MASTER EN SCIENCES MATHÉMATIQUES

Une méthode de type "branch and bound" en optimisation globale

PREZ, Nicole

Award date:
1989

Awarding institution:
Universite de Namur

FACULTES UNIVERSITAIRES N.D. DE LA PAIX NAMUR

FACULTE DES SCIENCES

UNE METHODE DE TYPE " BRANCH AND BOUND " EN OPTIMISATION GLOBALE.

Mémoire présenté pour l'obtention du grade de Licencié en Sciences mathématiques par

Nicole PREZ

Promoteur :

J.-J. STRODIOT

ANNEE ACADEMIQUE 88-89

Je remercie Monsieur J-J Strodiot pour ses conseils et sa disponibilité. Sans son aide, l'élaboration de ce travail n'aurait pas été possible.

Sommaire

1. Stratégie.

2. Partition.

   2.1. Donnée d'un n-rectangle.

   2.2. Définition.

   2.3. Bissection.

3. Bornes inférieures et supérieures.

4. Règles de suppression.

   4.1. pour les ensembles iminteressants.

   4.2. pour les ensembles non admissibles.

5. L'algorithme.

6. Commentaires.

   6.1. Règle de sélection.

   6.2. La suite (B_k).

   6.3. La suite (a_k).

7. Convergence de l'algorithme.

   7.1. Lemme.

   7.2. Convergence de la méthode.

Chapitre 3 : Implémentation.

1. Démarche suivie pour l'implémentation.

2. Exemples et résultats numériques.

3. Etude détaillée d'un problème.

Chapitre 4 : Application à un problème de découpe de diamant.

Introduction.

1. Description du problème.
2. Formulation mathématique.
   - 2.1. Introduction des notations.
   - 2.2. Formulation de la fonction objectif.
   - 2.3. Formulation des contraintes.
   - 2.4. Expression du problème.
3. Description de la méthode utilisée.
4. Résultats numériques.

Introduction

La plupart des méthodes numériques en programmation mathématique non convexe ne permettent que de trouver un minimum ou un maximum local. Dans ce mémoire, on examinera le problème de la recherche d'un minimum global dans le cas où la fonction f à minimiser est continue, et le domaine des contraintes D est un compact de $\mathbb{R}^m$.

Le problème général de minimisation globale d'une fonction quelconque f sur un ensemble D dont la solution existe a été étudié par R. Horst [Réf 1]. Cette méthode a été adaptée à beaucoup de cas particuliers : la minimisation concave [Réf 1,2], le cas où f peut s'exprimer comme la différence de deux fonctions convexes [Réf 1] ou lorsque f est une fonction Lipschitzienne [Réf 5].

La méthode étudiée dans ce mémoire est du type " Branch and Bound ". Elle consiste à partitionner un ensemble M_0 qui contient le domaine des contraintes dans le but de trouver une borne inférieure β et une borne supérieure α sur la valeur optimale du problème et cela de

telle façon que $\alpha - \beta$ tende vers zéro. Un exemple d'ensemble M_0 facile à utiliser est un m-rectangle de $\mathbb{R}^m$.

Le premier chapitre est consacré au cas où la fonction f est continue et D une partie compacte de $\mathbb{R}^m$. Après avoir décrit l'algorithme correspondant, on a démontré sa convergence. [Réf 2, 4]

Dans le deuxième chapitre, on adapte l'algorithme décrit précédemment à un cas particulier : le cas où la fonction f est concave et où le compact D est de la forme : $\{ x \in \mathbb{R}^m \mid g_i(x) \leq 0 \quad i = 1 \dots m, \ \underline{m} \leq x \leq \overline{m} \}$ où les fonctions g_i, $i = 1 \dots m$, sont des fonctions quadratiques séparables, $\underline{m}$ et $\overline{m}$ étant deux vecteurs de $\mathbb{R}^m$. [Réf 3]

La méthode décrite au deuxième chapitre est implémentée dans le troisième chapitre où six algorithmes sont présentés et testés sur différents problèmes.

Finalement, dans le dernier chapitre, on appliquera la méthode qui s'est révélée la plus efficace à un problème de découpe de diamants.

Chapitre 1 :

Un algorithme de type "Branch and Bound"

Motivation.

Ce chapitre propose une étude théorique d'une méthode de type Branch-and-Bound pour la résolution du problème suivant, noté (P) :

$$\min f(D) := \min \{ f(x) : x \in D \} = f(\bar{x})$$

avec . D ensemble compact de $\mathbb{R}^m$

. $f : \mathbb{R}^m \longrightarrow \mathbb{R}$ continue.

On recherche le minimum global de (P).

1. STRATEGIE.

En construisant des partitions de plus en plus raffinées d'un ensemble M_0, compact, contenant l'ensemble D, on va générer deux suites :

. la première, notée (α_k), décroissante

. la seconde, notée (β_k), croissante

telles que

$$\beta_k \leq \min f(D) \leq \alpha_k \quad \forall k.$$

On va donc situer la valeur minimale de f sur D dans un intervalle de plus en plus petit. On s'arrêtera lorsque l'on estimera que la différence $\alpha_k - \beta_k$ sera suffisamment petite.

2. ALGORITHME GENERAL.

Avant de présenter l'algorithme étudié pour la résolution du problème (P), rappelons la définition suivante:

2.1. Définition.

Soit M, une partie compacte de $\mathbb{R}^m$ et soit I, un ensemble fini d'indices.

Un ensemble $\{M_i : i \in I\}$ de sous-ensembles compacts est appelé partition de M si les deux égalités suivantes sont vérifiées:

$$\cdot\ M = \bigcup_{i \in I} M_i$$

$$\cdot\ M_i \cap M_j = \partial M_i \cap \partial M_j \qquad \forall i, j \in I\ ;\ i \neq j$$

avec ∂M_i désignant la frontière relative de M_i par rapport à M.

2.2. Algorithme.

Dans ce paragraphe, nous présentons l'algorithme de base. Des commentaires seront donnés au paragraphe suivant.

Pas 0

Etape 01 : Choisir $M_0 \supset D$, compact ; un ensemble fini d'indices I_0 ; une partition compacte $\mathcal{M}_0 = \{M_{0,i} : i \in I_0\}$ de M_0 satisfaisant

$$M_{0,i} \cap D \neq \phi \quad \forall i \in I_0.$$

Etape 02 : Pour chaque $i \in I_0$, déterminer

$$S_{0,i} \subseteq M_{0,i} \cap D \quad , S_{0,i} \neq \phi$$

et

$$\alpha_{0,i} = \alpha(M_{0,i}) := \min f(S_{0,i})$$

$$x^{0,i} \in \arg\min f(S_{0,i}).$$

Etape 03 : Pour chaque $i \in I_0$, déterminer

$$\beta_{0,i} = \beta(M_{0,i}) \leq \min f(M_{0,i} \cap D).$$

Etape 04 : Calculer

$$\alpha_0 := \min_{i \in I_0} \alpha_{0,i}$$

$$\beta_0 := \min_{i \in I_0} \beta_{0,i}$$

$$x^0 \in \arg\min \{ f(x^{0,i}) : i \in I_0 \}.$$

Etape 05 : Si $\alpha_0 = \beta_0$ Alors stopper l'algorithme (x^0 est solution)

Sinon aller au pas 1.

Pas k

Au début du $k^{ième}$ pas, nous avons :

- x^{k-1} (le meilleur point admissible obtenu jusqu'à présent)
- $\mathcal{M}_{k-1}$ (la partition actuelle de M_0)
- $\alpha_{k-1,i}$ et $\beta_{k-1,i}$ $\forall i \in I_{k-1}$
- α_{k-1} et β_{k-1} (les bornes actuelles du minimum)

Etape k1 : Supprimer tout $M_{k-1,i} \in \mathcal{M}_{k-1}$ tel que

$$\beta_{k-1,i} \geq \alpha_{k-1}$$

Noter R_k la collection des éléments $M_{k-1,i}$ de $\mathcal{M}_{k-1}$ non éliminés.

Etape k2 : Choisir $M_{k-1,i_k} \in R_k$.

Choisir un ensemble fini d'indices J_k.

Construire une partition

$$\mathcal{M}_{k-1,i_k} = \{ M_{k,i} : i \in J_k \}$$ de M_{k-1,i_k}

satisfaisant $M_{k,i} \cap D \neq \phi$.

Etape k3 : Pour chaque $i \in J_k$, déterminer

. $S_{k,i} \subseteq M_{k,i} \cap D$; $S_{k,i} \neq \phi$

tels que $S_{k-1,i_k} \subset \bigcup_{i \in J_k} S_{k,i}$

. $\alpha_{k,i} = \alpha(M_{k,i}) := \min \{ f(S_{k,i}) \}$

. $x^{k,i} \in \arg\min \{ f(S_{k,i}) \}$

. $\beta_{k,i}$ satisfaisant

$$\beta_{k-1,i_k} \leq \beta_{k,i} \leq \min \{ f(M_{k,i} \cap D) \}.$$

Etape k4 : Poser

$$\mathcal{M}_k = (R_k \setminus M_{k-1,i_k}) \cup \mathcal{M}_{k-1,i_k}.$$

Soit I_k, l'ensemble d'indices tel que

$$\mathcal{M}_k = \{ M_{k,i} : i \in I_k \}$$

représente la partition actuelle de l'ensemble M_0.

On note $\alpha_{k,i}$, $\beta_{k,i}$, $S_{k,i}$ et $x^{k,i}$ les

quantités correspondant à $M_{k,i}$ $\forall i \in I_k$.

Calculer

$$\alpha_k := \min_{i \in I_k} \alpha_{k,i}$$

$$\beta_k := \min_{i \in I_k} \beta_{k,i}$$

$$x^k \in \arg\min \{ f(x^{k,i}) : i \in I_k \}.$$

Etape k5 : Si $\alpha_k = \beta_k$

Alors stopper l'algorithme (x^k est une solution)

Sinon aller à l'étape $k+1$.

2.3. Description de l'algorithme.

Reprenons pas par pas l'algorithme afin d'en dégager l'idée générale. Rappelons-nous de cet algorithme comme l'algorithme 1.

Pas 0

On se propose de trouver le minimum global de la fonction f sur D, un compact.

On choisit un ensemble M_0, compact, contenant D, que l'on partitionne au sens de la définition 2.1. en ensembles $M_{0,i}$, $i \in I_0$, compacts, tels que chaque élément $M_{0,i}$ ait une intersection avec D. Dans chacune de ces intersections, on choisit un ensemble $S_{0,i}$ non

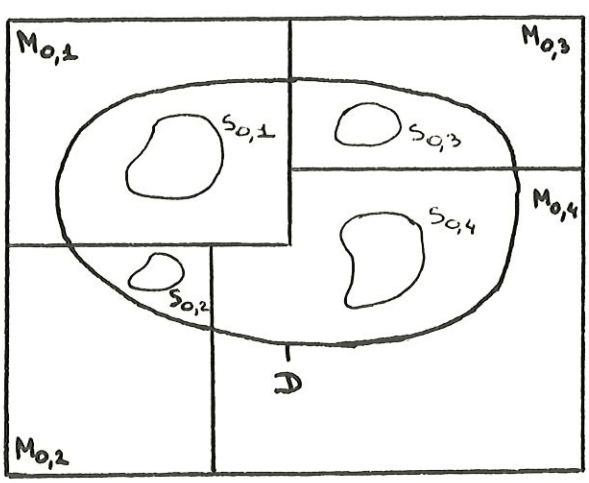
vide.

On calcule α_0 et β_0, les premiers éléments des deux suites à construire, de la façon suivante:

$$\alpha_{0,i} = \min f(S_{0,i})$$

$$\beta_{0,i} \leq \min f(M_{0,i} \cap D),$$

ce qui correspond à la situation suivante:



On calcule alors

$$\min_{i \in I_0} \alpha_{0,i} = \alpha_0$$

$$\min_{i \in I_0} \beta_{0,i} = \beta_0 .$$

On a bien que α_0 et β_0 encadrent la valeur de la solution car

. par construction, on a que $\bigcup_{i \in I_0} S_{0,i} \subset D$ et donc on obtient que $\min f(D) \leq \alpha_0$

. De plus, toujours par construction, on a que

$$\beta_0 \leq \beta_{0,i} \leq \min f(M_{0,i} \cap D) \quad \forall i \in I_0$$

et d'autre part les ensembles $M_{0,i}$; $i \in I_0$ forment une partition de M_0. On en déduit que

$$\beta_0 \leq \min f(D).$$

On prend ensuite un point x^0 vérifiant

$$f(x^0) = \alpha_0.$$

Si on a que $\alpha_0 = \beta_0$, alors $\alpha_0 = f(x^0) = \min f(D)$ et x^0 est bien solution.

Le $k^{ième}$ pas.

Au début de la $k^{ème}$ itération, on a une partition $\mathcal{M}_{k-1}$ de l'ensemble M_0. On va maintenant prendre un élément de $\mathcal{M}_{k-1}$ et le partitionner. On aura ainsi une partition plus raffinée de M_0.

Pourquoi l'étape k1?

Soit un élément $M_{k-1,i}$ de la partition tel que

$$\alpha_{k-1} \leq \beta_{k-1,i}.$$

Comme $\beta_{k-1,i} \leq \min f(M_{k-1,i} \cap D)$, on a que

$$f(x^{k-1}) \leq \min f(M_{k-1,i} \cap D).$$

Donc la valeur la plus faible dans D trouvée jusqu'à présent est inférieure ou égale à

toute valeur de $M_{k-1,i} \cap D$. On ne peut donc plus descendre la valeur de f dans D pour cet élément $M_{k-1,i}$ de la partition. On peut ainsi l'éliminer.

$\mathcal{M}_{k-1}$ devient R_k en enlevant de tels éléments.

Pourquoi les étapes k2 et k3?

Ayant enlevé de $\mathcal{M}_{k-1}$ les éléments non intéressants, on choisit un élément M_{k-1,i_k} que l'on partitionne à l'étape k2.

De la même manière qu'au pas 0, on choisit $s_{k,i}$ et $\beta_{k,i}$, et on calcule $\alpha_{k,i}$ avec les conditions

- $\beta_{k-1,i_k} \leq \beta_{k,i}$ (pour garantir le fait que la suite (β_k) soit croissante)
- $s_{k-1,i_k} \subset \bigcup_{i \in \mathcal{I}_k} s_{k,i}$ (pour garantir le fait que la suite (α_k) soit décroissante).

Pourquoi l'étape k4?

D'abord, la nouvelle partition est notée $\mathcal{M}_k$ et est définie par

$$\mathcal{M}_k = \left(R_k \setminus M_{k-1,i_k}\right) \cup \mathcal{M}_{k-1,i_k}.$$

Donc, on remplace l'élément partitionné par les ensembles qui le partitionnent.

On calcule $\min_{i \in I_k} \alpha_k$ et $\min_{i \in I_k} \beta_k$ en espérant que $\alpha_k < \alpha_{k-1}$ et que $\beta_{k-1} < \beta_k$. On choisit enfin x^k tel que $f(x^k) = \alpha_k$.

Le critère d'arrêt.

L'étape k5 dit de stopper l'algorithme quand $\alpha_k = \beta_k = f(x^k)$.

En pratique, on se donnera un réel $\varepsilon > 0$, représentant la tolérance. La différence $\alpha_k - \beta_k$ mesurant la proximité à la solution $f(\bar{x})$. On stoppera l'algorithme quand :

$$\alpha_k - \beta_k \leq \varepsilon.$$

3. CONVERGENCE de l'ALGORITHME.

Comme théorème de convergence, on obtient que tout point d'accumulation de la suite (x^k) généré par l'algorithme est solution du problème (P).

Pour établir ce théorème, on doit imposer deux règles sur la façon de construire les partitions.

3.1. Règle de sélection.

A l'étape k2, on doit préciser comment on choisit l'élément M_{k-1, i_k} à partitionner.

La règle de sélection adoptée est la suivante: on choisit M_{k-1,i_k} tel que

$$M_{k-1,i_k} \in \arg\min \{ \beta(M) : M \in R_k \}$$

c'est-à-dire

$$\beta(M_{k-1,i_k}) = \beta_{k-1}$$

et cela après au moins un nombre fini d'itérations.

Pratiquement, on appliquera cette règle à toutes les itérations.

3.2. <u>règle de consistance.</u>

On a vu que l'idée de base de l'algorithme était d'encadrer la solution. On dit que cette opération " d'encadrement " est consistante si :

Pour toute suite décroissante infinie $(M_{k_j,i_{k_j}})_j$ d'ensembles de la partition (c'est-à-dire telle que $M_{k_{j+1},i_{k_{j+1}}} \subset M_{k_j,i_{k_j}}$), on ait que

$$\lim_{j \to \infty} (\alpha_{k_j,i_{k_j}} - \beta_{k_j,i_{k_j}}) = 0.$$

3.3. <u>remarque.</u>

On note respectivement ces deux règles par (S) et (C).

3.4. Théorème de convergence.

Soit $\bar{x} \in \arg\min f(D)$.

D compact de $\mathbb{R}^m$ et $f : \mathbb{R}^m \to \mathbb{R}$ continue.

Alors, en appliquant l'algorithme, on a les cinq propriétés suivantes :

(a) Les points générés par l'algorithme sont admissibles : $x^k \in D \quad \forall k \geq 1$.

(b) Les suites (α_k) et (β_k) sont respectivement décroissante et croissante. De plus, elles encadrent la solution :

$$\beta_{k-1} \leq \beta_k \leq \min f(D) \leq \alpha_k \leq \alpha_{k-1} \quad \forall k \geq 1$$

(c) On décroit à chaque itération :

$$f(x^{k-1}) \geq f(x^k) \quad \forall k \geq 1.$$

(d) Si l'algorithme s'arrête à l'étape k Alors $f(x^k) = f(\bar{x})$. Dans ce cas, on a la solution après un nombre fini d'itérations.

(e) Si l'algorithme n'est pas terminé après un nombre fini d'itérations, si on applique la règle de sélection (S) et si l'opération d'encadrement est consistante, Alors tout point d'accumulation x de la suite (x^k) est solution du problème : $f(\bar{x}) = f(x)$.

