Points intérieurs et plans coupants : mise en œuvre et développement d'une méthode pour l'optimisation convexe et la programmation linéaire structurée de grande taille.

Olivier du Merle

Pour l'obtention du grade de Docteur ès science économiques et sociales mention économie commercial et industrielle.

Thèse présentée à la Faculté des Sciences Economiques et Sociales l'Université de Genève, Département HEC.

Membre du jury de thèse :
-M. Manfred GILLI, Professeur, Genève
-M. Jean-Louis GOFFIN, Professeur, Université McGill, Montréal
-M. Alain HAURIE, Professeur, Genève, Président du jury
-M. Brigitte JAUMARD, Professeur, Ecole polytechnique de Montréal
-M. Jean-Philippe VIAL, Professeur, Genève, Directeur de thèse

Thèse N^o 414
Genève, 1995

La Faculté des Sciences Economiques et Sociales, sur préavis du jury, a autorisé l'impression de la présente thèse, sans entendre, par là, émettre aucune opinion sur les propositions qui s'y trouvent énoncées et qui n'engagent que la responsabilité de leur auteur.

Genève, le 24 Mars 1995

Le Doyen
Paolo URIO

Impression d'après le manuscrit de l'auteur

à mes parents,
à Nathalie.

Remerciements

Je tiens à exprimer mes vifs remerciements au professeur J.P. Vial, directeur de thèse, sans qui j'aurais été dans l'impossibilité de réaliser ce travail. Je lui suis reconnaissant pour son soutien permanent et pour les moyens mis à ma disposition durant toutes ces années. Par sa patience il a su me faire découvrir et apprécier l'univers de l'optimisation. J'aimerais aussi exprimer mon admiration pour son attitude loyale aussi bien dans le cadre scientifique que dans les rapports humains.

Mes remerciements s'adressent aussi à J.L. Goffin pour ses nombreux conseils et encouragements prodigués. Avec J.P. Vial, il est à l'origine de la méthode sans laquelle mon travail n'aurait jamais pu voir le jour. De plus, j'ai beaucoup apprécié les séances de travail passées en sa compagnie.

Je remercie également les professeurs A. Haurie et M. Gilli d'avoir acceptés de juger cette thèse, en ayant une attention particulière à A. Haurie pour avoir accepté de présider le jury.

Je tiens également à remercier B. Jaumard non seulement pour avoir accepté de juger cette thèse mais aussi pour m'avoir fait découvrir et apprécier le monde de la recherche. C'est en effet dans le cadre de mon mémoire d'Ingénieur que, sous sa direction, j'ai été initié aux techniques de la Recherche Opérationnelle.

Je souhaite aussi exprimer ma reconnaissance à E. Fragnière, J. Gondzio et R. Sarkissian collègues de travail pour m'avoir supporté, encouragé et aidé tout au long de ces années passées à l'Université de Genève.

Je remercie aussi M. Doggwiler et S. Lasne pour s'être donné la peine de relire cette thèse.

Enfin, et afin de n'oublier personne, je remercie la terre entière et le ciel avec...

Table des matières

Liste des figures

Liste des tableaux

Introduction

Sujet de la recherche

L'optimisation offre un cadre mathématique permettant d'interpréter et de résoudre un grand nombre de problèmes de Gestion, d'Economie, de Mathématique et de Physique.

Dans ce mémoire nous supposerons la pré-existence d'un modèle mathématique et d'un critère à optimiser. Le modèle mathématique est un premier pas vers une analyse quantitative et qualitative d'un système. Il contient un ensemble de variables (stock d'une ressource, mesure d'une grandeur physique, achat ou vente...) liées entres elles par des relations, dites contraintes (bornes admissibles sur les états ou les commandes, limites de production d'une usine, équation d'évolution du système...). Le critère que l'on cherche à optimiser peut être une énergie, un coût, une mesure physique par rapport à un état désiré du système, un temps ou toute autre grandeur sur laquelle les variables peuvent influer.

L'optimisation consiste à fournir une ou plusieurs solutions optimales du modèle mathématique. Si la phase de modélisation a été faite correctement, elle permettra non seulement d'améliorer la gestion du système en question, mais aussi de mieux cerner celui-ci.

La capacité et la puissance des ordinateurs ne permettent pas et ne permettront jamais de résoudre tous les problèmes posés. Même lorsque le système que l'on cherche à analyser est modélisé dans sa plus simple expression, la recherche d'une solution n'est pas un problème trivial. De surcroît, le nombre de difficultés augmente lorsque les modèles sont affinés pour dépeindre la réalité de manière plus précise.

Pour résoudre ces modèles il faut faire appel aux techniques les plus avancées de la programmation mathématique en considérant aussi bien le problème théorique que le problème pratique induit par toute implantation informatique.

En 1947, Dantzig proposa un algorithme dénommé “méthode du simplexe” permettant de résoudre les problèmes de programmation linéaire. Il donna alors naissance à la programmation mathématique, en offrant un outil puissant permettant d'analyser un grand nombre de systèmes. Cependant, la programmation linéaire resta longtemps le seul type de modélisation que l'on puisse réellement optimiser. Nombreuses furent alors les applications pratiques qui ont été modélisées comme telles.

Pour offrir un champ plus large à l'optimisation et afin d'affiner les modélisations il faut pouvoir aller au delà de l'aspect linéaire. L'optimisation globale (pour les fonctions quelconques) restant encore mal maîtrisée, on se contente souvent de modèles où les fonctions linéaires sont remplacées par des fonctions convexes. Si des algorithmes efficaces existent dans le cas différentiable, des difficultés surgissent lorsque l'on se place dans le cadre plus général de l'optimisation convexe non différentiable.

L'algorithme qui va faire l'objet de cette étude a été proposé par Goffin, Haurie et Vial. Par la suite nous y ferons référence en le nommant algorithme ACCPM (Analytic Center Cutting Plane Method). Il appartient à la famille des méthodes de plans coupants et plus particulièrement à la sous-famille des méthodes des centres. Il découle des récentes recherches sur les méthodes de points intérieurs dont l'impulsion a été donnée en 1984 par Karmarkar.

L'objet de la thèse a été de mettre en œuvre cet algorithme et d'y apporter les modifications et les précisions nécessaires pour que d'un concept théorique naisse un algorithme fiable et robuste. Dans le document qui va suivre nous allons essayer de donner une description aussi complète que possible de cette méthode aussi bien d'un point de vue théorique que d'un point de vue pratique. Nous terminerons par quelques illustrations numériques de son comportement. Nous verrons à cette occasion qu'il donne de très bons résultats pour des problèmes aussi variés que la programmation géométrique, les problèmes minmax, les problèmes de décomposition, les problèmes de multiflots, de dimensionnement de structures métalliques, etc...

