes.

Conclusion générale

Le retour au premier plan de l'optimisation globale correspond à un besoin industriel. De nombreuses applications, que ce soit au niveau de la conception ou de l'exploitation se ramènent à la recherche d'optima qui n'entrent pas dans le cadre des hypothèses simples qui facilitent cette tâche (convexité et donc unicité, différentiabilité, existence de points stationnaires,...). L'avantage de ces méthodes par intervalles, est de fournir avec certitude l'optimum global ainsi que tous les optimiseurs, quelle que soit la nature du problème : continu, mixte, avec ou sans contrainte.

Dans ce qui précède, nous avons tenté de montrer l'intérêt des algorithmes de type Branch and Bound par intervalles. Les développements successifs de ces méthodes permettent aujourd'hui la résolution exacte de problèmes jugés hier encore comme extrêmement difficiles, voire impossibles, à résoudre. Il semble cependant que ces algorithmes de type Brache and Bound par intervalle avec tous leurs accessoires (méthodes d'accélération, de calculs de bornes...) aient atteint leur niveau de plénitude ; de nombreuses applications ont été et sont résolues de manière exacte par ces algorithmes, mais l'on stagne maintenant sur de nouveaux challenges : comment améliorer ces méthodes pour franchir un nouveau pallier ?

Dans ce travail, on a présenté un algorithme Branch and Bound exploitant une nouvelle arithmétique définie entre les intervalles et permettant de trouver des inclusions de la fonction objectif et ainsi fournir ses bornes inférieures et supérieures. L'idée de base consiste à diviser la région faisable et à encadrer la fonction objectif sur les sous-pavés issus de la subdivision. Ensuite, on rejeté ceux qui ne peuvent pas contenir un minimiseur global et on divise la partie restante. On procède de cette manière jusqu'à l'obtention de tous les minimiseurs.

On a utilisé trois techniques d'encadrement basé sur l'arithmétique des intervalles : la première est l'extension naturelle qui fait appel à des fonctions d'inclusion, et les deux autres méthodes ont été développées, en exploitant notamment des informations sur le gradient. Puisque l'algorithme Branch and Bound dépend des bornes inférieures et du record exploités, on est toujours à la recherche d'un bon encadrement de la fonction objectif.

Enfin, durant ces vingt dernières années, un long chemin a été parcouru en optimisation globale par intervalles. La nécessite pratique énorme exigeant la résolution de problèmes d'optimisation globale nous incite à chercher de nouvelles approches qui prendraient en considération l'avantage du développement très rapide des ordinateurs et répondraient aux difficultés inhérentes à ce genre de problèmes.

Bibliographie

[1] *E. Baumann. Optimal centered forms.* BIT, 28(1), *pages 80–87, 1988.*

[2] *M. Ceberio.* Technique de préconditionnement et de résolution pour l'optimisation globale avec contraintes. *Doctorat, Institut de recherche en informatique de Nantes, Septembre 1999.*

[3] *Lamia. CHOUCHANE. Optimisation globale, méthode basé sur l'arithmétique des intervalles.* Université Ferhat Abbas, *2003.*

[4] *L. de Figueiredo et J. Stolfi. Adaptive enumeration of implicit surfaces with affine arithmetic.* Computer Graphics Forum, 15(5) :287–296, *1996.*

[5] *K. DU and R.B. KEARFOTT. The cluster problem in multivariate global.* Journal of Global Optimization, *1996.*

[6] *J. Stolfi et L. de Figueiredo. Self-validated numerical methods and applications.* Monograph for 21st Brazilian Mathematics, *1997.*

[7] *A.A GAGANOV. Computational complexity of the range of the polunomial in several variables.* Cybernetics, *pages 418–425, 1985.*

[8] *E.R. Hansen.* Global Optimisation using Interval Analysis,. *Doctorat, Marcel Dekker, Inc,New York, 1992.*

[9] *IEEE. Standard for binary floating-point arithmetic.* ANSI/IEEE Std 754, *1985.*

[10] *Sun Microsystems Inc. Fortran 95 interval arithmetic programming reference..* Rapport technique, Iuniverse Inc, *2002.*

[11] *W.M. KAHAN. A more complete interval arithmetic, technical report lecture notes for a summer courses.* University of Michigan, *1986.*

[12] *R.B. Kearfott.* Rigorous Global Search : Continuous Problems. *PhD thesis, Kluwer Academis Publishers, 1996.*

[13] *O. Knuppel.* A fast interval library,Computing. *Doctorat, PROFIL/BIAS, 53(3-4) :277–287, 1994.*

[14] *J.-T. LAPRESTE. Introduction à matlab ellipses. 2000.*

[15] *A.E. Csallner Marcot, T. Csendes. Multisection in interval branch-and-bound methods for global optimization ii. numerical tests.* Journal of Global Optimization, *2000.*

[16] *F. MESSINE. New forme of affine arithmétique in interval global optimisation algorithme.* Département d'Informatique de l'université de Pan, .

[17] *F. Messine.* Méthodes d'Optimisation Globale basées sur l'Analyse d'Intervalle Pour la Résolution de Problème avec Contraintes. *Doctorat, ENSEEIHT, Institut de Recherche en Informatique de Toulouse, Université Paul Sabatier- 118 Route de Narbonne - 31062 Toulouse Cédex 4, Septembre 1997.*

[18] *F. Messine.* L'optimisation Globale par Intervalles : de l'Etude Théorique aux Applications. *Habilitation, ENSEEIHT, ENSEEIHT-IRIT UMR-CNRS 5055, 2 rue Camichel, 31000 Toulouse, Février 2006.*

[19] *W. Miles.* Operator overloading in C. *Dctorat, University of British Columbia, 1995.*

[20] *R.E. Moore.* Interval Analysis. Prentice-Hall Inc. *Englewood Cliffs, N.J, 1966.*

[21] *J.L.D. COMBA M.V. ANDREADE and J. STOLFI. Affine arithmétic,interval.* Petersburg, *1994.*

[22] *American National.* IEEE standard for binary floating-point arithmetic. *Doctorat, Standards Institute and Institute of Electrical and Electronic Engineers, Std 754-2008. ANSI/IEEE Standard,, 2008.*

[23] *J. NININ. Résolution de problèmes d'optimisation par des méthodes de reformulations liniéres basées sur l'arithmétique affine.* IEEE trans on pattern Analysis and machine Intelligence, *2007.*

[24] *J. Ninin.* Optimisation Globale basée sur l'Analyse d'Intervalles : Relaxation Affine et Limitation de la Mémoire. *Doctorat, ENSEEIHT, ENSEEIHT-IRIT UMR-CNRS 5055, 2 rue Camichel, 31000 Toulouse, Décembre 2010.*

[25] *J. Nocedal and Stephen J. Wright. Numerical optimization.* Operations Research, *1999.*

[26] *M. POSTEL. Introduction au logiciel matlab.* Mémoire de DEA, université de d'Angers, *2004.*

[27] *M. Ratschak and Rokne.* New Copmuter Methods for Globale Optimisation. *PhD thesis, Ellis Horwood.*

[28] *NATHALIE REVOL. Introduction à l'arithmétique par intervalles. 2001.*

[29] *S.M. Rump.* INTLAB - INTerval LABoratory. In Developments in Reliable Computing,. *Doctorat, Kluwer Academis Publishers,, pages 77–104., 1999.*

[30] *A. TOUHAMI. Utilisation et extension de l'arithmétique affine dans les algorithmes déterministes d'optimisation globale.* Institut National Polytechnique de Toulouse-ENSEEIHT, *2002.*