Démonstration de (a).

Soit $k \geq 1$ fixé.

On choisit au $k^{ième}$ pas $x^k \in \arg\min \{ f(x^{k,i}) : i \in I_k \}$

avec

- $x^{k,i} \in \arg\min f(S_{k,i})$
- $S_{k,i} \subset D$.

Dès lors, on a bien que

$$x^k \in D \quad \forall k \geq 1.$$

Démonstration de (b).

1°) $\beta_{k-1} \leq \beta_k$

Rappelons que $\beta_{k-1} \triangleq \min_{i \in I_{k-1}} \beta_{k-1,i}$

et $\beta_k \triangleq \min_{i \in I_k} \beta_{k,i}$.

Comparons les deux ensembles suivants:

$A = \{ \beta_{k-1,i} : i \in I_{k-1} \}$ c'est-à-dire formé des β relatifs à la $k-1^{ème}$ partition

$N = \{ \beta_{k,i} : i \in I_k \}$ c'est-à-dire formé des β relatifs à la $k^{ème}$ partition.

L'ensemble N est formé des mêmes éléments que A sauf que l'élément β_{k-1,i_k} a disparu et a été remplacé par les $\beta_{k,i}$ avec $i \in J_k$ construits à l'étape k3 tels que $\beta_{k-1,i_k} \leq \beta_{k,i}$ $\forall i \in J_k$.

On a donc remplacé β_{k-1,i_k} par des éléments qui lui sont supérieurs ou égaux pour former

l'ensemble N. Comme β_{k-1} et β_k sont, respectivement, les éléments minimaux de A et de N, on peut conclure.

2°) $\beta_k \leq f(\bar{x})$.

Rappelons que $f(\bar{x}) = \min f(D)$

et $\beta_k = \min_{i \in I_k} \beta_{k,i}$

avec $\beta_{k,i} \leq \min f(D \cap M_{k,i}) \quad \forall i \in I_k$.

En vertu de la définition de β_k, on peut donc écrire que

$$\beta_k \leq \min f(D \cap M_{k,i}) \quad \forall i \in I_k .$$

De plus, les éléments $M_{k,i}$, $i \in I_k$, forment à chaque étape k une partition finie de l'ensemble de départ M_0. On en déduit donc

$$\beta_k \leq \min f(D).$$

3°) $f(\bar{x}) \leq \alpha_k$.

Rappelons que $f(\bar{x}) = \min f(D)$

et $\alpha_k = \min_{i \in I_k} \alpha_{k,i}$.

On doit montrer que

$$f(\bar{x}) \leq \alpha_k ,$$

ce qui revient à prouver l'inégalité suivante:

$$\min f(D) \leq \alpha_{k,i} \quad \forall i \in I_k \qquad (1).$$

Or, nous avons choisi $S_{k,i} \subset D \ \forall i \in I_k$ et donc

$$\min f(D) \leq \min f(S_{k,i}) \quad \forall i \in I_k .$$

En vertu de la définition de $\alpha_{k,i}$, (1) est démontré.

4°) $\alpha_k \leq \alpha_{k-1}$.

Rappelons que $\alpha_k \triangleq \min_{i \in I_k} \alpha_{k,i}$

et $\alpha_{k-1} \triangleq \min_{i \in I_{k-1}} \alpha_{k-1,i}$.

Par un raisonnement analogue à celui tenu en 1°), montrons que $\alpha_k \leq \alpha_{k-1}$.

Comparons dans ce but les deux ensembles suivants:

$$A = \{ \alpha_{k-1,i} : i \in I_{k-1} \}$$

$$N = \{ \alpha_{k,i} : i \in I_k \}.$$

A est pris entier pour former N sauf que l'élément α_{k-1,i_k} est remplacé par les $\alpha_{k,i}$ construits à l'étape k3 et vérifiant

$$\alpha_{k,i} = \min \{ f(S_{k,i}) \} \quad \forall i \in J_k.$$

Comme $S_{k-1,i_k} \subset \bigcup_{i \in J_k} S_{k,i}$, on a bien que le minimum de l'ensemble N est plus petit ou égal au minimum de l'ensemble A.

Démonstration de (c).

Prouvons que $f(x^k) \leq f(x^{k-1}) \quad \forall k = 1, 2, \ldots$

On a vu en (b) que $\alpha_k \leq \alpha_{k-1}$. Comme, par définition, $\alpha_k = f(x^k)$, la démonstration est immédiate.

Démonstration de (d).

Supposons que l'algorithme s'arrête à l'étape k.

Montrons que $f(x^k) = f(\bar{x})$.

On a vu en (b) que $\beta_k \leq f(\bar{x}) \leq d_k$. D'autre part, le critère d'arrêt est $d_k = \beta_k$ et donc la preuve est immédiate.

Démonstration de (e).

Supposons que l'algorithme ne se termine pas après un nombre fini d'itérations.

. Comme par hypothèse D est compact, la suite (x^k) de D possède un point d'accumulation.

Soit x un point d'accumulation de cette suite.

Soit $(x^k : k \in I)$ la sous-suite convergeant vers ce point x.

Par continuité de f, on a

$$\lim_{k \in I} f(x^k) = f(x).$$

. On a construit la suite $(d_k) = (f(x^k))$. On a vu en (b) que cette suite est décroissante et bornée inférieurement par $f(\bar{x})$.

On a aussi vu en (b) que la suite (β_k) est croissante et bornée supérieurement par $f(\bar{x})$.

On peut dès lors écrire que

$$\beta = \lim_{k \to \infty} \beta_k \leq f(\bar{x}) \leq \lim_{k \to \infty} \alpha_k = \alpha \qquad (1).$$

. La thèse peut donc s'écrire

(c) et (s) impliquent $\alpha = \beta$.

. On sait d'autre part que la règle (s) n'est pas appliquée à toutes les itérations.

Soit $\{ M_{k_j, i_{k_j}} \}$ la suite d'ensembles de la partition pour lesquels (s) est appliqué, c'est-à-dire que chaque élément de cette suite a été partitionné par l'algorithme.

Voyons que cette suite $\{ M_{k_j, i_{k_j}} \}$ contient une sous-suite emboitée. On la notera elle-même $\{ M_{k_j, i_{k_j}} \}$.

En effet, chaque partition est finie et on a une infinité de vecteurs x^k. Un élément de la partition doit donc contenir une infinité de vecteurs x^k. Celà implique donc qu'il a été partitionné, par-tition toujours finie. On peut continuer le raisonnement et ainsi trouver une suite emboitée.

. Comme (s) est appliqué pour cette suite emboitée $\{ M_{k_j, i_{k_j}} \}$, on a que:

$$\beta_{k_j} = \beta(M_{k_j, i_{k_j}}) = \beta_{k_j, i_{k_j}}. \qquad (2)$$

D'autre part, par définition de α_{k_j}, on peut écrire

$$\alpha_{k_j, i_{k_j}} \geqslant \alpha_{k_j}. \tag{3}$$

De plus, la règle (c) assure que

$$\lim_{j \to \infty} \left(\alpha_{k_j, i_{k_j}} - \beta_{k_j, i_{k_j}} \right) = 0. \tag{4}$$

Rassemblant (1) - (4) :

$$\begin{aligned} 0 &\overset{(4)}{=} \lim_{j \to \infty} \left(\alpha_{k_j, i_{k_j}} - \beta_{k_j, i_{k_j}} \right) \\ &\overset{(2)}{=} \lim_{j \to \infty} \left(\alpha_{k_j, i_{k_j}} - \beta_{k_j} \right) \\ &\overset{(3)}{\geqslant} \lim_{j \to \infty} \left(\alpha_{k_j} - \beta_{k_j} \right) \\ &\overset{(1)}{=} \alpha - \beta \overset{(1)}{\geqslant} 0. \end{aligned}$$

En lisant, cette suite d'inégalités, on peut conclure que $\alpha = \beta$.

. On a ainsi par (1) que $\alpha = f(\bar{x}) = \beta$

et donc

$$\lim_{k \in I} f(x^k) = f(\bar{x}),$$

x est donc bien solution.

Ce qui achève la démonstration du théorème.

4. CONDITION SUFFISANTE pour la CONSISTANCE.

Le point (e) du théorème de convergence demande que l'opération d'encadrement

soit consistante. On va donc essayer de trouver des conditions pour assurer la règle de consistance.

4.1. Rappel : trois définitions.

Soit (T_m) une suite d'ensembles de $\mathbb{R}^k$.

1. On définit la limite supérieure de (T_m) par

$$\overline{\lim_{m\to\infty}} T_m \equiv \{ x \in \mathbb{R}^k : x = \lim_{j\to\infty} x_{m_j}$$

avec . $m_j \in \mathbb{N}$ $\forall j$
. $(m_j)_j$ infinie
. $x_{m_j} \in T_{m_j}$ $\forall j$ }.

2. On définit la limite inférieure de (T_m) par

$$\underline{\lim_{m\to\infty}} T_m \equiv \{ x \in \mathbb{R}^k : x = \lim_{m\to\infty} x_m$$

avec $x_m \in T_m$ $\forall m$ sauf à un nombre fini }.

3. Un ensemble T de $\mathbb{R}^k$ est la limite de (T_m) si $T = \underline{\lim}_{m\to\infty} T_m = \overline{\lim}_{m\to\infty} T_m$.

On note alors $T = \lim_{m\to\infty} T_m$ ou $T_m \longrightarrow T$.

4.2. Lemme.

Avant de présenter une condition suffisante pour la règle de consistance, démontrons le lem-

me suivant :

Soit T, un ensemble non vide de $\mathbb{R}^k$, compact.

Soit (T_m) avec $\forall m$: T_m compact de $\mathbb{R}^k$,

tels que $\tilde{T} \supseteq T_m \supseteq T \quad \forall m \in \mathbb{N}$

avec $\tilde{T} \subset \mathbb{R}^k$ compact

et tels que $T_m \longrightarrow T$.

Soit $f : \mathbb{R}^k \longrightarrow \mathbb{R}$ continue

Alors, on a que

$$\min f(T_m) \xrightarrow[m \to \infty]{} \min f(T).$$

Démonstration

. Tous les ensembles considérés sont compacts, et f est continue, tous les minima existent donc.

. Soit $x_m \in \arg\min f(T_m) \ \forall m$.

On a $x_m \in T_m \ \forall m$, et donc $x_m \in \tilde{T} \ \forall m$, avec l'ensemble $\tilde{T}$ compact.

La suite (x_m) a un point d'accumulation x c'est-à-dire

$$\exists (m_j)_j \text{ telle que } \lim_{j \to \infty} x_{m_j} = x$$

et donc $x \in \overline{\lim_{m \to \infty}} T_m$.

Or, on a par hypothèse que $T_m \longrightarrow T$, c'est-à-dire $T = \overline{\lim_{m \to \infty}} T_m = \underline{\lim_{m \to \infty}} T_m$.

Donc, $x \in T$ et ainsi

$$f(x) \geq \min f(T). \qquad (1).$$

Voyons dans un second temps que

$$f(x) \leq \min f(T). \qquad (2).$$

Nous avons par hypothèse que $T \subseteq T_m \ \forall m$. On déduit que

$$\min f(T_m) \leq \min f(T) \ \forall m.$$

Or,

$$x = \lim_{j \to \infty} x_{m_j}.$$

Et donc, en vertu de la continuité de f, on obtient que

$$f(x) = \lim_{j \to \infty} f(x_{m_j}).$$

avec

$\forall j : x_{m_j} \in \arg\min f(T_{m_j})$, ce qui peut s'écrire

$$\forall j : f(x_{m_j}) = \min f(T_{m_j}) \leq \min f(T).$$

Par conséquent,

$$f(x) = \lim_{j \to \infty} f(x_{m_j}) \leq \min f(T)$$

et (2) est démontré.

Rassemblant (1) et (2), on obtient que

$$f(x) = \min f(T)$$

avec x un point d'accumulation de (x_m) et

$x_m \in \arg\min f(T_m)$ $\forall m$.

Donc, tout point d'accumulation de la suite $(\min f(T_m))$ vaut $\min f(T)$.

4.3. Réalisation.

Pratiquement, les éléments de la partition seront des polytopes convexes, des simplexes, des ensembles rectangulaires, ...

Au fur et à mesure des étapes, la partition de l'ensemble de départ sera de plus en plus fine. On engendrera donc une suite (M_m) décroissante, convergeant vers un ensemble compact M, c'est-à-dire

$$M := \bigcap_m M_m$$

avec $M \cap D$ non vide, ce qui sera noté de la façon suivante :

$$M_m \downarrow M.$$

D'autre part, les bornes $\alpha(M_m)$ construites par l'algorithme devront converger vers $\min f(M \cap D)$, M pouvant être un singleton.

Le lemme suivant donnera une condition suffisante pour que l'opération d'encadrement soit consistante.

Lemme 2

Soit $\mathbb{D}$, compact de $\mathbb{R}^k$

$f : \mathbb{R}^k \longrightarrow \mathbb{R}$ fonction continue.

Si pour toute suite décroissante d'ensembles M_m compacts de la partition, on a que :

1. $M_m \downarrow M$

   avec M compact, d'intersection non vide avec $\mathbb{D}$.

2. Il existe une suite d'ensembles compacts (T_m) vérifiant

   . $M_m \cap \mathbb{D} \subseteq T_m \subseteq M_m \quad \forall m$

   . $T_m \longrightarrow M \cap \mathbb{D}$.

3. $\min f(T_m) \leq \beta(M_m) \leq \min f(M_m \cap \mathbb{D})$.

4. $\alpha(M_m) \xrightarrow[m \to \infty]{} \min f(M \cap \mathbb{D})$.

Alors l'opération d'encadrement est consistante.

Démonstration

Considérons (M_m) une suite décroissante d'ensembles compacts vérifiant 1. – 4.

Remarquons d'abord que, par l'hypothèse 1.,

$$M_m \downarrow M.$$

On peut donc écrire que

$$M_m \cap \mathcal{D} \downarrow M \cap \mathcal{D}.$$

Notons T l'ensemble $M \cap \mathcal{D}$. Comme on peut le constater, cet ensemble T est compact, non vide, et vérifie

$$M_m \cap \mathcal{D} \downarrow T. \qquad (1).$$

Ayant établi ces faits, revenons à la thèse qui consiste à voir que sous les hypothèses 1.-4., l'opération d'encadrement est consistante.

En vertu de la définition de la consistance, la thèse sera démontrée si

$$\lim_{m \to \infty} \beta(M_m) = \lim_{m \to \infty} \alpha(M_m) = \min f(T).$$

Voyons dans un premier temps que,

$$\lim_{m \to \infty} \alpha(M_m) = \min f(T). \qquad (2).$$

L'hypothèse 4. nous assure que

$$\lim_{m \to \infty} \alpha(M_m) = \min f(M \cap \mathcal{D}),$$

et, en vertu de la notation introduite, (2) est assuré.

Dans un second temps, prouvons que

$$\lim_{m \to \infty} \beta(M_m) = \min f(T). \qquad (3).$$

L'hypothèse 2. nous donne l'existence d'une suite d'ensembles compacts (T_m) vérifiant

$$\cdot\ M_m \cap \mathcal{D} \subseteq T_m \subseteq M_m \quad \forall m$$

$$\cdot\ T_m \longrightarrow M \cap D.$$

On en déduit que

$$T_m \longrightarrow T \quad \text{et} \quad T \subseteq T_m \ \forall m.$$

En prenant $\tilde{T} = M_1$, nous avons toutes les hypothèses nécessaires à l'application du lemme 1. En vertu de ce lemme, nous avons que

$$\min f(T_m) \xrightarrow[m \to \infty]{} \min f(T). \qquad (4)$$

D'autre part, on déduit de l'hypothèse 3. de notre lemme que

$$\min f(T_m) \leq \beta(M_m) \leq \min f(T). \qquad (5)$$

Passons à la limite dans l'expression (5). On a par (4) que

$$\min f(T_m) \xrightarrow[m \to \infty]{} \min f(T),$$

et, en considérant, la suite constante $(\min f(T))$, on en déduit que

$$\lim_{m \to \infty} \beta(M_m) = \min f(T),$$

ce qui est bien la relation (3).

Rassemblant (2) et (3), en vertu de la définition de la constance, on a la thèse.

Chapitre 2 :

Application au cas de contraintes séparables quadratiques avec contraintes de bornes

Motivation.

La méthode présentée au chapitre 1 a été conçue afin de résoudre le problème :

$$(P) \begin{cases} \min f(D) \\ \text{avec } D \text{ compact de } \mathbb{R}^m \\ \qquad f : \mathbb{R}^m \longrightarrow \mathbb{R} \text{ continue.} \end{cases}$$

Dans ce second chapitre, on adaptera l'algorithme à un cas particulier du problème (P) : le cas où le compact D est déterminé par l'intersection d'un m-rectangle et du domaine déterminé par les contraintes $g_i(x) \leq 0$, $i = 1, \dots, m$, où chaque g_i est une fonction quadratique séparable. De plus, la fonction f sera concave.

Position du problème.

On se propose de résoudre le problème suivant :

$$\min f(x)$$

$$\text{s.c.} \begin{cases} g_i(x) = \sum_{k=1}^{m} \left(\frac{1}{2} p_{ik} x_k^2 + q_{ik} x_k + r_{ik} \right) \leq 0 \qquad i = 1 \dots m \\ \underline{m}_k \leq x_k \leq \bar{m}_k \qquad k = 1 \dots m. \end{cases}$$

On note ce problème (Q).

avec

. $\forall k = 1 \dots m \quad \forall i = 1 \dots m$: p_{ik}, q_{ik}, r_{ik}, $\underline{m}_k$ et $\bar{m}_k$ réels donnés

. $f(x)$ fonction concave à valeur réelle définie sur un ouvert convexe contenant le rectangle $M_0 = \{ x : \underline{m} \leq x \leq \bar{m} \}$

avec

$$\underline{m} = \begin{pmatrix} \underline{m}_1 \\ \vdots \\ \underline{m}_m \end{pmatrix} \quad \text{et} \quad \bar{m} = \begin{pmatrix} \bar{m}_1 \\ \vdots \\ \bar{m}_m \end{pmatrix}$$

. les fonctions f et g_i sont continues.