En conclusion, l'algorithme ACCPM offre une méthode de résolution puissante et efficace pour l'optimisation convexe non différentiable.

Contexte de la recherche

Le cadre de cette thèse est donc l'optimisation convexe non différentiable. Si des méthodes d'optimisation existent dans le cas différentiable, nous verrons que celles-ci ne peuvent pas être utilisées pour résoudre des problèmes non différentiables. Pour surmonter cette difficulté, des méthodes particulières ont été développées.

Ces méthodes peuvent être classées en deux catégories. La première regroupe les méthodes qui dérivent de la méthode du gradient (méthode du sous-gradient et méthode des ellipsoïdes) ; la second contient les méthodes de plans coupants. Si les meilleures variantes de la première peuvent avoir une complexité[1] théorique optimale, elles ne donnent pas de bons résultats en pratique. De plus il est difficile de leur définir un critère d'arrêt. A contrario, on trouve dans les méthodes de plans coupants des méthodes ayant une piètre estimation de la complexité mais un bon comportement en pratique (méthodes de faisceaux, l'agorithme ACCPM).

Les méthodes de plans coupants peuvent être, elles mêmes, divisées en deux catégories. Celles qui proposent comme solution optimale un point sur la frontière de la relaxation linéaire courante du problème et celles qui proposent un point à l'intérieur de l'ensemble convexe compact formé par la relaxation linéaire et une borne supérieure (ensemble de localisation). Dans la première catégorie on trouve la méthode de Kelley et les méthodes de faisceaux ; dans la seconde on trouve la méthode d'Elzinga et Moore, la méthode du centre de gravité et la méthode volumétrique. L'algorithme ACCPM se range dans cette dernière catégorie.

Dans le cadre de l'optimisation convexe non différentiable, la méthode de Kelley (1960) est la plus ancienne des méthodes de plans coupants. Elle est aussi, probablement, la plus facile à mettre en œuvre. Nous verrons cependant que l'estimation de sa complexité dans le pire cas est très médiocre. Par opposition à la méthode de Kelley, la méthode du centre de gravité est la plus efficace, d'un point de vue théorique, des méthodes des centres. Malheureusement, sa mise en œuvre est irréalisable. Entre ces deux extrêmes nous trouvons la méthode d'Elzinga et Moore, la méthode volumétrique et, plus récemment, les méthodes de faisceaux et l'algorithme ACCPM.

[1]En nombre d'itérations pour déterminer l'optimum (pour une précision donnée).

Contenu du mémoire

Méthodes générales

Dans un première partie nous allons décrire le contexte général de l'optimisation convexe non différentiable.

Dans le premier chapitre nous introduirons la problématique liée à l'optimisation convexe. Nous donnerons tout d'abord les définitions de base (convexité, sous-gradient, problème à optimiser), puis nous présenterons les différents types de problèmes qui peuvent être résolus et enfin, nous introduirons les méthodes standards (pour l'optimisation différentiable) en mentionnant les difficultés rencontrées lorsqu'on les applique aux problèmes non différentiables.

Dans les deux chapitres suivants nous ferons un rapide tour d'horizon sur les différentes méthodes existantes pour résoudre les problèmes non différentiables. Dans le premier chapitre nous verrons la méthode des sous-gradients et la méthode des ellipsoïdes ; dans le second nous décrirons les différentes méthodes de plans coupants. Dans le chapitre sur les méthodes de plans coupants nous aborderons tout d'abord l'algorithme de Kelley puis les méthodes de faisceaux et enfin nous verrons une famille de méthodes appelées méthodes des centres (méthode d'Elzinga et Moore, méthode du centre de gravité et méthode volumétrique).

Théorie sur l'algorithme ACCPM

Dans les trois chapitres qui vont suivre nous allons voir les éléments fondamentaux de l'algorithme ACCPM. Ces chapitres décrivent l'algorithme de manière relativement générale mais indiquent les procédures indispensables à mettre en œuvre pour pouvoir le faire fonctionner.

Le quatrième chapitre pose les fondements théoriques de l'algorithme ACCPM. Nous verrons tout d'abord son principe puis nous développerons la notion de relaxation linéaire, de coupes, d'oracle, d'ensemble de localisation et de centre analytique. Nous donnerons ensuite un rapide aperçu de l'algorithme et terminerons par un tour d'horizon sur l'état actuel des études faites sur sa convergence théorique.

L'algorithme ACCPM est une procédure itérative. A chaque itération il émet une proposition comme solution optimale. Dans le cinquième chapitre nous développerons les calculs permettant de déterminer cette proposition. Ce point est dénommé centre analytique. Nous verrons de plus comment construire des ellipsoïdes centrés en ce point et contenant toutes les solutions optimales. Nous verrons aussi comment calculer un centre analytique pondéré, c'est-à-dire un point qui associe une importance plus ou moins grande aux différentes coupes.

Le sixième chapitre regroupe les différents ingrédients qui, d'un point de vue pratique, sont nécessaires pour faire fonctionner la méthode. Nous y trouverons la recherche unidimensionnelle, le calcul d'une borne inférieure, la technique pour ajouter une coupe à la relaxation linéaire courante et le raffinement de la direction de Newton utilisée pour le calcul du centre analytique. Nous verrons aussi comment procéder pour utiliser l'algorithme quand aucune borne sur la valeur de la solution n'est connue.

Implantation de l'algorithme ACCPM

Dans cette dernière partie nous allons voir comment procéder pour implanter l'algorithme ACCPM. Il s'agit d'une application pratique de ce qui a été développé durant la partie précédente.

Le septième chapitre développe les techniques avancées utilisées lors de la mise en œuvre de l'agorithme. Nous y verrons comment gérer le concept de multicoupes, de phase initiale de l'algorithme et comment éliminer de la relaxation linéaire courante les coupes inutiles. Enfin nous terminerons par une description complète de l'algorithme ACCPM.

Le huitième chapitre donne la description des principales procédures intervenant dans l'algorithme. Il permet d'avoir un aperçu global de la méthode et de synthétiser ainsi les informations des chapitres précédents. Il a aussi pour objectif de faciliter la tâche de celui qui désirerait implanter l'algorithme ACCPM.