Notons D l'ensemble formé par les contraintes du problème (Q).

Cherchons $\min f(D)$.

1. STRATEGIE.

Pour résoudre (Q), on fait appel à un algorithme de type Branch-and-Bound. L'idée générale de la méthode est basée sur les principes suivants :

1. Construction d'une partition de plus en plus raffinée du rectangle $M_0 \supset D$

2. Détermination, pour chaque élément M de la partition construit par l'algorithme, d'une borne inférieure pour le minimum de f sur ce M.
3. Détermination à chaque étape de l'algorithme d'une borne supérieure $\alpha_k \geq \min f(D)$
4. Se donner des règles de suppression visant à éliminer des éléments M de la partition de M_0 vérifiant $M \cap D = \phi$ ou ne pouvant contenir le minimum recherché.

Avant de présenter un algorithme adapté à ce problème particulier, détaillons ces quatre points.

2. PARTITION.

2.1. Donnée d'un n-rectangle.

Un n-rectangle est un ensemble de type

$$M := \{x : a \leq x \leq b\} \quad \text{avec } a < b$$

$$a, b \in \mathbb{R}^n.$$

Un n-rectangle est donc déterminé univoquement par la donnée du sommet inférieur a et du sommet supérieur b.

On définit le diamètre de M par $\|b-a\|$ avec $\|.\|$ la norme euclidienne sur $\mathbb{R}^m$, et on le note $d(M)$.

2.2. Définition.

Soit (M_q) une suite infinie décroissante de m-rectangles construits par l'algorithme. La procédure la construisant est dite exhaustive si

$$\lim_{q \to \infty} d(M_q) = 0.$$

2.3. Bissection.

Une partition simple consiste à diviser un m-rectangle $M = \{x : a \leq x \leq b\}$ en deux m-rectangles à l'aide d'un hyperplan passant par $\frac{a+b}{2}$ et perpendiculaire au plus long côté de M. Cette opération est appelée bissection.

3. BORNES INFERIEURES et SUPERIEURES.

Propriété.

Soit M un m-rectangle.

Notons $V(M)$ l'ensemble des sommets de M.

Comme f est concave, on a que

$$\min f(M) = \min f(V(M)) .$$

Donc le minimum de f sur M est atteint en un des sommets du m-rectangle.

Détermination d'une borne inférieure.

Comme on veut que $\beta(M) \leq \min f(M)$ pour tout élément M de la partition, on prendra $\beta(M) = \min f(V(M))$. On aura ainsi que

$$\beta(M) = \min f(M) \leq \min f(M \cap D)$$
$$\text{si } M \cap D \neq \phi$$

Détermination d'une borne supérieure.

On prend α_r comme borne supérieure au minimum de f sur D : $\alpha_r = \min f(S_r)$

avec S_r l'ensemble des points admissibles trouvés jusqu'à l'étape r.

On a en fait $\alpha_r = f(x^r)$ avec x^r le meilleur point admissible obtenu jusqu'à présent.

Si S_r est vide, on pose alors $\alpha_r = +\infty$.

4. REGLES de SUPPRESSION.

4.1. Ensembles inintéressants.

On éliminera des éléments M de la par-

tition tels que

$$\beta(M) > \alpha_s$$

pour une itération s pour les mêmes raisons qu'au chapitre 1.

4.2. Ensembles n'ayant aucune intersection avec D.

Il sera plus complexe de supprimer les m-rectangles M tels que $M \cap D = \phi$. En effet, un m-rectangle est donné par ses sommets $V(M)$ et la seule chose que l'on puisse évaluer est $V(M) \cap D$. Mais si cette intersection est vide, on ne peut évidemment pas conclure que $M \cap D = \phi$.

Une règle de suppression notée (DR) sera néanmoins introduite, permettant de détecter certains éléments n'ayant aucune intersection avec D.

Vu la forme des fonctions-contraintes g_i, on peut voir que celles-ci sont Lipschitziennes :

$\forall i = 1 \dots m \quad \exists L_i = L_i(M) > 0$ tel que

$$|g_i(z) - g_i(x)| \leq L_i \, \|z - x\| \quad \forall x, z \in M.$$

Une borne supérieure pour chaque L_i étant donnée par $A_i = \max \{ \|\nabla g_i(y)\| : y \in M \}$.

Cherchons à évaluer A_i.

On a successivement que

$$g_i(y) = \sum_{k=1}^{m} \left(\frac{1}{2} p_{ik} y_k^2 + q_{ik} y_k + r_{ik} \right)$$

$$\nabla g_i(y) = \begin{pmatrix} p_{i1} y_1 + q_{i1} \\ \vdots \\ p_{im} y_m + q_{im} \end{pmatrix}$$

$$\| \nabla g_i(y) \| = \sqrt{\sum_{k=1}^{m} (p_{ik} y_k + q_{ik})^2}$$

et $$A_i = \max_{y \in M} \| \nabla g_i(y) \| .$$

Comme $M = \{ x : a \leq x \leq b \}$, on a que

$$A_i = \max_{y \text{ tq } a_k \leq y_k \leq b_k} \sqrt{\sum_{k=1}^{m} (p_{ik} y_k + q_{ik})^2}$$

$$= \sqrt{\sum_{k=1}^{m} \max_{a_k \leq y_k \leq b_k} | p_{ik} y_k + q_{ik} |^2} .$$

On voit donc que

$$A_i = \left\{ \sum_{k \in N_i^1} (p_{ik} a_k + q_{ik})^2 + \sum_{k \in N_i^2} (p_{ik} b_k + q_{ik})^2 \right\}^{1/2}$$

avec

$$N_i^1 = \left\{ k : -\frac{q_{ik}}{p_{ik}} \geq \frac{a_k + b_k}{2} \right\}$$

$$N_i^2 = \left\{ k : -\frac{q_{ik}}{p_{ik}} < \frac{a_k + b_k}{2} \right\} .$$

On introduit alors la règle suivante notée (DR) (Deletion Rule) :

> Enlever de la partition tout élément M vérifiant la propriété :
> $\exists\, i \in \{1 \dots m\}$ tel que
> $$\max \{ g_i(x) : x \in V'(M) \} - A_i\, d(M) > 0$$
> où
> $$d(M) = \| a - b \|$$
> $$V'(M) = \{ a, b \}.$$

Un ensemble M éliminé par la règle (DR) est en fait non admissible.

En effet :

On a $\forall x, z \in M : |g_i(z) - g_i(x)| \leq L_i \| z - x \|$
$\forall i = 1 \dots m$.

Comme $A_i \geq L_i$, on a $\forall i = 1 \dots m$ que

$$g_i(z) \geq g_i(x) - L_i \| z - x \| \quad \forall x, z \in M$$

$$g_i(z) \geq g_i(x) - A_i \| z - x \| \quad \forall x, z \in M$$

et donc

$$g_i(z) \geq g_i(x) - A_i \cdot d(M) \quad \forall x, z \in M.$$

Comme M est éliminé par la règle (DR) :

$\exists\, x \in V'(M)$ $\exists\, i \in \{1 \dots m\}$ tels que

$$g_i(x) - A_i\, d(M) > 0$$

et donc $g_i(z) > 0$ $\forall z \in M$.

5. L'ALGORITHME

Ayant déterminé la stratégie, on peut présenter l'algorithme adapté à la résolution du problème (Q).

Pas 0

On a M_0.

Prendre $S_{M_0} = V(M_0) \cap D \subset D$.

Calculer

$$\beta(M_0) = \min f(V(M_0)) = \beta_0$$

$$\alpha(M_0) = \min f(S_{M_0}) = \alpha_0$$

$(\alpha_0 = +\infty \text{ si } S_{M_0} = \emptyset)$.

Poser $I_0 = \{M_0\}$.

Si $\alpha_0 < \infty$

Alors prendre $x^0 \in \arg\min f(S_{M_0})$

(donc $f(x^0) = \alpha_0$).

Si $\alpha_0 - \beta_0 \leq \varepsilon$ (avec $\varepsilon > 0$ donné)

Alors STOP

Sinon poser $r = 1$ et aller au pas r.

Pas r

Au début de l'étape r, on a la partition rectangulaire I_{r-1} de M_0.

$\forall M \in I_{r-1}$, on a $S_M = V(M) \cap D \subset M \cap D$ et on a

$$\beta(M) = \min f(M) \leq \alpha(M).$$

De plus, on a les bornes β_{r-1} et α_{r-1} satisfaisant

$$\beta_{r-1} \leq \min f(D) \leq \alpha_{r-1}.$$

D'autre part, si $\alpha_{r-1} < \infty$, on a un point $x^{r-1} \in D$ satisfaisant

$$f(x^{r-1}) = \alpha_{r-1}$$

avec x^{r-1} le meilleur point admissible obtenu jusqu'à présent.

Etape r1

Enlever tout $M \in I_{r-1}$ tel que $\beta(M) \geq \alpha_{r-1}$.

Soit R_r la collection des n-rectangles restants.

Etape r2

Choisir une collection non vide $P_r \subset R_r$ telle que

$$\arg\min \{ \beta(M) : M \in I_{r-1} \} \subset P_r$$

et partitionner en n-rectangles (bissection par exemple) tout Elément de P_r.

Soit P_r' la collection de tous les nouveaux éléments de la partition.

Etape r3

Construire I_r' la collection des éléments de P_r' non éliminés par (DR).

Etape r4

A tout élément $M \in I_r'$, lui associer

$$S_M = V(M) \cap D \subset M \cap D.$$

et

$$\alpha(M) = \min f(S_M)$$

(on pose $\alpha(M) = +\infty$ si $S_M = \phi$)

$$\beta(M) = \min f(V(M)).$$

Etape r5

Poser $I_r = (R_r \setminus P_r) \cup I_r'$.

Calculer

$$\alpha_r = \inf \{ \alpha(M) : M \in I_r \}$$

$$\beta_r = \min \{ \beta(M) : M \in I_r \}.$$

Si $\alpha_r < +\infty$

Alors prendre $x^r \in D$ tel que $f(x^r) = \alpha_r$.

Etape r6

Si $\alpha_r - \beta_r \leq \epsilon$

Alors STOP

Sinon $r = r + 1$ et recommencer le pas r.

6. COMMENTAIRES.

Désignons l'algorithme qui vient d'être présenté par l'algorithme 2. Il est l'adaptation de l'algorithme 1 au problème (Q). Néanmoins, il présente certaines différences. Passons en revue certaines d'entre-elles.

6.1. Règle de sélection.

L'algorithme 1 demande à l'étape k2 de choisir un ensemble à partitionner. La

règle de sélection (S) avait été introduite : elle déterminait, "après un nombre fini d'itérations," l'ensemble à partitionner.

L'algorithme 2, dans le prolongement de l'idée de la règle de sélection, demande de partitionner, à chaque étape, au moins les éléments atteignant β_{r-1}.

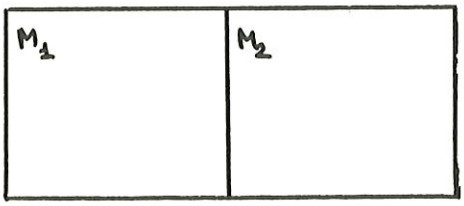
6.2. La suite (β_r).

L'algorithme 1 demandait à la $k^{ième}$ étape de choisir, pour chaque nouvel élément de la partition, c'est-à-dire pour chaque $i \in J_k$, un nombre $\beta_{k,i}$ vérifiant

$$\beta_{k-1,i_k} \leq \beta_{k,i} \cdot \qquad (1)$$

L'algorithme 2 n'impose pas de condition équivalente. En fait, c'est une propriété de l'algorithme :

En effet, soit $M \in P_r$. Cet élément M est partitionné au $r^{ième}$ pas en deux rectangles M_1 et M_2. On a donc la situation suivante :



Comme $M_i \subset M$, $i=1,2$, on peut écrire

$$\min f(M) \leq \min f(M_i) \qquad i=1,2 .$$

Or, comme f est concave, nous avons immédiatement que

$$\min f(V(M)) \leq \min f(V(M_i)) \qquad i=1,2 .$$

Par définition de $\beta(M)$, cette inégalité peut s'écrire

$$\beta(M) \leq \beta(M_i) \qquad i=1,2 .$$

Ainsi, la condition (1) n'est plus à imposer.

6.3. La suite (d_k).

L'algorithme 1 demandait à la $k^{ième}$ étape de choisir, pour chaque nouvel élément de la partition, c'est-à-dire pour chaque $i \in J_k$, un sous-ensemble $S_{k,i}$ vérifiant

$$S_{k-1,i_k} \subset \bigcup_{i \in J_k} S_{k,i} . \qquad (2)$$

Cette propriété permettait de démontrer le caractère décroissant de la suite (d_k).

L'algorithme 2 impose le choix suivant du sous-ensemble S_M :

$$S_M = V(M) \cap D .$$

L'algorithme 2 n'impose pas de condition équivalente à (2), mais le choix particulier des S_M garantit cette propriété.

En effet, soit $M \in P_{x}$, partitionné donc au $x^{ième}$ pas en deux m-rectangles: M_1 et M_2. Tout sommet de M sera un sommet de M_1 ou de M_2 par bissection. Donc, la condition (2) n'est pas à imposer pour l'algorithme 2, elle en est une propriété.

7. CONVERGENCE de l'ALGORITHME.

Avant d'établir un théorème de convergence, prouvons le lemme suivant :

7.1. Lemme.

> Si la procédure est exhaustive et si on applique la règle (DR)
> Alors toute suite infinie décroissante (M_q) d'ensembles de la partition construits par l'algorithme vérifie
>
> $$M_q \xrightarrow[q \to \infty]{} \{\bar{x}\} \quad \text{avec } \bar{x} \in D.$$

Démonstration

Soit (M_q) une suite décroissante d'éléments de la partition.

Comme la procédure de subdivision est exhaustive, on a que

$$\lim_{q \to \infty} d(M_q) = 0 \quad .$$

On en déduit l'existence d'un point $\bar{x}$ tel que

$$M_q \xrightarrow[q \to \infty]{} \{\bar{x}\} \qquad (1)$$

Pour achever la démonstration, il reste à voir que le point $\bar{x}$ appartient à D.

Pour tout ensemble M_q et pour chaque fonction g_i $(i = 1 \dots m)$, considérons la constante $A_i(M_q)$ telle que

$$A_i(M_q) \geq L_i(M_q)$$

avec $L_i(M_q)$: constante de Lipschitz de la fonction g_i sur l'ensemble M_q.

Comme la suite (M_q) est décroissante, nous ferons l'hypothèse supplémentaire que

$$A_i(M_{q+1}) \leq A_i(M_q)$$

et que

$$A_i(M_q) \leq A \qquad \forall q \quad \forall i = 1 \dots m .$$

Pour se mettre dans le cas du lemme, supposons que l'on applique la règle (DR) à chaque itération.

Supposons, par l'absurde, que $\bar{x} \notin D$. On examine l'expression

$$\max \{ g_i(x) : x \in V'(M_q) \} - A_i(M_q) \cdot d(M_q)$$

. quand q tend vers l'infini

$\forall i = 1 \dots m$

. pour toute suite d'ensembles non vide $V'(M_q) \subset V(M_q)$.

Or, nous avons (1) qui nous dit que

$$M_q \xrightarrow[q \to \infty]{} \{\bar{x}\}.$$

De plus, on a successivement que :

. les fonctions g_i sont continues $\forall i = 1 \dots m$

. $A_i(M_q)$ borné supérieurement $\forall q$ $\forall i = 1 \dots m$

. $d(M_q) \xrightarrow[q \to \infty]{} 0$.

On en déduit donc que

$$\max \{ g_i(x) : x \in V'(M_q) \} - A_i(M_q) \cdot d(M_q) \xrightarrow[q \to \infty]{} g_i(\bar{x})$$

$$\forall i = 1 \dots m . \qquad (2)$$

Comme on a supposé que $\bar{x} \notin D$, avec D l'ensemble déterminé par les contraintes du problème considéré (Q), on en déduit l'existence d'un indice $i \in \{1 \dots m\}$ tel que $g_i(\bar{x}) > 0$.

En vertu de (2) :

$\exists\, q_0 \quad \forall q \geq q_0$:

$$\max \{ g_i(x) : x \in V'(M_q) \} - A_i(M_q) \cdot d(M_q) > 0$$

pour l'indice i tel que $g_i(\bar{x}) > 0$.

Ce qui contredit le fait que l'on applique la règle (DR). L'hypothèse $\bar{x} \notin D$ était donc absurde.

7.2. Convergence de la méthode.

On établit la convergence de la méthode grâce au théorème suivant :

> Si l'algorithme est appliqué au problème (Q)
>
> Alors
>
> (i) Les β_r construits par l'algorithme vérifient :
>
> $$\beta := \lim_{r \to \infty} \beta_r = \min f(D).$$
>
> (ii) Si pour chaque élément M de la partition, on a $S_M \neq \phi$
>
> Alors
>
> . $\beta = \min f(D) = \lim_{r \to \infty} \alpha_r := \alpha$.
>
> . Chaque point d'accumulation de la suite (x^r) est solution.

Démonstration de (i)

On peut montrer par un raisonnement analogue à celui tenu pour démontrer la convergence de la méthode décrite au premier chapitre qu'il existe une suite $\{M_q\}$ décroissante d'ensembles de la partition avec

$$\beta(M_q) = \beta_q.$$

De plus, par construction, la suite (β_x) est mon décroissante, bornée supérieurement par le minimum de f sur D de sorte que

$$\beta := \lim_{x \to \infty} \beta_x \leq \min f(D). \qquad (1)$$

D'autre part, le lemme précédent a établi que

$$\lim_{q \to \infty} M_q = \{\bar{x}\} \qquad \text{avec } \bar{x} \in D.$$

Or, nous avons aussi par construction que

$$\beta_q = \beta(M_q) = \min f(V(M_q)) = f(V_q)$$
$$= \min f(M_q)$$

avec $V_q \in V(M_q)$.