Le dernier chapitre regroupe toutes les études expérimentales que nous avons menées. Nous ferons tout d'abord une rapide description des problèmes et de leur oracle puis nous analyserons le comportement de l'algorithme lorsque l'on fait varier les quelques paramètres de la méthode. Enfin nous donnerons les résultats numériques obtenus pour chaque problème et nous finirons cet exposé par une comparaison avec la méthode de Dantzig-Wolfe.

Partie I

METHODES GENERALES

Chapitre 1

Généralités

Les trois chapitres de cette première partie font un rapide tour d'horizon sur les méthodes existantes pour résoudre les problèmes d'optimisation convexe non différentiable.

Dans le chapitre qui va suivre nous allons tout d'abord donner les définitions de bases liées à la convexité, la non différentiabilité et la minimisation d'une fonction possédant ces caractéristiques. Ensuite nous verrons quelques types de problèmes convexes non différentiables que l'on peut être amener à résoudre. Cette section montrera les transformations usuelles qui aboutissent à la résolution d'un problème convexe non différentiable. Enfin, nous donnerons un rapide aperçu des méthodes que l'on peut appliquer dans le cas différentiable et nous indiquerons les difficultés que l'on rencontre lorsqu'on les applique aux problèmes non différentiables.

1.1 Définitions de base

1.1.1 Ensemble convexe, fonction convexe

Un ensemble convexe se définit ainsi ([16]) :

Définition : 1.1 *Un sous-ensemble de C de $\mathbb{R}^n$ est dit convexe si* $(1-\lambda)x+\lambda y \in C$ *pour tout* $x \in C$, $y \in C$ *et* $0 < \lambda < 1$.

De nombreuses propriétés découlent de cette définition. Elles ne feront pas l'objet de notre étude et seront supposées connues. On se réfèrera à l'ouvrage de Rockafellar [16] pour plus d'informations.

Avant de définir une fonction convexe définissons l'épigraphe d'une fonction :

Définition : 1.2 *Soit f une fonction définie sur X, sous-ensemble de $\mathbb{R}^n$, et prenant ses valeurs dans $\mathbb{R}$, l'ensemble*

$$\text{epi } f = \{(x, z) : x \in X, z \in \mathbb{R}, z \geq f(x)\}$$

est appelé l'épigraphe de f (noté epi f*).*

En se basant sur les définitions 1.1 et 1.2, on donne la définition suivante pour une fonction convexe :

Définition : 1.3 *Soit f une fonction définie sur X un sous-ensemble de $\mathbb{R}^n$ et prenant ses valeurs dans $\mathbb{R}$, f est dite convexe si* epi f *est un sous-ensemble convexe de $\mathbb{R}^{n+1}$.*

Les fonctions convexes se partagent en deux catégories : les fonctions propres et les fonctions impropres. Une fonction convexe est dite propre si son épigraphe est non vide et s'il ne contient pas de lignes verticales, c'est-à-dire si $f(x) < +\infty$ pour au moins un x et si $f(x) > -\infty$ pour tout x. Bien que les fonctions impropres existent, elles ne présentent que peu d'intérêt et nous ferons généralement l'hypothèse que la fonction f étudiée est propre. Un exemple de fonction convexe impropre est :

$$f(x) = \begin{cases} -\infty & \text{si } |x| < 1 \\ 0 & \text{si } |x| = 1 \\ +\infty & \text{si } |x| > 1. \end{cases}$$

1.1.2 Dérivée directionnelle, sous-gradient et sous-différentiel

Les fonctions convexes possèdent de nombreuses propriétés. Nous nous contenterons ici d'en énoncer les principales.

Dérivée directionnelle

La dérivée directionnelle d'une fonction f est définie ainsi :

$$f'(x,d) = \lim_{t \to 0} \frac{f(x+td) - f(x)}{t}. \tag{1.1}$$

Dans notre cas, comme f est une fonction convexe, si on suppose qu'elle est finie en x la dérivée directionnelle a les propriétés suivantes [16] :

- elle existe dans toutes les directions d,
- $\frac{f(x+td)-f(x)}{t}$ est une fonction non-décroissante de $t > 0$,
- $f'(x,d) = \inf_{t>0} \frac{f(x+td)-f(x)}{t}$,
- $f'(x,0) = 0$,
- $-f'(x,-d) \leq f'(x,d)$.

Considérons par exemple, la fonction suivante :

$$f(x) = \begin{cases} -x & \text{si} \quad x \leq 0 \\ x^2 & \text{si} \quad x > 0. \end{cases} \tag{1.2}$$

Pour x différent de 0 le gradient de f est défini ($\nabla f(x) = -1$ si $x < 0$ et $\nabla f(x) = 2x$ si $x > 0$). En $x = 0$, le gradient n'est pas défini, mais les deux limites $f'_-(0) = -f'(0,-1) = -1$ et $f'_+(0) = f'(0,1) = 0$ caractérisent le comportement (du premier ordre) de f au voisinage de 0.

Sous-gradient et sous-différentiel

Le vecteur g est un sous-gradient de la fonction convexe f au point x^k si

$$f(x) \geq f(x^k) + g^T(x - x^k), \qquad \forall x.$$

Cette condition, que l'on nomme "inégalité du sous-gradient", a une signification géométrique simple lorsque f est finie en x^k. Elle dit que le graphe de la fonction affine $f(x) = f(x^k) + g^T(x - x^k)$ est un hyperplan de support non vertical de l'ensemble convexe *epi* f au point $(x^k, f(x^k))$. Cette remarque nous sera particulièrement utile dans les méthodes de plans coupants que nous verrons plus tard. L'ensemble des sous-gradients de f au point x est appelé sous-différentiel et est noté $\partial f(x)$.

Le sous-différentiel est un ensemble convexe qui se réduit au gradient si f est différentiable en x. Si cet ensemble est non vide, on dit alors que f est sous-différentiable en x.

Pour la fonction f mentionnée ci-dessus, on obtient :

$$\partial f(x) = \begin{cases} -1 & \text{pour} \quad x < 0 \\ \{g \in \mathbb{R} : -1 \leq g \leq 0\} & \text{pour} \quad x = 0 \\ 2x & \text{pour} \quad x > 0. \end{cases}$$

Un exemple de fonction non partout sous-différentiable est :

$$f(x) = \begin{cases} -(1 - |x^2|)^{1/2} & \text{pour} \quad |x| \leq 1 \\ +\infty & \text{sinon.} \end{cases}$$

Cette fonction, bien que sous-différentiable (voir même différentiable) pour tout $|x| < 1$, ne l'est pas quand $|x| \geq 1$ (c.à.d. $\partial f(x) = \emptyset$ si $|x| \geq 1$).