Ayant la continuité de f, on peut écrire

$$\lim_{q \to \infty} \beta(M_q) = f(\bar{x}). \qquad (2)$$

Dès lors, rassemblant (1) et (2), on a que

$$\beta = f(\bar{x}) \leq \min f(D) := \min \{f(x) : x \in D\}.$$

Comme $\bar{x} \in D$, on en déduit

$$\beta = f(\bar{x}) = \min f(D).$$

Par conséquent, $\bar{x}$ est bien solution du problème, avec $\{M_q\} \xrightarrow[q \to \infty]{} \{\bar{x}\}$, ce qui achève la démonstration de (i).

Démonstration de (ii)

Comme demandé, supposons que l'ensemble

S_M soit non vide, pour chaque élément M de la partition, ce qui revient à demander $\alpha(M) < +\infty$ pour chacun des M.

Afin de pouvoir recommencer un raisonnement analogue à celui tenu pour la démonstration de convergence de la méthode décrite au chapitre 1, montrons que, pour toute suite (M_q) décroissante, on a :

$$\alpha(M_q) - \beta(M_q) \xrightarrow[q \to \infty]{} 0 .$$

Soit (M_q) une suite décroissante d'éléments de la partition engendrée par l'algorithme.

Le lemme précédent a établi que

$$M_q \xrightarrow[q \to \infty]{} \{\bar{x}\} \qquad \text{avec } \bar{x} \in D$$

et la première partie de ce théorème assure que

$$\beta(M_q) \xrightarrow[q \to \infty]{} f(\bar{x}) . \tag{3}$$

Or, par construction, on a que :

$$\alpha(M_q) = \min f(S_{M_q}) \tag{4}$$

$$\text{avec } S_{M_q} \subset M_q$$

$$S_{M_q} \neq \Phi .$$

Rassemblant (3) et (4), on peut affirmer que

$$\alpha(M_q) - \beta(M_q) \xrightarrow[q \to \infty]{} f(\bar{x}) \ .$$

On peut donc recommencer le raisonnement tenu au chapitre précédent et ainsi prouver la convergence de la méthode.

Chapitre 3 :

Implémentation.

1. Démarche suivie pour l'implémentation.

La méthode présentée au deuxième chapitre a été implémentée afin de résoudre tout problème de type :

$$\min f(x)$$

$$\text{s.c.} \begin{cases} g_i(x) \leq 0 & i = 1 \dots m \\ \underline{m} \leq x \leq \overline{m} \end{cases}$$

avec

- f, fonction concave de $\mathbb{R}^m$ à valeur réelle,
- g_i, $i = 1 \dots m$, fonctions de $\mathbb{R}^m$ à valeurs réelles, quadratiques séparables,
- $\underline{m}$ et $\overline{m}$, deux vecteurs de $\mathbb{R}^m$.

Dans l'implémentation de l'algorithme 2 vu précédemment, il reste à préciser un certain nombre de procédures.

La première est la manière de partitionner un élément M. La bissection avait été suggérée mais tout autre manière de partitionner un élément en divers n-rectangles était permise. Dans notre implémentation, deux partitionnements différents ont été testés : la bissection et la trisection.

Précisons la notion de <u>trisection</u> :

La trisection consiste à diviser un m-rectangle $M = \{x : a \leq x \leq b\}$ en trois m-rectangles à l'aide des 2 hyperplans : le premier passant par $\frac{a+b}{3}$ et le second passant par $\frac{2(a+b)}{3}$. Les deux hyperplans sont perpendiculaires au plus long côté de M.

Une seconde procédure à préciser est la manière de choisir à chaque étape les éléments à partitionner. Théoriquement, on demande de partitionner au moins l'(les) élément(s) atteignant la valeur minimale des $\beta(M)$, mais rien ne nous oblige à se limiter à ce choix. Par exemple, un autre choix est motivé par le critère d'arrêt : rappelons que l'algorithme est stoppé quand la condition $d_k - \beta_k \leq \varepsilon$ est vérifiée, la suite (β_k) étant croissante et la suite (d_k) décroissante. D'une étape $k-1$ à une étape k, on espère donc avoir $d_k < d_{k-1}$ et $\beta_{k-1} < \beta_k$. Le partitionnement des éléments atteignant la valeur minimale des $\beta(M)$ est réalisé dans le but de faire croître β_{k-1} en β_k. Mais on peut aussi espérer qu'en partitionnant en plus les éléments atteignant d_{k-1}, on arrive à faire décroître d_{k-1} en d_k.

D'autre part, rien ne permet de penser que seuls ces éléments sont intéressants. Une tactique serait de partitionner tous les éléments non éliminés par le par r1 ou par la règle (DR).

Ainsi, six algorithmes différents ont été implémentés. Ils diffèrent par la manière de partitionner et par le choix des éléments à partitionner à chaque étape r :

A. Bissection des éléments atteignant β_{r-1}.

B. Bissection des éléments atteignant β_{r-1} ou d_{r-1}.

C. Bissection de tous les éléments.

D. Trisection des éléments atteignant β_{r-1}.

E. Trisection des éléments atteignant β_{r-1} ou d_{r-1}.

F. Trisection de tous les éléments.

2. Exemples et résultats numériques.

La méthode a été implémentée en FORTRAN 77 sur VAX et testée sur plusieurs problèmes.

Des problèmes très simples ont d'abord été testés et d'emblée la méthode consistant à partitionner en trois n-rectangles tous les éléments non éliminés ou par r1 ou par la

règle (DR) s'est révélée peu efficace. Seuls les cinq algorithmes restant ont donc été comparés.

Pour chaque problème, les résultats suivants seront présentés et cela pour chacun des algorithmes:

EPS = la précision voulue sur la valeur de la fonction à la solution,

NBITER = le nombre d'itérations,

NBPART = le nombre d'éléments dans la partition,

SOL = la solution obtenue.

On désigne les cinq algorithmes suivant par les lettres introduites dans le premier paragraphe.

On présente maintenant quelques résultats numériques. Pour chaque problème, les cinq algorithmes ont été comparés pour EPS = 10^{-3}, et le plus performant est ensuite testé avec une valeur de EPS beaucoup plus petite.

Test 1

$$\min \; -x_1^2 - x_2^2$$

$$\text{s.c.} \begin{cases} x_1^2 + x_2 - 8 \leq 0 \\ -x_1^2 + x_2 - 4 \leq 0 \\ x_1 - x_2^2 - 2 \leq 0 \\ -4x_1 + x_2 - 4 \leq 0 \\ -3 \leq x_1 \leq 3 \\ 0 \leq x_2 \leq 8 \end{cases}$$

La solution exacte de ce problème est $(\sqrt{2}, 6)^T$. Le tableau de résultats est :

	eps	nbiter	nbpart	sol
A	0.001	148	89	1.414215 5.999938
B	0.001	125	108	1.414215 5.999938
C	0.001	104	89	1.414209 5.999988
D	0.001	101	110	1.414209 5.999988
E	0.001	37	169	1.414215 5.999984

La bissection ou la trisection à chaque étape des éléments atteignant la valeur minimales des $\beta(M)$ demandent le même nombre d'éléments dans la partition.

Si on teste l'algorithme A avec $EPS = 10^{-11}$, on obtient que

$$NBITER = 241$$

$$NBPART = 207$$

$$SOL = \begin{pmatrix} 1.414213562373 \\ 5.999999999999 \end{pmatrix}.$$

Le temps CPU dans ce cas est de 6.04 secondes.

Test 2

$$\min \; -x_1^2 - x_2^2$$

$$\text{s.c.} \begin{cases} -4x_1^2 + x_2 - 4 \leq 0 \\ 0.75\,x_1 - x_2 - 1.5 \leq 0 \\ -1.5 \leq x_1 \leq 2 \\ -3 \leq x_2 \leq 1 \end{cases}$$

Le domaine de contraintes est formé par deux ensemble disjoints. La solution de ce problème est $(-1.5, -2.625)^T$. Le tableau de résultats est:

	eps	mbiter	mbpart	sol
A	1.10^{-7}	121	68	-1.5 -2.625
B	1.10^{-7}	65	72	-1.5 -2.625
C	1.10^{-7}	90	102	-1.5 -2.624999
D	1.10^{-7}	90	104	-1.5 -2.624999
E	1.10^{-7}	62	71	-1.5 -2.625

La bissection des éléments atteignant la valeur minimale des $\beta(M)$ demande le moins

d'éléments dans la partition. Si on teste cet algorithme avec $eps = 10^{-11}$, on obtient

NBITER = 151

NBPART = 88

$SOL = (-1.5 \; ; \; -2.625)^T$.

Le temps CPU est de 3.54 secondes.

Test 3

$$\min x_1$$

$$\text{s.c.} \begin{cases} x_1^2 - x_2 + 4 \leq 0 \\ x_1^2 + x_2 - 8 \leq 0 \\ 4x_1 + x_2 - 8 \leq 0 \\ -0.5 x_1 + x_2 - 7 \leq 0 \\ -3 \leq x_1 \leq 3 \\ 3 \leq x_2 \leq 9 \end{cases}$$

La solution exacte de ce problème est $(-\sqrt{2}, 6)^T$. Du tableau des résultats qui suit, on constate de nouveau que la bissection à chaque étape des éléments atteignant la valeur minimale des $\beta(M)$ demande le moins d'éléments dans la partition.

	eps	nbiter	nbpark	sol
A	0.001	50	110	-1.4135 6.001
B	0.001	29	129	-1.4141 6.001
C	0.001	29	219	-1.4133 5.9995
D	0.001	23	215	-1.4139 5.9995
E	0.001	28	125	-1.4139 5.9995

L'algorithme A avec $eps = 10^{-11}$, nous donne que

NBITER = 177

NBPART = 507

$$SOL = \begin{pmatrix} -1.41421356236787 \\ 6.00000000001094 \end{pmatrix}.$$

Le temps CPU est de 10.44 secondes.

Test 4

$$\min \; -x_1^2 - x_2^2 - x_3^2$$

$$\text{s.c.} \begin{cases} 6x_1 + 10x_2 + 15x_3 - 30 \leq 0 \\ x_1^2 + x_2^2 + x_3^2 - 25 \leq 0 \\ -x_1 - x_2 - x_3 + 1 \leq 0 \\ 0 \leq x_1 \leq 6 \\ 0 \leq x_2 \leq 5 \\ 0 \leq x_3 \leq 3 \end{cases}$$

La solution exacte de ce problème est $(5,0,0)^T$.

Le tableau des résultats est :

	eps	nbiter	nbpart	sol
A	0.001	52	22	4.999694 0 0
B	0.001	45	42	4.999694 0 0
C	0.001	46	28	4.999983 0 0
D	0.001	23	28	4.999983 0 0
E	0.001	45	51	4.999694 0 0

De nouveau, la bissection à chaque étape de éléments atteignant la valeur minimale de $\beta(M)$ demande le moins d'éléments dans la partition. Si on teste cet algorithme avec $eps = 10^{-10}$, on obtient que:

NBITER = 124

NBPART = 56

$$SOL = \begin{pmatrix} 4.999999999998181 \\ 0 \\ 0 \end{pmatrix}.$$

Le temps CPU est alors de 3.32 secondes.

Test 5

$$\min \; -x_1^2 - x_2^2 - x_3^2$$

$$\text{s.c.} \begin{cases} x_1^2 + x_2^2 + x_3^2 - 4 \leq 0 \\ 0.25\, x_1 + x_2 + x_3 - 0.5 \leq 0 \\ -x_1 - x_2 - x_3 + 1 \leq 0 \\ -x_1^2 - x_2^2 - x_3^2 + 1 \leq 0 \\ 0 \leq x_1 \leq 3 \\ 0 \leq x_2 \leq 3 \\ 0 \leq x_3 \leq 3 \end{cases}$$

La solution de ce problème est $(2,0,0)^T$. Le tableau des résultats est:

	eps	nbiter	nbpart	sol
A	0,001	44	35	1.99987 0 0
B	0,001	40	72	1.99987 0 0
C	0,001	163	110	1.99984 0 0
D	0,001	76	146	1.99999 0 0
E	0,001	40	72	1.99987 0 0

L'algorithme A demande le moins d'éléments dans la partition. Si on teste cet algorithme avec $eps = 10^{-11}$, on obtient que:

NBITER = 125

NBPART = 82

$$SOL = \begin{pmatrix} 1.99999999999995 \\ 0 \\ 0 \end{pmatrix}.$$

Le temps CPU est alors de 5,14 secondes.

3. Etude détaillée d'un problème.

Les quelques exemples qui viennent d'être présentés mènent à la conclusion que l'algorithme A est le meilleur, c'est-à-dire qu'à une étape r, on partitionnera en deux n-rectangles les éléments atteignant β_{r-1}. Un listing de ce programme est donné en annexe.

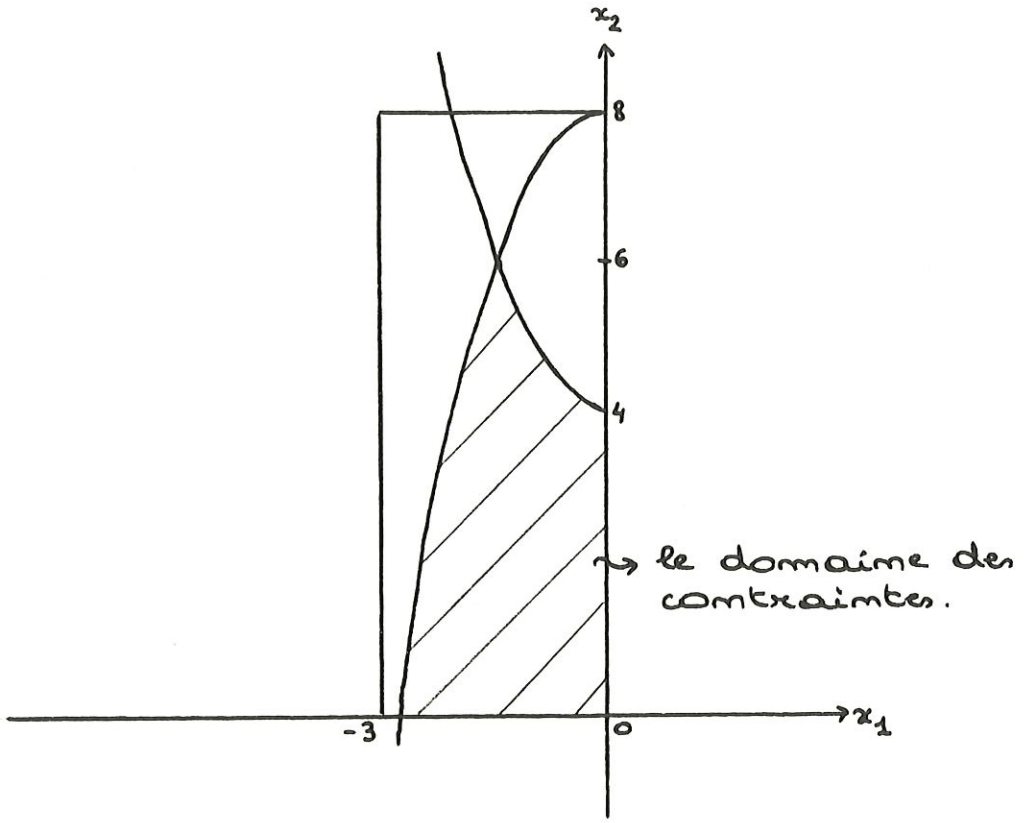
Néanmoins, cet algorithme reste assez lent. Dans le but d'en détecter la cause, examinons plus attentivement la résolution du problème suivant par l'algorithme A :

$$\min -x_1^2 - x_2^2$$

$$\text{s.c.} \begin{cases} x_1^2 + x_2 - 8 \leq 0 \\ -x_1^2 + x_2 - 4 \leq 0 \\ -3 \leq x_1 \leq 0 \\ 0 \leq x_2 \leq 8 \end{cases}$$

La solution exacte de problème est $(-\sqrt{2}, 6)^T$. Pour $EPS = 10^{-11}$, l'algorithme trouve la solution $(-1.41421356236 78, 5.99999999995 34)^T$ en 6.88 secondes.

Graphiquement, on a la situation suivante :



Examinons plus attentivement la résolution de ce problème.

Au premier pas, on a deux éléments dans la partition et le point $(0,4)^T$ atteint la valeur minimale de $d(M)$. Cette valeur minimale reste inchangée jusqu'au pas 22. On n'améliore pas la solution pendant ces itérations. Pourquoi ?

Au pas 2, trois n-rectangles forment la partition. Il s'agit de n-rectangles :

1. de borne inférieure $(-3,0)^T$ et de borne supérieure $(0,4)^T$,
2. de borne inférieure $(-3,4)^T$ et de borne

supérieure $(0,6)^T$,

3. de borne inférieure $(-3,6)^T$ et de borne supérieure $(0,8)^T$.

En fait, ce dernier m-rectangle ne comprend qu'un seul point admissible : $(-\sqrt{2},6)^T$. Cet élément comprenant le sommet atteignant β_2, on le partitionne en deux m-rectangles au pas 3. Les deux nouveaux éléments sont :

. le m-rectangle de borne inférieure $(-3,6)^T$ et de borne supérieure $(-1.5,8)^T$ (Cet élément étant non admissible),

. le m-rectangle de borne inférieure $(-1.5,6)^T$ et de borne supérieure $(0,8)^T$ (Cet élément ne contenant qu'un seul point admissible).

Du pas 4 au pas 21, l'algorithme partitionnera ces deux derniers m-rectangles. Il en éliminera des parties grâce à la règle (DR).

Au pas 22, l'algorithme partitionne le m-rectangle de borne inférieure $(-3,4)^T$ et de borne supérieure $(0,6)^T$, car il contient à ce moment-là le sommet atteignant β_{21}. Le point $(-15,4)^T$ devient le point courant.

Vingt itérations ont donc été consacrées au partitionnement d'un élément ne pouvant accélérer la convergence de l'algorithme, la règle (DR) ne pouvant détecter suffisamment tôt les éléments non admissibles. La lenteur de l'algorithme vient donc de la non-efficacité de la règle (DR).

Chapitre 4 :

Application à un problème de découpe de diamant

Introduction.

Le problème abordé dans ce dernier chapitre est celui de la découpe d'une pierre brute pour en faire un diamant. La façon de procéder dépend en fait de la forme du diamant désiré, mais aussi des propriétés physiques et géométriques de la pierre brute.

Une première approche de ce problème a été présentée [Réf 6]. Elle consiste à trouver, dans une pierre brute, convexe ou non, le plus gros diamant d'une forme spécifique donnée et d'une orientation fixée.

Dans ce chapitre, on se limitera au cas d'une pierre brute convexe et au cas où l'orientation n'est pas fixée. Des résultats numériques seront présentés après avoir modélisé le problème.

1. Description du problème.

Le but est évidemment de maximiser la valeur du diamant taillé. Le problème est donc :
connaissant la forme exacte d'une pierre brute, comment y tailler un diamant d'une forme

déterminée pour que la valeur de celui-ci soit la plus grande possible?

Les hypothèses sont les suivantes :

- la pierre brute est convexe et ne comporte pas d'impureté
- la valeur du diamant est basée sur son poids, c'est-à-dire son volume.

D'autre part, le problème résolu ici sera à deux dimensions.

2. Formulation mathématique.

2.1. Introduction des notations.

On travaille donc dans le plan $\mathbb{R}^2 = x\,O\,y$. On se donne un polygône convexe P_0, c'est-à-dire la pierre brute. On note s le nombre de ses sommets qui est aussi son nombre de faces.

D'autre part, on désigne par P_R la pierre de référence qui est un polygône convexe de forme donnée dans le plan. On désigne par (x_j^R, y_j^R) les coordonnées cartésiennes de ses sommets, $j = 1 \ldots s$, numérotés dans le sens des aiguilles d'une montre, avec s le nombre de ses sommets.

Le problème est : comment inscrire dans P_0 un polygône convexe P_T le plus grand possible et de même forme que le diamant de référence P_R ?

On a que P a la même forme que P_R si et seulement si on peut trouver $d \geq 0$, $\theta \in [0, 2\pi[$, $p \in \mathbb{R}$ et $q \in \mathbb{R}$ tels que

$\forall j = 1, 2, \ldots s$:

$$(x_j, y_j, 1) = (x_j^R, y_j^R, 1) \begin{pmatrix} d\cos\theta & d\sin\theta & 0 \\ -d\sin\theta & d\cos\theta & 0 \\ p & q & 1 \end{pmatrix}$$

avec . P polygône convexe de sommets (x_j, y_j) où $j = 1, \ldots, s$ qui sont numérotés dans le même sens que P_R.

. d facteur de dilatation

. θ l'angle de rotation

. p la translation suivant l'axe des x

. q la translation suivant l'axe des y.

2.2. Formulation de la fonction objectif.

On note donc par P_T le diamant à tailler. Notons S_T sa surface et désignons par S_0 la surface de la pierre brute P_0.

Le problème est donc de trouver s coor-

données (x_j, y_j) du diamant P_T tel que sa surface résiduelle $(S_0 - S_T)$ soit minimale.

Comme S_0 est donné, maximiser S_T revient équivalent à minimiser $(S_0 - S_T)$.

D'autre part, la surface S_T est donnée par la formule :

$$S_T = \frac{1}{2} \sum_{j=1}^{s} (x_j y_{j+1} - x_{j+1} y_j) \qquad (1)$$

avec la convention suivante : $(x_{s+1}, y_{s+1}) = (x_1, y_1)$.
Grâce à la numérotation des sommets, la valeur de S_T est positive.

2.3. Formulation des contraintes.

Deux contraintes sont à formuler :

- le fait que la pierre taillée doit être inscrite dans la pierre brute P_0
- le fait que le diamant à tailler doit avoir la même forme que P_R.

Examinons-les :

1. P_T inscrit dans P_0.

Désignons par $a_i x + b_i y + c_i = 0 \quad i = 1, \ldots, r$, les équations des r droites déterminant la frontière de P_0.

Si on suppose que P_0 est déterminé par les

r inéquations suivantes :

$$a_i x + b_i y + c_i \leq 0 \qquad i = 1, \ldots, r \qquad (2).$$

Pour que P_T soit inscrit dans P_0, il suffit que les s coordonnées vérifient ces r inégalités puisque P_0 est convexe.

2. P_T a la même forme que P_R.

Il suffit pour cela que :

$\forall j = 1, \ldots, s$:

$$(x_j, y_j, 1) = (x_j^R, y_j^R, 1) \begin{pmatrix} d\cos\theta & d\sin\theta & 0 \\ -d\sin\theta & d\cos\theta & 0 \\ p & q & 1 \end{pmatrix} \qquad (3).$$

2.4. Expression du problème.

On porte les valeurs des s coordonnées données par (3) dans l'expression (1) et on obtient :

$$\max_{d,\theta,p,q} \frac{1}{2} \sum_{j=1}^{\rightarrow} (x_j^R y_{j+1}^R - x_{j+1}^R y_j^R) d^2.$$

D'autre part, on a que :

$$S_R = \frac{1}{2} \sum_{j=1}^{\rightarrow} (x_j^R y_{j+1}^R - x_{j+1}^R y_j^R)$$

et donc le problème devient

$$\max_{d,\theta,p,q} d^2.$$

Si on porte les mêmes expressions données par l'expression (3) dans l'expression (2), on

obtient que :

$$C_{ij}(d,\theta,p,q) \equiv$$
$$(a_i x_j^R + b_i y_j^R)\, d\cos\theta + (b_i x_j^R + a_i y_j^R)\, d\sin\theta +$$
$$+ a_i p + b_i q + c_i \leq 0.$$

Le problème à résoudre est donc le suivant :

$$\max_{d,\theta,p,q} \; d^2$$
$$\text{s.c.} \begin{cases} C_{ij}(d,\theta,p,q) \leq 0 & i = 1,\dots,r \quad j = 1,\dots,s \\ 0 \leq \theta \leq 2\pi. \end{cases}$$

On a donc un problème non linéaire à quatre variables et r.s contraintes non linéaires avec une contrainte de borne.

Afin de mettre les contraintes sous forme linéaire, on effectue le changement de variables suivant :

$$u = d\cos\theta$$

$$v = d\sin\theta.$$

En remplaçant dans (3), on obtient l'expression des coordonnées (x_j, y_j) en fonction de nouvelle variable :

$\forall j = 1,2,\dots s$:

$$(x_j, y_j, 1) = (x_j^R, y_j^R, 1) \begin{pmatrix} u & v & 0 \\ -v & u & 0 \\ p & q & 1 \end{pmatrix}.$$

Le problème à résoudre devient donc :

$$\begin{cases} \min\limits_{u,v,p,q} -u^2 - v^2 \\ \text{s.c.} \quad Q_{ij}(u,v,p,q) \leq 0 \end{cases} \qquad \begin{matrix} i = 1, \dots, r \\ j = 1, \dots, s \end{matrix}$$

avec

$$Q_{ij}(u,v,p,q) \equiv$$
$$(a_i x_j^R + b_i y_j^R)\, u + (b_i x_j^R + a_i y_j^R)\, v + a_i p + b_i q + c_i .$$

On a donc formalisé le problème de découpe de diamants comme un problème à 4 variables, concave, sous r.s contraintes linéaires.

3. Description de la méthode utilisée.

On a donc formalisé le problème de la manière suivante :

$$\begin{cases} \min f(x) \\ \text{s.c.} \quad Ax - b \leq 0 \end{cases}$$

avec

- f, fonction concave de $\mathbb{R}^4$ à valeur réelle
- A matrice réelle de r.s lignes et 4 colonnes
- b vecteur colonne réel de r.s lignes.

L'algorithme décrit au deuxième chapitre pourra être utilisé si on arrive à trouver un m-rectangle de départ, c'est-à-dire des

bornes sur les quatre variables à trouver.

Néanmoins, l'algorithme utilisé pour résoudre ce problème comporte une différence par rapport à l'algorithme étudié au deuxième chapitre. En effet, la méthode utilisée pour l'implémentation en FORTRAN 77 sur l'ordinateur VAX exploitera la forme particulière du problème: les contraintes ne sont plus séparable quadratiques, mais bien linéaires. La règle (DR) avait été introduite et permettait de détecter certains éléments de la partition n'ayant aucune intersection avec le domaine des contraintes. Dans le cas particulier des contraintes linéaires, toutes les partitions non admissibles pourront être détectées. On fait, dans ce but, appel à la sous-routine LA02AD de la librairie Harwell, permettant de trouver, si possible, un point admissible $\underline{x} = (x_1, \dots x_m)^T$ pour l'ensemble déterminé par les r contraintes linéaires suivantes:

$$\begin{cases} \underline{l} \leq \underline{x} \leq \underline{u} \\ C^T x \geq \underline{d} \end{cases}$$

avec

. $\underline{l}$ et $\underline{u}$ vecteurs colonnes de m lignes

. $\underline{d}$ vecteur de $\mathbb{R}^{n-2m}$

. C matrice à m lignes et $n-2m$ colonnes.

Si aucun élément admissible n'a été trouvé, elle renvoie ce résultat.

Dans les pages qui suivent, on présente quelques résultats.

4. Résultats numériques.

Pour chacun des problèmes traités, on présente un tableau comparant, suivant les valeurs de EPS, les variables suivantes :

NBITER = le nombre d'itérations,

NBPART = le nombre d'éléments dans la partition,

NBELIM = le nombre de n-rectangles éliminés par la sous-routine LA02AD et

TPS = le temps CPU (en secondes).

Test 1

Le diamant de référence P_R est déterminé par les sommets :

1. $(2,2)^T$
2. $(0,-2)^T$
3. $(-2,2)^T$.

On veut tailler un diamant dans une pierre brute déterminée par les quatres inégalités :

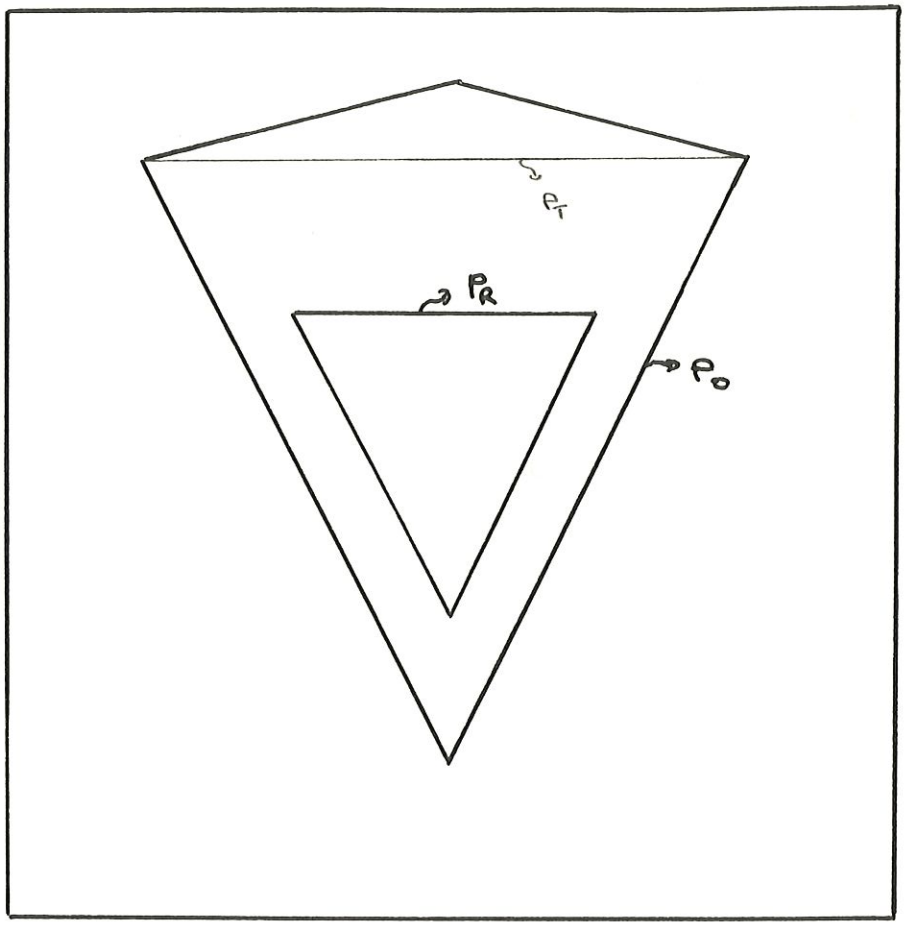
A. $-\frac{1}{4}x + y - 5 \leq 0$

B. $\frac{1}{4}x + y - 5 \leq 0$

C. $2x - y - 4 \leq 0$

D. $-2x - y - 4 \leq 0$.

Le problème a donc 12 contraintes. La représentation graphique du problème nous amène à choisir comme borne inférieure du n-rectangle de départ le point $(-2,-2,-1,-1)^T$ et comme borne supérieure le point $(2,2,1,1)^T$.



Le facteur de dilatation est 2. Le tableau des résultats est :

EPS	NBITER	NB PART	NBELIM	TPS
0.001	25	348	318	21.80
0.000001	45	740	738	42.31

Test 2

Le diamant de référence est déterminé par les sommets :

1. $(1, 1)^T$
2. $(-1, 1)^T$
3. $(1, -1)^T$
4. $(-1, -1)^T$.

On veut tailler un diamant dans une pierre brute déterminée par les cinq inégalités :

A. $-\frac{7}{4}x + y - 7 \leq 0$

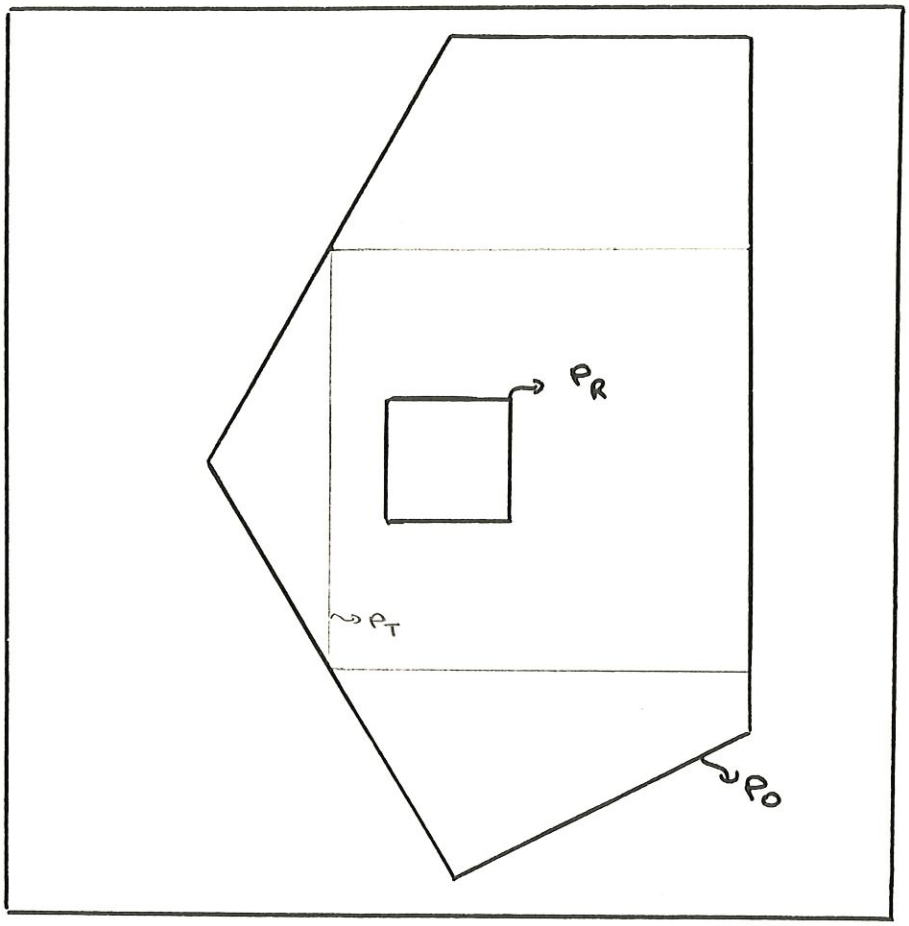
B. $y - 7 \leq 0$

C. $x - 5 \leq 0$

D. $\frac{1}{2}x - y - 7 \leq 0$

E. $-\frac{7}{4}x - y - 7 \leq 0$.

Le problème possède donc 20 contraintes. La pierre de référence étant un carré, on peut supposer que les valeurs de $\sin\theta$ et $\cos\theta$ sont comprises entre 0 et 1. La représentation graphique du problème nous amène à choisir comme borne inférieure du m-rectangle de départ le point $(0, 0, 0, -2)^T$ et comme borne supérieure le point $(4, 4, 2, 2)^T$.



Le facteur de dilatation est 3.5. Le tableau de résultats est :

EPS	NBITER	NBPART	NBELIM	TPS
0.001	126	880	1198	82.64
0.000001	148	932	1229	99.36

Test 3

Le diamant de référence est déterminé par les sommets:

1. $(2, 0)^T$
2. $(0.5, 2.5)^T$
3. $(1.5, 3.5)^T$
4. $(2.5, 3.5)^T$
5. $(3.5, 2.5)^T$.