Il existe une relation entre le sous-différentiel et la dérivée directionnelle. Un théorème fondamental de l'analyse convexe [16] établit que la dérivée directionnelle $f'(x; .)$ est une fonction de support de $\partial f(x)$. Nous avons la relation suivante :

$$f'(x, d) \geq g^T d, \qquad \text{avec} \quad g \in \partial f(x),$$

et dans le cas où la fonction convexe est propre nous avons l'égalité :

$$f'(x, d) = \max_{g \in \partial f(x)} g^T d. \tag{1.3}$$

On le vérifie aisément pour l'exemple (1.2) puisque sa dérivée directionnelle est :

$$f'(0, d) = \begin{cases} -d & \text{pour} \quad d \leq 0 \\ 0 & \text{pour} \quad d > 0. \end{cases}$$

On remarquera que si f est convexe, g est un de ses sous-gradients en x^k si et seulement si $\binom{g}{-1}$ est le vecteur normal d'un hyperplan de support dans $\mathbb{R}^{n+1}$

du graphe de f en $\binom{x^k}{f(x^k)}$. Ce qui conduit à une autre caractéristique du sous-différentiel :

$$\partial f(x^k) = \{g \in \mathbb{R}^n : g^T(x - x^k) \leq f(x) - f(x^k) \quad \forall x \in \mathbb{R}^n\}. \tag{1.4}$$

1.1.3 Minimum d'une fonction convexe

L'ensemble convexe de niveau α de f (noté $\text{lev}_\alpha f$) est défini de la façon suivante :

$$\text{lev}_\alpha f = \{x : f(x) \leq \alpha\}, \qquad\qquad \alpha \in \mathbb{R}.$$

Soit f^* le minimum de $f(x)$ pour $x \in \mathbb{R}^n$, $\text{lev}_\alpha f = \emptyset$ pour $\alpha < f^*$ et $\text{lev}_{f^*} f$ est appellé l'ensemble minimum de f. Il apparaît clairement que l'étude de cet ensemble minimum est de première importance : est-il vide ou non ? contient-il un point unique ?... Si f est strictement convexe l'ensemble minimum ne pourra contenir plus d'un point. Cependant, dans tous les cas, c'est un sous-ensemble convexe de $\mathbb{R}^n$.

Une condition nécessaire et suffisante pour un point x donné d'appartenir à l'ensemble minimum est : $0 \in \partial f(x)$. D'où la condition nécessaire et suffisante d'optimalité :

$$x^* \text{est optimal} \Longleftrightarrow 0 \in \partial f(x^*). \tag{1.5}$$

ou bien, en utilisant la dérivée directionnelle :

$$x^* \text{est optimal} \Longleftrightarrow f'(x^*, d) \geq 0, \quad \forall d.$$

L'ensemble X^* des points optimaux est donc caractérisé par :

$$X^* = \{x^* \in \mathbb{R}^n : 0 \in \partial f(x^*)\}$$

Nous ferons par la suite l'hypothèse que cet ensemble est non vide et borné.

1.1.4 Minimum d'une fonction convexe sous contraintes convexes

L'objet de notre étude est la résolution du problème :

$$\begin{aligned} \text{Minimiser} \quad & f(x) \\ \text{s.c.} \quad & x \in X, \end{aligned} \tag{1.6}$$

où $f : \mathbb{R}^n \to \mathbb{R}$ est une fonction convexe, et X est un ensemble convexe non vide. Par la suite nous ferons les hypothèses suivantes :

Hypothèse : 1.1 *En tout point x de $X \in \mathbb{R}^n$, $f(x)$ est connue et on peut extraire un vecteur g de l'ensemble des sous-gradients ($g \in \partial f(x)$).*

Hypothèse : 1.2 *L'une des deux propositions suivantes au moins doit être satisfaite :*

- *l'ensemble X est compact,*
- *la fonction f est infcompacte[1] sur X.*

Remarques :

- On ne fait aucune hypothèse sur la façon dont g est choisie dans $\partial f(x)$.
- La deuxième hypothèse garantit que la fonction admet un minimum sur l'ensemble X même si celui ci n'est pas compact.

L'algorithme ACCPM qui est à l'origine de ce travail, résoud directement ce problème. Cependant, d'autres méthodes exigent que l'on se ramène au contexte plus restreint de la minimisation d'une fonction convexe sans contraintes. Pour pouvoir les utiliser deux transformations sont possibles : par pénalisation ou par dualisation.

Transformation par pénalisation

L'idée consiste à associer aux contraintes du problème (1.6) une fonction de pénalité $h(x)$:

$$h(x) = \begin{cases} 0 & \text{si } \ x \in X, \\ \geq 0 & \text{sinon.} \end{cases}$$

Par exemple si $X = \{x : g(x) \leq 0\}$, la fonction de pénalité pourrait être : $h(x) = \max\{0, g(x)\}$.

[1] On dit qu'une fonction est infcompacte sur X si et seulement si $\mathrm{lev}_\alpha f \cap X$ est compact sur X.

On peut alors définir le problème suivant :

$$\min_x f_\mu(x) = f(x) + \mu h(x). \tag{1.7}$$

On peut montrer [12] que sous certaines hypothèses[2] la solution du problème (1.7) converge vers la solution du problème (1.6) quand $\mu \to +\infty$. La solution du problème (1.6) est donc approchée par la minimisation du problème (1.7).

Il est à noter que même si, dans l'exemple, les fonctions f et g sont différentiables, la fonction barrière (1.7) ne l'est pas nécessairement. C'est le prix à payer pour l'intégration des contraintes dans l'objectif.

Transformation par dualisation

Reprenons l'exemple :

$$\begin{aligned} \text{Minimiser} \quad & f(x) \\ \text{s.c.} \quad & g(x) \leq 0. \end{aligned} \tag{1.8}$$

La méthode consiste à écrire le Lagrangien de ce problème :

$$L(x, \lambda) = f(x) + \lambda g(x), \qquad \text{avec} \quad \lambda \in \mathbb{R}^+.$$

Le théorème de Kuhn et Tucker [16] indique que si (x^*, λ^*) est un point de selle de ce Lagrangien (c.à.d $\forall x, \forall \lambda \quad L(x^*, \lambda) \leq L(x^*, \lambda^*) \leq L(x, \lambda^*)$) alors x^* est l'optimum du problème original (1.8). L'existence d'un point de selle est garantie dans le cas convexe. Le Lagrangien est une fonction convexe-concave. Nous verrons plus tard comment il peut être résolu (sous-section §1.2.3 : “Recherche de point de selle”).

1.2 Problèmes résolus par ces méthodes

Ce travail se place dans le contexte général de la recherche du minimum d'une fonction convexe sur un ensemble convexe (problème (1.6)). De nombreux problèmes font partie de ce contexte. Par exemple, la simple recherche du minimum d'une fonction convexe (sans contraintes). Plus concrètement, voici quelques problèmes

[2] $\lim_{x^k \to x} f(x^k) \geq f(x)$, $\lim_{\|x\| \to +\infty} f(x) = \infty$ et f bornée inférieurement.

types auxquels on peut être confronté et que les méthodes que nous allons voir vont pouvoir résoudre.

1.2.1 Problème minmax

Dans de nombreuses applications la fonction à minimiser est définie comme le maximum d'un ensemble de fonctions. Ce problème s'écrit :

$$\min_{x \in \mathbb{R}^n} \max_{1 \le i \le m} f_i(x).$$

On remarquera que, même si les fonctions f_i sont toutes différentiables sur leur ensemble de définition, le problème "minmax" ci-dessus n'est pas différentiable en tout point. Un exemple classique de ce type de problème est la minimisation d'un vecteur dans la norme l_1 ou la norme l_∞ :

$$\begin{aligned} &\min_{x \in \mathbb{R}^n} \|f_1(x), f_2(x), \ldots, f_m(x)\|_1, \\ &\min_{x \in \mathbb{R}^n} \|f_1(x), f_2(x), \ldots, f_m(x)\|_\infty. \end{aligned}$$

Récrits sous forme minmax ces problèmes deviennent respectivement :

$$\begin{aligned} &\min_{x \in \mathbb{R}^n} \max\{\pm f_1(x) \pm f_2(x) \pm \ldots \pm f_m(x)\}, \\ &\min_{x \in \mathbb{R}^n} \max\{f_1(x), -f_1(x), \ldots, f_m(x), -f_m(x)\}. \end{aligned}$$

Une autre classe de problèmes qui peuvent être traités par ce type de formulation est le problème de réalisabilité.

$$\text{trouver } x \text{ tel que} \quad f_i(x) \le 0, \quad i = 1, \ldots, m.$$

Le problème s'écrit alors :

$$\min_x \max\{f_1(x), \ldots, f_m(x), 0\}.$$

1.2.2 Décomposition

Dans de nombreux cas, la taille du problème ou l'exploitation de sa structure implique que l'on va chercher, pour le résoudre, à le décomposer. Nous pouvons définir deux types de décompositions : par les variables ou par les contraintes.

Décomposition par les variables

Si le problème (1.6) peut s'écrire de la manière suivante :

$$\begin{aligned} \text{Minimiser} \quad & f_1(x) + f_2(y) \\ \text{s.c.} \quad & g_1(x) + g_2(y) \leq 0, \end{aligned}$$

on peut le récrire :

$$\text{Minimiser} \quad f_1(x) + h(x), \tag{1.9}$$

où

$$h(x) = \begin{cases} +\infty & \text{si} \quad \{y : g_1(x) + g_2(y) \leq 0\} = \emptyset \\ \min\{f_2(y) : g_1(x) + g_2(y) \leq 0\} & \text{sinon.} \end{cases}$$

On vérifiera facilement que l'ensemble $\{y : g_1(x) + g_2(y) \leq 0\}$ est convexe pour tout x et que $h(x)$ l'est aussi sur cet ensemble.

Ainsi, si y est un vecteur de grande dimension et x de petite dimension, le principe de décomposition transforme un problème d'optimisation de grande taille en (x, y) en un problème d'optimisation de petite taille en x. Cependant, il ne faut pas négliger le fait que pour une valeur de x donnée il nous faudra trouver la valeur de h. La nature de ce dernier problème peut donc grandement influencer l'intérêt de la décomposition.

Un exemple intéressant est celui de la résolution du programme linéaire de grande taille ayant la structure particulière suivante :

$$\begin{aligned} \text{Minimiser} \quad & c_0^T x_0 + \sum_{i=1}^{m} c_i^T x_i \\ \text{s.c.} \quad & A_i x_0 + B_i x_i \leq b_i, \qquad i = 1, \ldots, m. \end{aligned}$$

Etant donnée la structure de ce problème, il est possible de le décomposer. Dans une première étape on résoud, pour un vecteur x_0 fixé, l'ensemble des sous-problèmes linéaires suivants :

$$g_i(x_0) = \min_x \{c_i^T x : B_i x \leq b_i - A_i x_0\}$$

Le problème original se réduit alors au problème suivant :

$$\min_{x_0} \{c_0^T x_0 + \sum_{i=1}^{m} g_i(x_0)\}.$$

On constatera, que cette reformulation permet de travailler avec un problème de plus petite taille. Le prix à payer est la perte de la notion de linéarité et de différentiabilité.

Décomposition par les contraintes

Pour éliminer les contraintes dans un problème de minimisation on en écrit le Lagrangien. Nous avons vu que nous sommes alors confrontés à un problème minmax (recherche de point de selle). Bien qu'il puisse être résolu de manière frontale (sous-section §1.2.3 : "Recherche de point de selle"), on peut aussi le résoudre en le décomposant. Plus précisément, le Lagrangien du problème (1.8) :

$$L(x,u) = f(x) + u^T g(x), \qquad \text{avec} \quad u \geq 0,$$

peut être exploité de deux manières : en minimisant une fonction $L_P(x)$ (problème primal) ou en cherchant le maximum d'une fonction $L_D(u)$ (problème dual).

Plus précisément, le problème primal est le suivant :

$$\min_x L_P(x),$$

$$\text{avec} \quad L_P(x) = \max_{u\geq 0} L(x,u) = \begin{cases} +\infty & \text{si} \quad g(x) > 0 \\ f(x) & \text{sinon.} \end{cases}$$

et le problème dual :

$$\max_{u\geq 0} L_D(u),$$

$$\text{avec} \quad L_D(u) = \min_x L(x,u) = \min_x f(x) + u^T g(x).$$

La fonction $L_P(x)$ est convexe ($f(x)$ est convexe) et $L_D(u)$ est concave (minimum d'une collection de formes linéaires). Le premier ne présente pas d'intérêt puisqu'il est de même dimension que le problème original (1.8). Comme de plus, sous certaines conditions élémentaires[3] [16],

$$\min_x L_P(x) = \max_{u\geq 0} L_D(u),$$

la résolution du problème dual est donc équivalente à la résolution du problème primal. Pour peu que la dimension de u soit plus petite que celle de x, on choisit

[3] Conditions garanties si f et g sont convexes et si x appartient à un ensemble convexe fermé et bornée.

alors de résoudre le problème dual :

$$\max_u \min_x \{f(x) + u^T g(x)\}, \qquad u \in \mathbb{R}^+. \tag{1.10}$$

Comme précédemment, il est à noter que même si les fonctions f et g sont différentiables, le problème (1.10) ne l'est pas nécessairement. C'est le prix à payer pour l'intégration des contraintes dans l'objectif. De plus on a deux problèmes : une minimisation en x (pour un u fixé) incluse dans la recherche d'un maximun en u.