On veut tailler un diamant dans une pierre brute déterminée par les sept inégalités:

A. $-y - 7 \leq 0$

B. $-2x - y - 11 \leq 0$

C. $-x - 5 \leq 0$

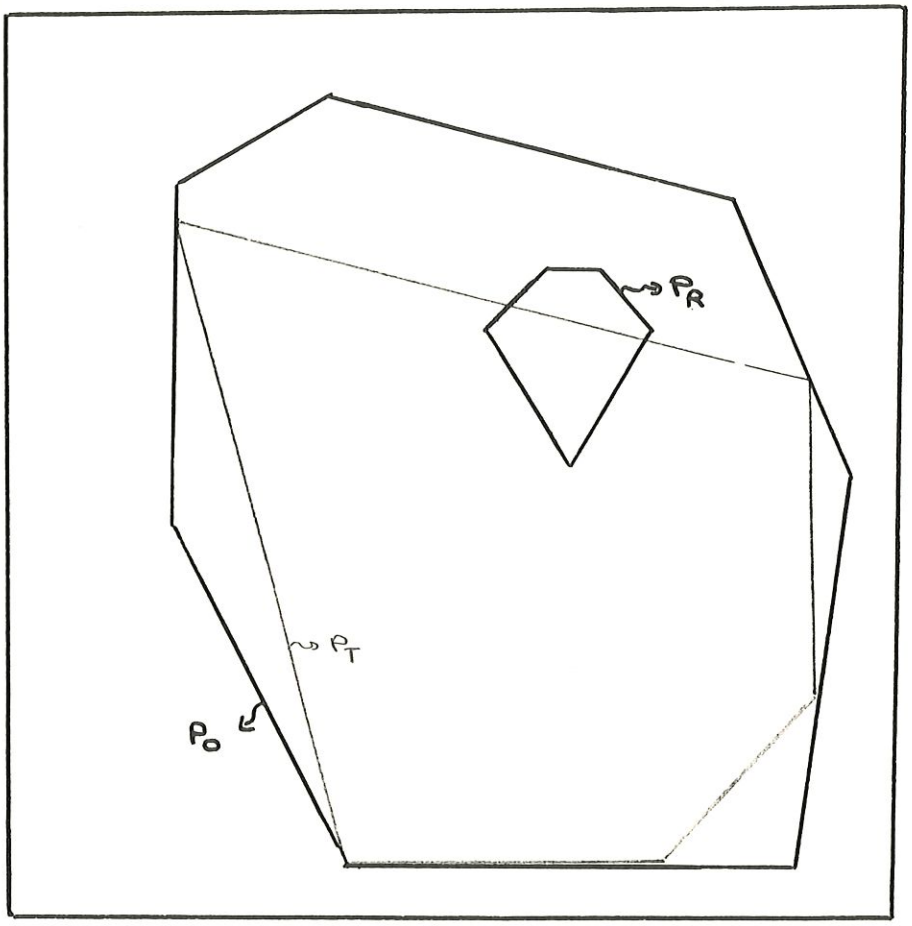
D. $-3x + 5y - 40 \leq 0$

E. $0.25x + y - 6 \leq 0$

F. $7x + 3y - 49 \leq 0$

G. $7x - y - 49 \leq 0$

Le problème a donc 35 contraintes. La représentation graphique du problème nous amène à choisir comme borne inférieure du problème le point $(-5, -5, -2, -6)^T$ et comme borne supérieure le point $(5, 5, 1, 1)^T$.



Le facteur de dilatation est 3.99. Le tableau des résultats est :

EPS	NBITER	NBPART	NBRLIM	TPS
0.001	98	512	459	63.12
0.000001	126	703	686	85.01

Test 4

Le diamant de référence P_R est déterminé par les sommets:

1. $(-1, 0.3)^T$
2. $(1, 0.3)^T$
3. $(1.5, 0)^T$
4. $(1.5, -0.1)^T$
5. $(0, -1.5)^T$
6. $(-1.5, -0.1)^T$
7. $(-1.5, 0)^T$.

On désire tailler un diamant dans une pierre brute déterminée par les six inégalités:

A. $y - 1 \leq 0$

B. $2x + y - 4 \leq 0$

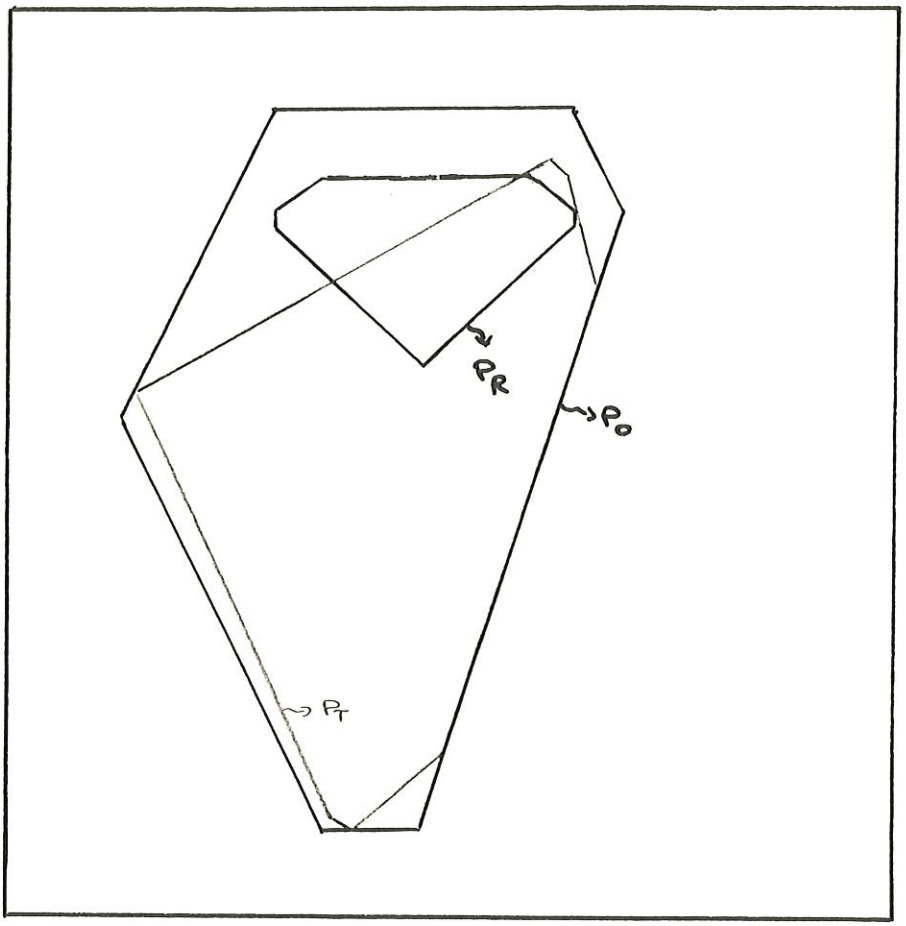
C. $3x - y - 6 \leq 0$

D. $-y - 6 \leq 0$

E. $-2x - y - 8 \leq 0$

F. $-2x + y - 4 \leq 0$.

Le problème a donc 42 contraintes. La représentation graphique du problème nous amène à choisir comme borne inférieure du m-rectangle de départ le point $(-4, -4, -\frac{1}{2}, -4)^T$ et comme borne supérieure le point $(4, 4, \frac{1}{2}, 0)^T$.



Le facteur de dilatation est 2.26. Le tableau des résultats est:

EPS	NBITER	NBPART	NBELIM	TPS
0.001	84	708	692	69.75
0.000001	113	952	946	92.19

Conclusion

Il existe beaucoup de méthodes de type "Branch and Bound" basées sur le partitionnement d'un ensemble, dans le but de déterminer un intervalle de plus en plus petit contenant la valeur optimale du problème traité. Elles diffèrent par la forme de l'ensemble de départ à partitionner, par le type de partitionnement utilisé et par le choix des bornes $\alpha(M)$ et $\beta(M)$ associées à chaque élément M de la partition vérifiant la relation suivante:
$\beta(M) \leq \min_{} f(M \cap D) \leq \alpha(M)$, D désignant le domaine admissible.

Pour le lecteur intéressé, la recherche doit se concentrer sur un choix de ces trois points adapté à chaque type de problème. De plus, il faut avoir à sa disposition une règle efficace de détection des éléments M de la partition non admissible. La lenteur de l'algorithme implementé dans ce mémoire était dûe au fait que la règle proposée n'est pas performante.