1.2.3 Recherche de point de selle

Soit la fonction convexe-concave $f(x, y)$ où $x \in C_x$ et $y \in C_y$ avec C_x et C_y deux ensembles convexes et compacts, un point (x^*, y^*) est dit point de selle s'il satisfait l'égalité suivante :

$$\max\{f(x^*, y) : y \in C_y\} = f(x^*, y^*) = \min\{f(x, y^*) : x \in C_x\}.$$

Etant donné qu'à l'optimum,

$$\min_{x\in C_x} \max_{y\in C_y} f(x, y) = \max_{y\in C_y} \min_{x\in C_x} f(x, y),$$

il est équivalent, pour trouver un point de selle, de résoudre l'un ou l'autre de ces problèmes. La difficulté de ces problèmes provient du fait que les objectifs sont définis implicitement (minimum d'un maximum et inversement).

Pour déterminer l'optimum (x^*, y^*) on procède alors différemment. On minimise la fonction convexe :

$$f(x, y) = \overline{f}(x) - \underline{f}(y),$$

où $\overline{f}(x) = \max_{y\in C_y} f(x, y)$ et $\underline{f}(y) = \min_{x\in C_x} f(x, y)$.

Malheureusement, il n'est généralement pas possible de déterminer le gradient et la valeur de cette fonction f en un point (x, y). Pour contourner cette difficulté, plusieurs méthodes peuvent être utilisées. Elles supposent cependant, qu'il est possible de calculer la valeur de $f(x, y)$, un de ses sous-gradients $f'_x(x, y)$ en x et un des ses super-gradients $f'_y(x, y)$ en y.

Une premiere méthode consiste à agréger les gradients en un seul vecteur :

$$g(x, y) = \begin{pmatrix} f'_x(x, y) \\ f'_y(x, y) \end{pmatrix}.$$

En effet, nous avons la propriété suivante [13] :

Théorème : 1.1 *Si* $z^* = (x^*, y^*)$ *est un point de selle de* f *alors*

$$(z - z^*)^T g(z^*) \geq 0, \qquad \forall z \in \mathrm{int}(C_x \times C_y).$$

On retrouve la condition d'optimalité qui utilise la dérivée directionnelle. Une méthode itérative (méthode de plans coupants) permet de trouver un tel point [13].

Une autre méthode approxime la fonction $f(x, y)$ par :

$$f_k(x, y) = \overline{f}_k(x) - \underline{f}_k(y),$$

où

$$\begin{aligned} \overline{f}_k(x) &= \max_i \{f(x_i, y_i) + f'_x(x_i, y_i)^T (x - x_i) : 1 \leq i \leq k\}, \\ \underline{f}_k(y) &= \min_i \{f(x_i, y_i) + f'_y(x_i, y_i)^T (y - y_i) : 1 \leq i \leq k\}. \end{aligned}$$

Une méthode itérative permet d'en déterminer le minimum [11].

1.2.4 Inégalité variationnelle d'un opérateur monotone

Généralités

Soit l'opérateur monotone[4] multi-valué[5] $F(x) : X \to \mathbb{R}^n$ où X est un sous-ensemble fermé convexe de $\mathbb{R}^n$, soit G un sous-espace fermé convexe de $\mathbb{R}^n$ avec un intérieur non vide, la paire (G, F) définit l'inégalité variationnelle :

$$\text{trouver} \quad x^* \in G \cap X \quad \text{et} \quad y^* \in F(x^*) \quad \text{tel que} \quad y^{*T}(x - x^*) \geq 0, \quad \forall x \in G.$$

Ce point x^* est appelé solution forte de l'inégalité variationnelle. Comme F est monotone, si x^* est une solution forte, pour tout $x \in \mathrm{int}G$ et $y \in F(x)$, on a :

$$y^T(x - x^*) \geq y^{*T}(x - x^*) \geq 0.$$

[4] L'opérateur multi-valué $F(x) : X \to \mathbb{R}^n$ où X est un sous-ensemble fermé convexe de $\mathbb{R}^n$ est dit monotone si $(x - \bar{x})^T(y - \bar{y}) \geq 0 \quad \forall x, \bar{x} \in X \quad y \in F(x), \quad \bar{y} \in F(\bar{x})$.

[5] L'image de $F(x)$ sur $x \in X$ est un sous-ensemble non vide de $\mathbb{R}^n$.

Donc, si x^* est une solution forte de l'inégalité variationnelle alors

$$y^T(x - x^*) \geq 0, \qquad \forall x \in \mathrm{int}G, \quad y \in F(x).$$

Le point x^* satisfaisant cette dernière relation est appelé solution faible de l'inégalité variationnelle. Comme nous venons de le voir, toute solution forte est aussi solution faible. L'inverse n'est généralement pas vrai mais, nous verrons qu'il le devient si l'on pose de légères restrictions sur l'opérateur F. Pour des raisons qui apparaîtront claires plus tard, il est plus raisonnable de chercher les solutions faibles de l'inégalité variationnelle. C'est pourquoi nous les appellerons simplement solutions de l'inégalité variationnelle.

Exemples

Minimum d'une fonction convexe : soit $f(x)$ une fonction convexe continue et finie définie sur G, et soit $F(x) = \partial f(x)$ l'opérateur qui à x associe l'ensemble de ses sous-gradients. On vérifiera aisément que cet opérateur est monotone et que les solutions fortes de l'inégalité variationnelle (G, F) ainsi définie ne sont autres que les minima de f sur G. De plus Nemirovskii [13] montre que pour ce type de problème, les solutions faibles sont identiques aux solutions fortes. Il montre aussi à cette occasion que l'ensemble des solutions faibles est plus stable que celui des solutions fortes puisque c'est un sous-ensemble fermé convexe non vide de G.

Point de Kuhn-Tucker pour l'optimisation sous contraintes : considérons le problème suivant :

$$\begin{aligned} \text{Minimiser} \quad & f(u) \\ \text{s.c.} \quad & g_i(u) \leq 0, \qquad i = 1, \ldots, p, \end{aligned}$$

et supposons, pour simplifier que les contraintes g_i et l'objectif f sont finies et continûment différentiables sur $\mathbb{R}^k \times \mathbb{R}$. Un point $x^*(u^*, \lambda^*) \in \mathbb{R}^n = \mathbb{R}^k \times \mathbb{R}^p$ est un point de Kuhn-Tucker s'il satisfait à la condition :

$$\begin{aligned} f'(u^*) + \lambda^{*T} g'(u^*) &= 0 \\ \lambda &\geq 0 \\ g_i(u^*) &\leq 0, \qquad i = 1, \ldots, p \\ \lambda_i^* g_i(u^*) &= 0, \qquad i = 1, \ldots, p. \end{aligned}$$

Une inégalité variationnelle (G, F) qui identifie le point de Kuhn-Tucker est

$$F(x) = F(u, \lambda) = (f'(u) + \lambda^T g'(u), -g_1(u), \ldots, -g_p(u)) : \mathbb{R}^n \to \mathbb{R}^{p+1},$$

avec $G = \{x = (u, \lambda) \in \mathbb{R}^n : \lambda \geq 0\}$.

Nemirovskii [13] montre que toute solution forte de cette inégalité est un point de Tuhn-Tucker et inversement. De plus, si le problème est convexe (f et g_i sont convexes) l'opérateur F est monotone sur G et les solutions faibles sont identiques aux solutions fortes.

Autres exemples :

1. Nemirovskii [13] montre qu'il est possible de décrire une inégalité variationnelle permettant de trouver l'équilibre de Nash. De plus, il montre que sous certaines hypothèses de convexité sur les fonctions de gain les solutions faibles sont des équilibres de Nash.

2. Lemaréchal, Nemirovskii et Nesterov [11] ont montré dans un cadre plus général que si l'un ou l'autre des cas suivants est vérifié :

   (a) Dom$F \subseteq G$ et F est mono-valué et continue sur G,

   (b) F est maximal[6] monotone,

   la solution faible est aussi solution forte de l'inégalité variationnelle.

Résolution

Les exemples ci-dessus montrent que la recherche d'une solution faible offre un large champ d'application. Nous allons voir maintenant comment les déterminer. Remarquons tout d'abord que x^* est une solution faible si toute forme linéaire $y^T(x - x^*)$ avec $y \in F(x)$ est positive ou nulle en x^* ou, de manière équivalente, x^* est une solution faible si toute forme linéaire $y^T(x^* - x)$ où $y \in F(x)$ est négative ou nulle en x^*. Une solution faible est donc un point où la fonction convexe[7] :

$$f(x) = \sup\{y^T(x - \tilde{x}) : \tilde{x} \in G, y \in F(\tilde{x})\},$$

[6] Un opérateur multi-valué F est maximal monotone s'il n'existe pas un autre opérateur multi-valué F' monotone tel que graphe$F \subset$ grapheF' si graphe$F \neq$ grapheF'.

[7] $f(x)$ est le suprémum d'un ensemble de fonctions linéaires

est négative ou nulle. La résolution du problème se fait donc par la minimisation d'une fonction convexe. Si pour tout point x il est possible de calculer une valeur de f, une méthode itérative (méthode de plans coupants) permet de trouver une solution à ce problème [13] [11].

On remarquera que la fonction

$$\tilde{f}(x) = \sup\{y^T(x - \tilde{x}) : \tilde{x} \in G, y \in F(x)\}$$

associée à la recherche d'une solution forte n'est pas nécessairement convexe.

1.2.5 Autres problèmes

Pour finir, il est à noter que les méthodes utilisées pour résoudre les problèmes non-différentiables peuvent aussi donner de bons résultats avec certains problèmes différentiables. C'est le cas de problèmes où le gradient varie rapidement. Ces problèmes difficiles à résoudre par les méthodes classiques sont alors abordés comme des problèmes non-différentiables.

La liste ci-dessus n'est pas exhaustive. Elle a simplement pour but de mettre en lumière quelques exemples d'applications possibles.

1.3 Difficultés avec les méthodes standards

1.3.1 Principales méthodes pour l'optimisation différentiable

Les méthodes classiques d'optimisation de fonctions convexes différentiables approximent f en x par un modèle linéaire ou quadratique. Comme, dans tous les cas, la stationnarité de f est une condition nécessaire d'optimalité, pratiquement toutes les méthodes d'optimisation sans contraintes dans $\mathbb{R}^n$ consistent à rechercher un point x^* stationnaire, c'est-à-dire chercher un point x^* tel que $0 \in \partial f(x^*)$. Ces méthodes opérant sur des fonctions différentiables elles cherchent donc un point x^* tel que $\nabla f(x^*) = 0$.

Quand elles approximent f en x par un modèle linéaire ($f(x+d) \approx f(x) + \nabla^T f(x)d$) on dit qu'elles résolvent le problème par une méthode du premier ordre et quand elles approximent f en x par un modèle quadratique ($f(x+d) \approx f(x) + \nabla^T f(x)d + \frac{1}{2}d^T \nabla^2 f(x)d$), qu'elles résolvent le problème par une méthode du deuxième ordre.

Toutes ces techniques sont des procédures itératives où l'on engendre une suite de points $x^0, x^1, \ldots, x^k$ convergeant vers un optimum de f. A chaque étape k, x^{k+1} est défini par : $x^{k+1} := x^k + \lambda_k d_k$ où d_k est une direction de déplacement choisie de façon plus ou moins arbitraire de manière à ce qu'elle soit une direction de "descente" c'est-à-dire que $\nabla f^T(x^k) d_k < 0$.

Les méthodes du premier ordre

Les méthodes de gradient [14] : leur direction de descente est $d_k = -\frac{\nabla f(x^k)}{||\nabla f(x^k)||}$. Elles diffèrent les unes des autres par la stratégie utililisée pour choisir le paramètre $\lambda_k \geq 0$. Leur convergence n'est assurée que si $\lim_{k\to\infty} \lambda_k = 0$ et $\sum_{k=0}^{\infty} \lambda_k = +\infty$. Elles présentent l'inconvénient d'avoir une convergence très lente. Par contre, elles offrent l'intérêt de se généraliser au cas où les fonctions ne sont pas partout différentiables.