Annexe

```
C     ****************************************************************
C     ****************************************************************
C     ****************************************************************
                  PROGRAM ALGO_BISSECT_MINBETA
C     ****************************************************************
C     ****************************************************************
C
C
C     But du programme :
C     ------------------
C     +
C     +  Ce programme cherche min F(x)
C     +                        s.c. Gi(x)<=0  i=1,...,m
C     +                             INF <= x <= SUP
C     +
C     +  avec : - F fonction concave
C     +         - Gi ( i=1,...,m ) fonctions separables quadrati-
C     +           ques
C     +         - INF et SUP bornes d'un n-rectangle.
C     +
C     +
C     +  L'algorithme est du type Branch-and-Bound. Il encadre la
C     +  valeur de la solution dans un intervalle de plus en plus
C     +  petit. On construit les bornes de cet intervalle en par-
C     +  titionnant le N-rectangle de borne inferieure INF et de
C     +  borne superieure SUP. Le partitionement est ici la bis-
C     +  section des elements M atteignant la valeur minimale des
C     +  Beta(M).
C
C
C     Table des variables :
C     ---------------------
C
C     1. Les variables entieres :
C
C      NMAX : borne superieure sur la dimension du probleme
C             traite. Elle vaut 20.
C      MMAX : borne superieure sur le nombre de contrainte
C             Gi. Elle vaut 120.
C      N : dimension du probleme (<= NMAX)
C      M : nombre de fonctions Gi (<= MMAX)
C      MAXPART : borne sur le nombre d'elements admis dans la
C                partition. Elle vaut 10000.
C      FINPART : nombre de vecteurs stockes + 1 (<=MAXPART)
C      U,J,W,R : indices compteurs de boucle
C      FINBE : entier tq a la Kieme iteration, FINBE indique
C              le nombre de vecteurs atteignant le minimum
C              des beta(M), cad MINBETA (<= MAXPART)
C      COMPT : indice compteur du nombre d'iteration de l'al-
C              gorithme
C      US : entier representant l'unite de sortie
C      IDS : entier de gestion d'erreur
C      NBCALLF : entier comptant le nombre d'appels a la
C                fonction objective F
C      IT, ISECOND, ISECONDIFF : variables de calcul du temps
C                                CPU
C      ATTEINTMINBETA : vecteur contenant les numeros des
C                       colonnes ou sont stockes dans TABINF
C                       et TABSUP les elements de la par-
C                       tition atteignant MINBETA
```

```
C
C      2. Les variables Double Precision :
C
C       TABINF : tableau tq la Jeme colonne contient la borne
C                inferieure du Jeme element de la partition
C       TABSUP : tableau tq la Jeme colonne contient la borne
C                superieure du Jeme element de la partition
C       BETA : vecteur tq son Jeme element contient la valeur
C              du beta relatif au Jeme element de la partition
C       ALPHA : vecteur tq son Jeme element contient la valeur
C               du alpha relatif au Jeme element de la partition
C       SOL : vecteur contenant le point courant
C       MINALPHA : reel contenant la valeur minimale des Alpha
C                  a chaque iteration
C       MINBETA : reel contenant la valeur minimale des Beta
C                 a chaque iteration
C       EPS : ecart demande entre MINALPHA et MINBETA pour
C             stopper l'algorithme
C       MINF : vecteur de stockage de la borne inferieure de
C              l'element a partitionner
C       MSUP : vecteur de stockage de la borne superieure de
C              l'element a partitionner
C       INFDROITE : vecteur contenant la borne inferieure de
C                   l'element de droite de l'element a parti-
C                   tionner
C       SUPDROITE : vecteur contenant la borne superieure de
C                   l'element de droite de l'element a parti-
C                   tionner
C       INFGAUCHE : vecteur contenant la borne inferieure de
C                   l'element de gauche de l'element a parti-
C                   tionner
C       SUPGAUCHE : vecteur contenant la borne superieure de
C                   l'element de gauche de l'element a parti-
C                   tionner
C       ALPART : reel de stockage temporaire de la valeur de
C                ALPHA(M) relatif a un element de la partition
C       BEPART : reel de stockage temporaire de la valeur de
C                BETA(M) relatif a un element de la partition
C       POINT : vecteur de stockage du point atteignant ALPART
C       CONTR : tableau contenant les reels definissant les
C               fonctions-contraintes Gi, separables quadra-
C               tiques :
C               CONTR(I,K,1) = Pik
C               CONTR(I,K,2) = Qik
C               CONTR(I,K,3) = Rik
C       BI : vecteur contenant la borne inferieure du N-rectangle
C            de depart
C       BS : vecteur contenant la borne superieure du N-rectangle
C            de depart
C       TABPOINT : tableau tel que la Jeme colonne contient les
C                  coordonnees du sommet atteignant le ALPHA(M)
C                  relatif au Jeme element de la partition
C       RES : nombre contenant la valeur de la derniere evaluation
C             de l'expression Ai calcule pour l'evaluation de la
C             regle DR
C
C      3. Les variables Boolennes :
C
C       ACCES : tableau booleen tel que
C                  ACCES(J) = .false. ssi le Jeme element de la
```

```
C                              partition est elimine par DR ou R1
C       DRG : variable boolenne tq DRG = .true. ssi le n-rectangle
C             de gauche obtenu apres partition d'un elememt est
C             elimine par DR
C       DRD : variable boolenne tq DRD = .true. ssi le n-rectangle
C             de droite obtenu apres partition d'un elememt est
C             elimine par DR
C
C      4. Les variables de type caractere :
C
C       FCHNOM : string contenant le nom du fichier de donnees
C       FCHSOR : string contenant le nom du fichier de sortie
C
C
C
C      Declaration des variables :
C      ---------------------------
C
       INTEGER NMAX,MMAX,N,M,MAXPART,FINPART,U,FINBE,J,W,COMPT,
      +        R,US,IOS,NBCALLF,IT,ISECOND,ISECONDIFF
       PARAMETER(NMAX=20)
       PARAMETER(MMAX=120)
       PARAMETER(MAXPART=10000)
       INTEGER ATTEINTMINBETA(MAXPART)
       DOUBLE PRECISION TABINF(NMAX,MAXPART),TABSUP(NMAX,MAXPART),
      +                 BETA(MAXPART),ALPHA(MAXPART),SOL(NMAX),
      +                 MINALPHA,MINBETA,EPS,MINF(NMAX),MSUP(NMAX),
      +                 INFDROITE(NMAX),SUPDROITE(NMAX),
      +                 INFGAUCHE(NMAX),SUPGAUCHE(NMAX),
      +                 ALPART,BEPART,POINT(NMAX),
      +                 CONTR(MMAX,NMAX,3),BI(NMAX),BS(NMAX),
      +                 TABPOINT(NMAX,MAXPART),RES,SOM
       LOGICAL ACCES(MAXPART),DRG,DRD,DR
       CHARACTER*10 FCHNOM,FCHSOR
C
C
C
C*************************************************************************
C*************************************************************************
C***************** Corps du programme ************************************
C*************************************************************************
C*************************************************************************
C
C
C      On fait appel a deux sous-routines de la librairie UTIL
C      pour le calcul du temps CPU :
C
       CALL INIT
       CALL ISECOND (IT)
C
C      On ouvre le fichier des donnees et le fichier de sortie :
C      ---------------------------------------------------------
C
        WRITE(*,5)
5       FORMAT(//,1X,'Quel est le nom du fichier de donnee?',/,
      +        1X,'(sans extension)')
        READ(*,10)FCHNOM
10      FORMAT(A10)
C
C
```

```
      WRITE(*,15)
15    FORMAT(//,1X,'Quel est le nom du fichier de sortie  ?',/,
     +       1X,'(sans extension)')
      READ(*,20)FCHSOR
20    FORMAT(A10)
C
C
      OPEN(UNIT=US,ERR=25,FILE=FCHSOR,STATUS='NEW',IOSTAT=IOS)
25    IF (IOS.NE.0) THEN
         WRITE(*,28)
28       FORMAT(1X,'PROBLEME A L''OUVERTURE DU FICHIER DE ',/,
     +          1X,'DONNEES')
         CLOSE(US)
      ENDIF
C
      WRITE(US,30)
30    FORMAT(50('*'))
      WRITE(US,30)
C
      WRITE(US,35)
35    FORMAT(/,1X,'Le probleme a ete resolu en bissequant',/,
     +       1X,'a chaque etape tous les elements atteignant',/,
     +       1X,'la valeur minimale des Beta(M)',1X,/)
C
      WRITE(US,30)
C
      WRITE(US,38)FCHNOM
38    FORMAT(////,' Le probleme traite vient du fichier : ',A10)
C
C
C     Initialisations :
C     ---------------
C
C     Pas r0 : on enregistre les donnees necessaires au probleme,
C     on initialise les variables necessaires et on inscrit les
C     donnees importantes dans le fichier de sortie (MINALPHA,
C     MINBETA et SOL ) :
C
C
      CALL DONNEES(FCHNOM,N,M,CONTR,BI,BS,EPS,NMAX,MMAX,US)
      FINPART=2
      DO 40,W=1,N
         TABINF(W,1)=BI(W)
         TABSUP(W,1)=BS(W)
40    CONTINUE
      ACCES(1)=.TRUE.
      WRITE(US,45)
45    FORMAT(///,' +++++ PAS 0 +++++ ')
      WRITE(US,50)
50    FORMAT(//)
C
      CALL CALCUL(N,M,BI,BS,ALPART,BEPART,POINT,CONTR,NMAX,MMAX,
     +            US)
      NBCALLF=2**N
      BETA(1)=BEPART
      ALPHA(1)=ALPART
      DO 55,W=1,N
         SOL(w)=POINT(W)
55    CONTINUE
      MINALPHA=ALPART
```

```
      MINBETA=BEPART
      ATTEINTMINBETA(1)=1
      FINBE=1
      COMPT=0
      WRITE(US,60)MINALPHA
60    FORMAT(1X,'MINALPHA = ',1X,1PG24.16E2)
      WRITE(US,65)MINBETA
65    FORMAT(1X,'MINBETA = ',1X,1PG24.16E2)
      WRITE(US,*)'Le point atteignant MINALPHA'
      DO 80,W=1,N
         WRITE(US,70)SOL(W)
70       FORMAT(1PG24.16E2)
80    CONTINUE
C
C
C
C     PAS r :
C     -----
C
C
C     On boucle tant que l'on n'a pas la precision voulue ou
C     que l'on ne depasse pas la capacite de stockage :
C
100   IF ( ((abs(MINALPHA-MINBETA)).GT.EPS).and.
     +     (FINPART+(2*FINBE).LE.MAXPART)) THEN
C
C
C
         WRITE(US,120)
120      FORMAT(//)
         WRITE(US,130)
130      FORMAT(1X,3(50('*'),1X,/))
         COMPT=COMPT+1
         WRITE(US,150)COMPT
150      FORMAT(//,' +++++ PAS',I4,1X,' +++++',1X,//)
C        (*pas r1*)
         CALL ENLEVE(MAXPART,FINPART,ACCES,BETA,MINALPHA)
C
C
C        Bissection des elements atteignant MINBETA ,on les
C        stocke si necessaires et on ajuste les vecteurs
C        ALPHA et BETA, ainsi que le tableau TABPOINT :
C
         DO 600, U=1,FINBE
C
            WRITE(US,175)
175         FORMAT(//)
            J=ATTEINTMINBETA(U)
            IF (ACCES(J)) THEN
               DO 200, W=1,N
                  MINF(W)=TABINF(W,J)
                  MSUP(W)=TABSUP(W,J)
200            CONTINUE
C              (*pas r2 : on bissecte *)
               CALL BISSEC(J,MAXPART,N,FINPART,TABSUP,TABINF,NMAX,
     +                     compt)
C
C              (*pas r3. On evalue la regle DR pour les 2
C                nouveaux elements de la partition : *)
               DO 300, W=1,N
```

```
                  INFGAUCHE(W)=MINF(W)
                  SUPGAUCHE(W)=TABSUP(W,J)
                  INFDROITE(W)=TABINF(W,FINPART)
                  SUPDROITE(W)=MSUP(W)
300               CONTINUE
                  WRITE(US,310)U
310               FORMAT(1X,'Partitionnement',1X,I2)
                  DRG=DR(N,M,INFGAUCHE,SUPGAUCHE,CONTR,NMAX,MMAX,
     +                  RES,US)
                  WRITE(US,320)RES
320               FORMAT(1X,'DRG=',1X,1PG24.16E2)
                  DRD=DR(N,M,INFDROITE,SUPDROITE,CONTR,NMAX,MMAX,
     +                  RES,US)
                  WRITE(US,330)RES
330               FORMAT(1X,'DRD=',1X,1PG24.16E2)
C
C
C                 On traite l'element de GAUCHE, stocke en Jeme
C                 colonne.
                  IF (DRG) THEN
C                    (*Pas r4 pour l'element de gauche*)
C                    L'element de gauche verifie la regle DR, on
C                    peut le rendre inaccessible.
                     ACCES(J)=.FALSE.
                  ELSE
C                    On evalue alpha(M) et beta(M) Bpour cet element.
                     CALL CALCUL(N,M,INFGAUCHE,SUPGAUCHE,ALPART,
     +                           BEPART,POINT,CONTR,NMAX,MMAX,US)
                     NBCALLF=NBCALLF+2**N
                     BETA(J)=BEPART
                     ALPHA(J)=ALPART
                     DO 350,W=1,N
                        TABPOINT(W,J)=POINT(W)
350                  CONTINUE
                  ENDIF
C
C
C                 On examine le n-rectangle de DROITE, stocke en
C                 FINPARTeme colonne. Si il verifie DR, on ne bouge
C                 pas FINPART, ie on ne le gardera plus en memoire :
C
                  IF (.NOT.DRD) THEN
C                    (*Pas r4 pour l'element de droite*)
                     ACCES(FINPART)=.TRUE.
C                    On evalue alpha(M) et beta(M) pour cet element.
                     CALL CALCUL(N,M,INFDROITE,SUPDROITE,ALPART,
     +                           BEPART,POINT,CONTR,NMAX,MMAX,US)
                     NBCALLF=NBCALLF+2**N
                     BETA(FINPART)=BEPART
                     ALPHA(FINPART)=ALPART
                     DO 450,W=1,N
                        TABPOINT(W,FINPART)=POINT(W)
450                  CONTINUE
                     FINPART=FINPART+1
                  ENDIF
               ENDIF
C
600         CONTINUE
C
C
```

```
C         (*Pas r5 : On ajuste MINALPHA, MINBETA, ATTEINTMINBETA ,
C           FINBE et SOL (le point courant) : *)
C
          MINALPHA=100000.
          MINBETA=100000.
          FINBE=0
          DO 650,W=1,N
             SOL(W)=100000.
650       CONTINUE
C
          DO 675,R=1,FINPART-1
              IF (ACCES(R)) THEN
C
                    IF (BETA(R).LT.MINBETA) THEN
                       MINBETA=BETA(R)
                       ATTEINTMINBETA(1)=R
                       FINBE=1
                    ELSE
                       IF (BETA(R).EQ.MINBETA) THEN
                          FINBE=FINBE+1
                          ATTEINTMINBETA(FINBE)=R
                       ENDIF
                    ENDIF
C
                    IF (ALPHA(R).LT.MINALPHA) THEN
                       MINALPHA=ALPHA(R)
                       DO 660, W=1,N
                          SOL(W)=TABPOINT(W,R)
660                    CONTINUE
                    ENDIF
C
              ENDIF
675       CONTINUE
C
C
C
C         On affiche a chaque etape dans le fichier de sortie
C         les variables "interessantes" ,ie FINPART, MINALPHA,
C         MINBETA et SOL :
          WRITE(US,680)
680       FORMAT(1X,//,50('-'))
          WRITE(US,690)
690       FORMAT(1X,//,1X,'Pour cette etape : ')
          WRITE(US,700)
700       FORMAT(1X,'----------------')
          WRITE(US,710)FINPART
710       FORMAT(1X,'Le nombre de vecteurs stockes :',1X,I6)
          WRITE(US,720)MINALPHA
720       FORMAT(1X,'La valeur de MINALPHA est : ',1X,1PG24.16E2)
          WRITE(US,730)MINBETA
730       FORMAT(1X,'La valeur de MINBETA est :',1X,1PG24.16E2)
          WRITE(US,*)'Le point atteignant MINALPHA : '
          DO 740,W=1,N
             WRITE(US,70)SOL(W)
740       CONTINUE
C
C
C
          GOTO 100
       ENDIF
```

```
C
C
C
C     On a la precision voulue. On termine par l'impression des
C     resultats finaux dans le fichier de sortie US (on imprime
C     COMPT, SOL, MINALPHA, MINBETA, FINPART , le temps CPU et
C     NBCALLF) :
C     ------------------------------------------------------
C
C
      WRITE(US,800)
800   FORMAT(1X,//)
      WRITE(US,130)
      WRITE(US,850)
850   FORMAT(///,' LA SOLUTION FINALE EST : ',///)
      WRITE(US,900)COMPT
900   FORMAT(1X,'Le nombre d''iterations',1X,I5)
      WRITE(US,1000)
1000  FORMAT(1X,'La solution trouvee est : ')
      DO 1100,W=1,N
         WRITE(US,70)SOL(W)
1100  CONTINUE
      WRITE(US,1200)MINALPHA
1200  FORMAT(1X,'La valeur a cette solution de F est : ',
     +       1X,1PG24.16E2)
      WRITE(US,1300)MINBETA
1300  FORMAT(1X,'La valeur de MINBETA est :',1X,1PG24.16E2)
      WRITE(US,1400)FINPART
1400  FORMAT(1X,'Le nombre de vecteurs stockes :',1X,I6)
      WRITE(US,1500)NBCALLF
1500  FORMAT(1X,'Le nombre d''appels a la fonction F :' ,1X,I5)
C
      IT = ISECONDIFF(IT)
      WRITE(US,1600)IT
1600  FORMAT(1X,'Le temps CPU : ',1X,I10)
C
      CLOSE(US)
C
      END
C
C
C
C*********************************************************************
C*********************************************************************
C*********************************************************************
C*********************************************************************
C
C
C
      LOGICAL FUNCTION DR(N,M,A,B,CONTR,NMAX,MMAX,RES,US)
C
C
C     Effet :
C     -----
C     Ayant un N-rectangle a sa disposition, cette sous-routine
C     evalue sa valeur pour la regle DR.
C     On renvoie DR=.TRUE. si l'element de la partition
C     considere verifie la regle DR, cad peut etre elimine,
C     car il est non admissible.
C
```

```
C
C       Declaration des arguments :
C       ---------------------------
        INTEGER M,N,NMAX,MMAX,US
        DOUBLE PRECISION A(NMAX),B(NMAX),CONTR(MMAX,NMAX,3),RES
        DOUBLE PRECISION GI
C
C
C       Table des arguments :
C       --------------------
C       M = nombre de contraintes Gi (<= MMAX)
C       N = dimension du probleme (<= NMAX)
C       NMAX = borne superieure sur la dimension du probleme
C              traite
C       MMAX = borne superieure sur le nombre de contraintes
C              Gi
C       US = entier representant l'unite de sortie
C       A = borne inferieure du N-rectangle considere
C       B = borne superieure du N-rectangle considere
C       CONTR = tableau contenant les reels definissant les
C               fonctions-contraintes Gi, separables quadra-
C               tiques :
C                CONTR(I,K,1) = Pik
C                CONTR(I,K,2) = Qik
C                CONTR(I,K,3) = Rik
C       RES = reel de stockage de l'expression
C                max { Gi(A),Gi(B) } - Ai * d(M)
C             avec d(M) le diametre du N-rectangle traite.
C       GI = la fonction permettant d'evaluer la valeur de la
C            Ieme contrainte en un point donne
C
C
C       Declaration des variables locales :
C       -----------------------------------
        DOUBLE PRECISION SOM,DIAM,Ai,VALA,VALB,SOM2,MAX
        INTEGER K,I
C
C
C       Table des variables locales :
C       -----------------------------
C       SOM = carre du diametre du N-rectangle considere
C       DIAM = diametre du N-rectangle considere
C       Ai = variable contenant la valeur de l'expression Ai
C       VALA = valeur prise par la fonction Gi au point A
C       VALB = valeur prise par la fonction Gi au point B
C       SOM2 = variable contenant la valeur de Ai au carre
C       MAX = maximum de VALA et VALB
C       K,I = indices compteur de boucle
C
C
C       Initialisations :
C       ---------------
        DR=.FALSE.
        I=1
C
C
C
C       Corps de la procedure :
C       ----------------------
C
```

```
C
C
C       On calcule le diametre du N-rectangle :
        SOM=0
        DO 100,K=1,N
           SOM= SOM + ( A(K)-B(K) )**2
100     CONTINUE
        DIAM=SQRT(SOM)
C
C
C       On boucle tant que l'on a pas une valeur positive de
C       l'expression ou que toutes les contraintes n'ont pas
C       ete examinees :
200     IF ( (.NOT.DR) .AND. (I.LE.M) ) THEN
C
C          On met dans MAX le maximum de Gi(A) et de Gi(b)
           VALA=GI(A,CONTR,M,N,I,NMAX,MMAX)
           VALB=GI(B,CONTR,M,N,I,NMAX,MMAX)
           IF (VALA.GT.VALB) THEN
              MAX=VALA
           ELSE
              MAX=VALB
           ENDIF
C
C          On calcule l'expression Ai dans la variable Ai
           SOM2=0
           DO 300,K=1,N
              IF (CONTR(I,K,1).EQ.0) THEN
                 SOM2=SOM2 + (CONTR(I,K,2)**2.)
              ELSE
                 IF ( (-CONTR(I,K,2)/CONTR(I,K,1)) .GE.
     +                 ( (A(K)+B(K))/2.) ) THEN
                    SOM2=SOM2+(CONTR(I,K,1)*A(K)+CONTR(I,K,2))**2
                 ELSE
                    SOM2=SOM2+(CONTR(I,K,1)*B(K)+CONTR(I,K,2))**2
                 ENDIF
              ENDIF
300        CONTINUE
           Ai=SQRT(SOM2)
C
C          On evalue l'expression :
           RES=MAX-Ai*DIAM
C
C          Si la valeur de l'expression est positive, on renvoie
C          DR = .TRUE.
           IF (RES.GT.0) THEN
              DR=.TRUE.
           ENDIF
C
           I=I+1
           GOTO 200
        ENDIF
C
C
        END
C
C
C
C*******************************************************************
C*******************************************************************
```

```
C*********************************************************************
C*********************************************************************
C
C
C
      SUBROUTINE CALCUL(N,M,INF,SUP,ALPART,BEPART,POINT,
     +                  CONTR,NMAX,MMAX,US)
C
C
C     Effet :
C     -----
C     Ayant un element M de la partition a sa disposition,
C     cette sous-routine calcule la valeur de alpha(M),
C     de beta(M) et POINT (le point atteignant alpha(M)).
C     Si aucun sommet n'est admissible, on affecte a
C     alpha(M) une valeur plus grande que toute valeur prise
C     par la fonction objective F sur le domaine des con-
C     traites et a POINT, un point non admissible.
C     Elle affiche a l'unite de sortie les resultats
C     BEPART, ALPART et rappelle les bornes inferieure et
C     superieure.
C
C
C     Declaration des arguments :
C     -------------------------
C
      INTEGER N,M,NMAX,MMAX,US
      DOUBLE PRECISION INF(NMAX),SUP(NMAX),ALPART,BEPART,
     +                 POINT(NMAX),CONTR(MMAX,NMAX,3),F,GI
C
C
C     Table des arguments :
C     -------------------
C     N = dimension du probleme (<= NMAX)
C     M = nombre de contraintes Gi (<= MMAX)
C     NMAX = borne superieure sur la dimension du probleme
C            traite
C     MMAX = borne superieure sur le nombre de contrainte
C             Gi
C     US = entier representant l'unite de sortie
C     INF = borne inferieure de l'element M de la partition
C           considere
C     SUP = borne superieure de l'element M de la partition
C          considere
C     ALPART = variable de sortie contenant alpha(M)
C     BEPART = variable de sortie contenant beta(M)
C     POINT = le sommet tq F(point)=alpha(M)
C     CONTR = tableau contenant les reels definissant les
C             fonctions-contraintes Gi, separables quadra-
C             tiques :
C              CONTR(I,K,1) = Pik
C              CONTR(I,K,2) = Qik
C              CONTR(I,K,3) = Rik
C     F = la fonction objective
C     Gi = la fonction permettant d'evaluer la valeur de la
C          Ieme contrainte en un point donne
C
C
C     Declaration des variables locales :
C     ---------------------------------
```

```
C
      INTEGER T,I,K,H,U,MAX
      PARAMETER(MAX=30)
      INTEGER COMPTBIN(MAX)
      DOUBLE PRECISION SOMMET(MAX),VALSOM
      LOGICAL DEDANS,PROBLEM
C
C
C     Table des variables locales :
C     -----------------------------
C     T,I,K,H,U = indice compteur de boucle
C     MAX = borne superieure pour la variable N
C     COMPTBIN = vecteur permettant d'engendrer tous les sommets
C                du N-rectangle considere ayant a sa disposition
C                sa borne inferieure et superieure
C     SOMMET = vecteur de stockage de sommets engendres
C     VALSOM = variable de stockage de la valeur de F au sommet
C              considere
C     DEDANS = .TRUE. quand rien ne permet de dire a ce moment
C                      que le sommet considere n'est pas dans D
C     PROBLEM=.TRUE. quand tous les sommets de la partition
C                    qui ont ete consideres jusque-la
C                    ne sont pas dans D, le domaine
C                    des contraintes
C
C
C     Initialisations :
C     ---------------
C
C     On recopie la borne inferieure et la borne superieure du
C     N-rectangle considere dans le fichier de sortie pour l'in-
C     terpretation des resultats :
      WRITE(US,20)
20    FORMAT(1X,46('-'))
      WRITE(US,30)
30    FORMAT(1X,'AU DEBUT DE CALCUL')
      WRITE(US,35)
35    FORMAT(1X,'Pour le N-rectangle de borne inferieure : ')
      DO 60,I=1,N
         WRITE(US,40)INF(I)
40       FORMAT(1PG24.16E2)
60    CONTINUE
C
      WRITE(US,70)
70    FORMAT(1X,'et de borne superieure : ')
      DO 80,I=1,N
         WRITE(US,40)SUP(I)
80    CONTINUE
C
C     On initialise COMPTBIN, T et PROBLEM :
      DO 100,I=1,N
         COMPTBIN(I)=0
100   CONTINUE
      T=0
      PROBLEM=.TRUE.
C
C
C
C     Corps de la procedure :
C     -----------------------
```

```
C
C
C
C     On a 2**N sommets a examiner.
200   IF (T.LT.((2**N))) THEN
C
C        1 on evalue le sommet
         DO 300,K=1,N
            IF (COMPTBIN(K).EQ.0) THEN
               SOMMET(K)=INF(K)
            ELSE
               SOMMET(K)=SUP(K)
            ENDIF
300      CONTINUE
         U=T+1
C
C        2 on calcule beta(M)
         VALSOM=F(SOMMET)
         IF (T.EQ.0) THEN
            BEPART=VALSOM
         ELSE
            IF (VALSOM.LT.BEPART) THEN
               BEPART=VALSOM
            ENDIF
         ENDIF
C
C        3 on calcule alpha(M)
C        3a on regarde d'abord si le sommet est dans D, le
C           domaine des contraintes :
         DEDANS=.TRUE.
         I=1
400      IF (DEDANS .AND. I.LE.M) THEN
            IF (GI(SOMMET,CONTR,M,N,I,NMAX,MMAX).GT.0) THEN
               DEDANS=.FALSE.
            ENDIF
            I=I+1
            GOTO 400
         ENDIF
C        3b On ajuste ALPART et POINT si le sommet est dans D,
C           le domaine des contraintes :
         IF (DEDANS) THEN
            IF (PROBLEM) THEN
               ALPART=VALSOM
               DO 500,H=1,N
                  POINT(H)=SOMMET(H)
500            CONTINUE
               PROBLEM=.FALSE.
            ELSE
               IF (VALSOM.LT.ALPART) THEN
                  ALPART=VALSOM
                  DO 600,H=1,N
                     POINT(H)=SOMMET(H)
600               CONTINUE
               ENDIF
            ENDIF
         ENDIF
C
C        4 on passe au point suivant a l'aide de l'instruction
C          REPETER
         I=1
```

```
700         CONTINUE
            IF (COMPTBIN(I).EQ.1) THEN
                COMPTBIN(I)=0
            ELSE
                COMPTBIN(I)=1
            ENDIF
            I=I+1
          IF (COMPTBIN(I-1).EQ.1) GOTO 700
C
          T=T+1
          GOTO 200
        ENDIF
C
C
C
C       Si aucun sommet du N-rectangle examine n'est admissible,
C       on donne a ALPART une valeur plus grande que toutes les
C       valeurs prises par la fonction objective dans le
C       domaine des contraintes et a POINT, on affecte un point
C       n'appartenant pas au domaine :
C
        IF  (PROBLEM) THEN
            ALPART=1000000.
            DO 800,H=1,N
               POINT(H)=1000000.
800         CONTINUE
        ENDIF
C
C       On affiche ALPART et BEPART a l'unite de sortie :
C
        WRITE(US,900)BEPART
900     FORMAT(1X,'La valeur de BEPART est : ',1PG24.16E2)
C
        WRITE(US,1000)ALPART
1000    FORMAT(1X,'La valeur de ALPART est : ',1PG24.16E2)
C
C
        END
C
C
C
C*******************************************************************
C*******************************************************************
C*******************************************************************
C*******************************************************************
C
C
C
        SUBROUTINE ENLEVE(MAXPART,FINPART,ACCES,BETA,MINALPHA)
C
C
C       Effet :
C       -----
C       Cette sous-routine passe en revue chaque element M de
C       la partition et l'elimine (cad le rend inaccessible)
C       si il verifie Beta(M) >= MINALPHA a l'aide du vecteur
C       ACCES. C'est le pas r1 de l'algoritnme.
C
C
C       Declaration des arguments :
```

```
C      -------------------------
C
       INTEGER MAXPART,FINPART
       LOGICAL ACCES(MAXPART)
       DOUBLE PRECISION MINALPHA,BETA(MAXPART)
C
C
C      Table des arguments :
C      ---------------------
C      MAXPART : nombre maximum d'elements admis dans la
C                partition
C      FINPART : nombre de vecteurs stockes + 1 (<= MAXPART)
C      ACCES : tableau de booleen tel que
C              ACCES(J) = .TRUE. ssi le Jeme element de la
C              partition est encore accessible cad non
C              elimine par DR ou r1
C      MINALPHA : reel contenant la valeur minimale de la
C                 fonction obtenue jusque-la
C      BETA : vecteur tel que son Jeme element contient la
C             valeur du BETA relatif au Jeme element de
C             la partition
C
C
C      Declaration des variables locales :
C      -----------------------------------
       INTEGER T
C
C
C      Table des variables locales :
C      -----------------------------
C      T : indice compteur de boucle
C
C
C
C      Corps de la procedure :
C      -----------------------
C
C
C
       DO 100,T=1,FINPART-1
          IF ( (ACCES(T)) .AND. (BETA(T).GE.MINALPHA) ) THEN
             ACCES(T)=.FALSE.
          ENDIF
100    CONTINUE
C
       END
C
C
C
C**********************************************************************
C**********************************************************************
C**********************************************************************
C**********************************************************************
C
C
C
       SUBROUTINE BISSEC(J,MAXPART,N,FINPART,TABSUP,TABINF,NMAX,
      +                  COMPT)
C
C
```

```
C      Effet :
C      -----
C      Cette sous-routine bissecte le Jeme element de la partition.
C      Elle stocke les 2 N-rectangles construits en Jeme et en
C      FINPARTeme colonne de TABINF et de TABSUP. La variable
C      FINPART reste inchangee.C'est le pas r2 de l'algorithme.
C
C
C      Declaration des arguments :
C      -------------------------
       INTEGER J,MAXPART,N,FINPART,NMAX,COMPT
       DOUBLE PRECISION TABINF(NMAX,MAXPART),TABSUP(NMAX,MAXPART)
C
C
C      Table des arguments :
C      -------------------
C       J : numero de l'element a partitionner ( < FINPART )
C       MAXPART : borne sur le nombre d'elements admis dans la
C                 partition
C       N : dimension du probleme (<= NMAX)
C       FINPART : nombre de vecteurs stockes + 1 (<=MAXPART)
C       NMAX : borne sur la dimension du probleme
C       COMPT : indice compteur de boucle
C       TABINF : tableau tq la Jeme colonne contient la borne
C                inferieure du Jeme element de la partition
C       TABSUP : tableau tq la Jeme colonne contient la borne
C                superieure du Jeme element de la partition
C
C
C      Declaration des variables locales :
C      ---------------------------------
       INTEGER K,DIMLONG
       DOUBLE PRECISION SUBLONG,LONG,LO
C
C
C      Table des variables locales :
C      ---------------------------
C      K : indice compteur de boucle
C      DIMLONG : dimension suivant laquelle le N-rectangle est
C                le plus long
C      SUBLONG : longueur des N-rectangles crees dans la dimen-
C                sion DIMLONG
C      LONG : longueur du N-rectangle partitionne dans la dimen-
C             sion DIMLONG
C      LO : reel de stockage de longueur du N-rectangle traite
C
C
C
C      Corps de la procedure :
C      ---------------------
C
C
C      1 on cherche la dimension suivant laquelle le N-rectangle
C        est le plus long : DIMLONG de longueur LONG
C
       LONG=TABSUP(1,J)-TABINF(1,J)
       DIMLONG=1
       DO 100,K=2,N
          LO=TABSUP(K,J)-TABINF(K,J)
          IF (LO.GT.LONG) THEN
```

```
            LONG=LO
            DIMLONG=K
         ENDIF
100   CONTINUE
C
      SUBLONG = TABSUP(DIMLONG,J)-TABINF(DIMLONG,J)
      SUBLONG = SUBLONG / 2.
C
C
C     2 on stocke celui de droite, cad celui de borne supe-
C       rieure identique, mais dont la borne inferieure
C       change dans la dimension DIMLONG. On le stocke
C       en FINPARTeme colonne :
C
      DO 200,K=1,N
         TABINF(K,FINPART)=TABINF(K,J)
         TABSUP(K,FINPART)=TABSUP(K,J)
200   CONTINUE
      TABINF(DIMLONG,FINPART)= TABINF(DIMLONG,J) + SUBLONG
C
C
C     3 on stocke celui de gauche, cad de borne inferieure
C       identique mais dont la borne superieure change
C       dans la dimension DIMLONG. On le stocke en Jeme
C       colonne :
C
      TABSUP(DIMLONG,J) = TABINF(DIMLONG,J) + SUBLONG
C
C
      END
C
C
C
C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
C
C
C
      SUBROUTINE DONNEES(FCHNOM,N,M,CONTR,BI,BS,EPS,
     +                   NMAX,MMAX,US)
C
C
C     Effet :
C     -----
C     Cette procedure enregistre les donnees necessaires a la
C     resolution du probleme traite se trouvant dans un fichier
C     FCHNOM.dat . De plus, elle va mettre dans un fichier de
C     sortie represente par US les donnees principales ( N, M,
C     CONTR, BI, BS, EPS ) afin de servir de rappel pour l'in-
C     terpretation des resultats.
C     Le fichier des donnees (qui contient 1 information par
C     ligne commencant en colonne 1) doit etre de la forme :
C       1ere ligne : N
C       2eme ligne : M
C       (*Ensuite, pour I allant de 1 a M : *)
C                    CONTRAINTE
C                    I
C                    (*Pour K allant de 1 a N : *)
```

```
C                       Pik
C                       Qik
C                       Rik
C         (*Des que toutes les contraintes y sont : *)
C                       FINCONTR
C         (*Suivi de la borne inferieure introduite de la facon
C           suivante : *)
C                       BORNE INF
C                       (*Pour I allant de 1 a N : *)
C                       la Ieme coordonnee de cette borne
C         (*Suivi de la borne superieure introduite de la facon
C           suivante : *)
C                       BORNE SUP
C                       (*Pour I allant de 1 a N : *)
C                       la Ieme coordonnee de cette borne
C         (*Finalement, on introduit la precision EPS de la facon
C           suivante : *)
C                       EPS
C                       La precision voulue sur la valeur de la
C                       fonction
C
C
C       Declaration des arguments :
C       ---------------------------
        INTEGER N,M,NMAX,MMAX,US
        CHARACTER*9 FCHNOM
        DOUBLE PRECISION CONTR(MMAX,NMAX,3),BI(NMAX),
       +                 BS(NMAX),EPS
C
C
C       Table des arguments :
C       ---------------------
C        N : dimension du probleme (<= NMAX)
C        M : nombre de fonctions Gi (<= MMAX)
C        NMAX : borne superieure sur la dimension du probleme
C               traite
C        MMAX : borne superieure sur le nombre de contraintes
C               Gi
C        US : entier representant l'unite de sortie
C        FCHNOM : fichier de donnees traite
C        CONTR : tableau contenant les reels definissant les
C                fonctions-contraintes Gi, separables quadra-
C                tiques :
C                CONTR(I,K,1) = Pik
C                CONTR(I,K,2) = Qik
C                CONTR(I,K,3) = Rik
C        BI : borne inferieure du probleme traite
C        BS : borne superieure du probleme traite
C        EPS : precision voulue sur la valeur de la solution
C
C
C       Declaration des variables locales :
C       -----------------------------------
        INTEGER UL,IOS,I,NBRE,K
        CHARACTER*10 BLABLA
C
C
C       Table des variables locales :
C       -----------------------------
C       UL : entier representant l'unite de lecture
```

```
C      IOS : variable de gestion d'erreur
C      I : entier compteur de boucle
C      NBRE : variable de controle de la correction du fichier
C             de donnees
C      K : entier compteur de boucle
C      BLABLA : variable de controle de la correction du fichier
C               de donnees
C
C
C
C      Corps de la procedure :
C      ----------------------
C
C
C
C      1. On ouvre le fichier de donnees :
C
       UL=10
       OPEN(UNIT=UL,ERR=900,FILE=FCHNOM,STATUS='OLD',IOSTAT=IOS)
C
C      2. On enregistre les valeurs de N et M et on les inscrit
C         dans US :
C
       READ(UL,5,IOSTAT=IOS,ERR=1000)N
       READ(UL,5,IOSTAT=IOS,ERR=1000)M
5      FORMAT(I5)
C
       WRITE(US,10)N
10     FORMAT(1X,'La dimension du probleme = ',1x,I3)
       WRITE(US,20)M
20     FORMAT(1X,'Le nombre de fonctions Gi = ',1X,I3)
C
C
C      3. On enregistre le tableau CONTR :
C
       DO 500,I=1,M
          READ(UL,30)BLABLA
30        FORMAT(A10)
          IF (BLABLA.NE.'CONTRAINTE') THEN
             WRITE(US,8000)
             CLOSE(UL)
             STOP
          ENDIF
          READ(UL,*,IOSTAT=IOS,ERR=7000)NBRE
          IF (NBRE.NE.I) THEN
             WRITE(US,8000)
             CLOSE(UL)
             STOP
          ENDIF
          DO 400,K=1,N
             READ(UL,*,IOSTAT=IOS,ERR=3000)CONTR(I,K,1)
             READ(UL,*,IOSTAT=IOS,ERR=3000)CONTR(I,K,2)
             READ(UL,*,IOSTAT=IOS,ERR=3000)CONTR(I,K,3)
400       CONTINUE
500    CONTINUE
C
       READ(UL,30)BLABLA
       IF (BLABLA.NE.'FINCONTR') THEN
           WRITE(US,8000)
           CLOSE(UL)
```

```
          STOP
        ENDIF
C
C
C       4. On enregistre la borne inferieure du probleme et
C          on l'inscrit dans US :
C
        READ(UL,30)BLABLA
        IF (BLABLA.NE.'BORNE INF') THEN
            WRITE(US,8000)
            CLOSE(UL)
            STOP
        ENDIF
        DO 600,K=1,N
           READ(UL,*,IOSTAT=IOS,ERR=5000)BI(K)
600     CONTINUE
C
C
        WRITE(US,625)
625     FORMAT(1X,'La borne inferieure = ')
        DO 650,K=1,N
           WRITE(US,*)BI(K)
650     CONTINUE
C
C
C       5. On enregistre la borne superieure du probleme et on
C          l'inscrit dans US :
C
        READ(UL,30)BLABLA
        IF (BLABLA.NE.'BORNE SUP') THEN
            WRITE(US,8000)
            CLOSE(UL)
            STOP
        ENDIF
        DO 700,K=1,N
           READ(UL,*,IOSTAT=IOS,ERR=5000)BS(K)
700     CONTINUE
C
        WRITE(US,725)
725     FORMAT(1X,'La borne superieure = ')
        DO 750,K=1,N
           WRITE(US,*)BS(K)
750     CONTINUE
C
C       6. On enregistre la precision voulue pour le probleme ,
C          on l'inscrit dans US et on ferme le fichier de
C          donnees :
C
        READ(UL,30)BLABLA
        IF (BLABLA.NE.'EPS') THEN
            WRITE(US,8000)
            CLOSE(UL)
            STOP
        ENDIF
        READ(UL,*,IOSTAT=IOS,ERR=9000)EPS
C
        CLOSE(UL)
C
        WRITE(US,800)EPS
800     FORMAT(1X,'La precision voulue :',1X,1PG24.16E2)
```

```
      WRITE(US,825)
825   FORMAT(///)
      WRITE(US,850)
850   FORMAT(1X,3(50('*'),1X,/))
C
C
C
C      Les messages a afficner en cas d'erreur du fichier de
C      donnees :
C
900   IF (IOS.NE.0) THEN
        WRITE(US,950)
950     FORMAT(1X,'Le fichier de donnees ne convient pas.',/,
     +          1X,'Probleme a l''ouverture. ')
        CLOSE(UL)
        STOP
      ENDIF
C
1000  IF (IOS.NE.0) THEN
        WRITE(US,2000)
2000    FORMAT(1X,'Le fichier ne convient pas. Probleme ',/,
     +          1X,'a la lecture de N ou M')
        CLOSE(UL)
        STOP
      ENDIF
C
3000  IF (IOS.NE.0) THEN
        WRITE(US,4000)
4000    FORMAT(1X,'Le fichier ne convient pas. La matrice ',/,
     +          1X,'des contraintes n''est pas correcte. ')
        CLOSE(UL)
        STOP
      ENDIF
C
5000  IF (IOS.NE.0) THEN
        WRITE(US,6000)
6000    FORMAT(1X,'Le fichier ne convient pas. Une borne ',/,
     +          1X,'du probleme n''est pas correcte. ')
        CLOSE(UL)
        STOP
      ENDIF
C
7000  IF (IOS.NE.0) THEN
        WRITE(US,8000)
8000    FORMAT(1X,'Le fichier ne convient pas. Un mot de ',/,
     +          1X,'controle n''est pas correct. ')
        CLOSE(UL)
        STOP
      ENDIF
C
9000  IF (IOS.NE.0) THEN
        WRITE(US,10000)
10000    FORMAT(1X,'Le fichier ne convient pas. La precision',/,
     +          1X,'n''est pas correcte. ')
        CLOSE(UL)
        STOP
      ENDIF
C
C
      END
```

```
C
C
C
C**********************************************************************
C**********************************************************************
C**********************************************************************
C**********************************************************************
C
C
C
      DOUBLE PRECISION FUNCTION GI(X,CONTR,M,N,I,NMAX,MMAX)
C
C
C     Effet :
C     -----
C     Cette fonction calcule la valeur que prend un
C     point X pour la fonction Gi (i donne), fonction
C     quadratique separable dont les coefficients
C     sont donnes par le tableau CONTR.
C
C     Declaration des arguments :
C     ---------------------------
C
      INTEGER NMAX,MMAX,N,M,I
      DOUBLE PRECISION CONTR(MMAX,NMAX,3),X(NMAX)
C
C     Table des arguments :
C     --------------------
C      NMAX : borne superieure sur la dimension du probleme
C             traite
C      MMAX : borne superieure sur le nombre de contraintes
C             Gi
C      N : dimension du probleme (<= NMAX)
C      M : nombre de fonctions Gi (<= MMAX)
C      I : indice (compris entre 1 et M) designant le numero
C          de la contrainte dont l'evaluation est souhaitee
C      CONTR : tableau contenant les reels definissant les
C              fonctions-contraintes Gi, separables quadra-
C              tiques :
C              CONTR(I,K,1) = Pik
C              CONTR(I,K,2) = Qik
C              CONTR(I,K,3) = Rik
C      X : point auquel on evalue la fonction
C
C     Declaration des variables locales :
C     -----------------------------------
C
      INTEGER K
C
C     Table des variables locales :
C     -----------------------------
C     K : indice compteur de boucle
C
C     Initialisation :
C     ----------------
C
      GI=0
C
C     Corps de la procedure :
C     -----------------------
```

```
C
      DO 100,K=1,N
         GI = GI + (1./2.)*CONTR(I,K,1)*(X(K)**2)
     +           + CONTR(I,K,2)*X(K)
     +           + CONTR(I,K,3)
100   CONTINUE
C
C
      END
```

References

[1] R. Horst et H. Tuy, "Convergence and restart in Branch-and-Bound algorithms for global optimization. Application to concave minimization and d.c. optimization problems", Mathematical Programming 41 (1988) 161-183.

[2] R. Horst, "A general class of Branch-and-Bound methods in global optimization with some new approaches for concave minimization", JOTA 51 (1986), 271-291.

[3] F. Al-Khayyal, R. Horst et P. Pardalos, "Global optimization of concave functions subject to separable quadratic constraints and of all-quadratic separable problems", rapport technique PDRC 88-04, Georgia Institute of Technology, Atlanta, Georgie, U.S.A. (1988)

[4] R. Horst, "An algorithm for nonconvex programming problems", Mathematical Programming, Vol 10, pp 312-321, 1976.

[5] R. Horst, "Deterministic global optimization with partition sets whose feasibility is not known: application to concave minimization, reverse convex constraints, dc - programming and Lipschitzian optimization ,,, JOTA 58 (1988), 11-37.

[6] Strodiot J-J, Nguyen VH, "Taille optimale de diamant : Méthode Diam 1 ,,, rapport technique n° 79/5. Departement Mathématique F.U.N.D.P., 1979.