La méthode de plus forte pente (Steepest descent) [2], [3] : sa direction de descente est $d_k = -\nabla f(x^k)$ et λ_k est déterminé en faisant une recherche unidimensionnelle suivant cette direction : $\lambda_k = \text{argmin}_{\lambda \geq 0} f(x^k + \lambda d_k)$. Si la fonction est continûment différentiable avec la propriété $\lim_{||x|| \to +\infty} f(x) = +\infty$, alors, pour tout point de départ x^0, cette méthode converge vers un point stationnaire de f. Cette convergence peut cependant être très lente. On prouve aisément [12] que deux directions successives sont orthogonales. Ainsi pour minimiser des fonctions mal conditionnées du type "vallée étroite et allongée" le nombre d'itérations peut être considérable.

La méthode de plus forte pente "accélérée" [9] : sa direction de descente est $d_k = y^k - x^k$ où y^k est le point obtenu à partir du point x^k après p itérations de la méthode de plus forte pente. Comme dans cette dernière, λ_k est déterminé par une recherche unidimensionnelle : $\lambda_k = \text{argmin}_{\lambda \geq 0} f(x^k + \lambda d_k)$. L'intérêt de cette méthode par rapport à la version ordinaire de la méthode de plus forte pente se vérifie facilement. En particulier, pour $p = n$, on démontre [9] que la vitesse de convergence est identique à celle de la méthode du gradient conjugué que nous verrons par la suite. Cependant, de par le fait qu'elle nécessite un grand nombre de recherches unidimensionnelles et un grand nombre d'évaluations de la fonction f et de son gradient, elle est assez coûteuse et peu utilisée dans la pratique.

La méthode du gradient conjugué [8] : sa direction de descente est $d_{k+1} = -\nabla f(x^{k+1}) + \beta_k d_k$ avec $\beta_k = \frac{||\nabla f(x^{k+1})||^2}{||\nabla f(x^k)||^2}$ et $d_0 = -\nabla f(x^0)$. Comme dans la méthode de plus forte pente, λ_k est déterminé par une recherche unidimensionnelle : $\lambda_{k+1} = \text{argmin}_{\lambda \geq 0} f(x^{k+1} + \lambda d_{k+1})$. Cette méthode est très intéressante, d'une part parce qu'elle nécessite le stockage de très peu d'informations et d'autre part, par sa vitesse de convergence très supérieure à celle des méthodes classiques de gradients. Il est important de noter que la convergence globale n'est assurée que si l'on procède à une réinitialisation périodique (par exemple toutes les n itérations le point courant est considéré comme le point x_0.)

Les méthodes du deuxième ordre

Si maintenant la fonction f est deux fois continûment différentiable et que l'on sait calculer en tout point ses dérivées secondes, il est alors possible d'utiliser la méthode de Newton. L'idée consiste à remplacer, au voisinage du point courant x^k, la fonction f par son approximation quadratique :

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

On prend alors comme point x^{k+1} le minimum de $q(x)$, lorsqu'il existe. Ceci ne peut être le cas que si $\nabla^2 f(x^k)$ est une matrice définie positive. La fonction $q(x)$ est alors strictement convexe et a un minimum unique x^{k+1} défini par $\nabla q(x^{k+1}) = 0$, qui conduit au systéme linéaire :

$$\nabla f(x^k) = -\nabla^2 f(x^k)(x^{k+1} - x^k).$$

D'où la forme itérative :

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

Une extension naturelle consiste à remplacer $[\nabla^2 f(x^k)]^{-1}$ par une matrice H_k définie positive. On a alors la forme itérative :

$$x^{k+1} := x^k - \lambda_k H_k \nabla f(x^k).$$

On parle alors de méthodes quasi newtoniennes (ou méthodes à métrique variable). Beaucoup de variantes ont été proposées pour choisir H_k. En général H_k est définie en fonction de H_{k-1}. On dit que l'on fait une correction : $H_{k+1} := H_k + \Delta_k$.

Correction de rang 1 [15] : dans ces méthodes la matrice Δ_k est de rang 1. Elles présentent l'avantage de ne pas nécessiter de recherche unidimensionnelle. En revanche, elle peuvent conduire à d'importantes instabilités numériques.

Correction de rang 2 : ces méthodes ne présentent pas les inconvénients de celles faisant intervenir une correction de rang 1. Par contre elles nécessitent une recherche unidimensionnelle. Les deux principales sont :

1. L'algorithme de Davidon [4] Fletcher-Powell [7] (DFP). La convergence est généralement plus rapide qu'avec la méthode du gradient conjugué. Cependant, cette méthode nécessite une recherche unidimensionnelle exacte qui peut s'avérer relativement coûteuse et occulter l'avantage de sa convergence. De plus, comme pour la méthode du gradient conjugué, la convergence n'est assurée que si elle est périodiquement initialisée.
2. L'algorithme de Broyden [1], Flecher [6], Goldfarb [10], Shanno [17] (BFGS). Cet algorithme conserve les avantages de la méthode (DFP) mais évite son inconvénient en proposant simplement une recherche unidimensionnelle approchée. Il doit lui aussi, pour assurer sa convergence, être périodiquement réinitialisé.

1.3.2 Difficultés rencontrées avec l'optimisation non différentiable

La convergence des méthodes standards vers l'optimum n'est plus assurée lorsque le gradient n'est pas défini en tout point, c'est-à-dire lorsque la fonction n'est plus partout différentiable. En effet, quand on s'approche des zones où le gradient n'est pas défini, ces méthodes ne fournissent plus une bonne approximation de f. Ainsi, si le minimum est atteint en un point x^* où la fonction est non différentiable (ce qui est souvent le cas), alors la direction de recherche proposée devient pratiquement inutilisable dans un voisinage de la solution. La méthode BFGS, par exemple, cesse de donner de bons résultats même pour une précision de 10^{-2} ou 10^{-3} [19].

Cela peut même être pire. Ces méthodes peuvent converger vers un point non optimal. Considérons par exemple une variante d'un problème proposé par Wolfe [18]. La fonction est la suivante :

$$f(z_1, z_2) = \begin{cases} 5(9z_1^2 + 16z_2^2)^{1/2} & \textit{pour} \quad z_1 \geq |z_2| \\ 9z_1^2 + 16|z_2| & \textit{pour} \quad 0 < z_1 < |z_2| \\ 9z_1^2 + 16|z_2| - z_1^9 & \textit{pour} \quad z_1 \leq 0. \end{cases} \tag{1.11}$$

Le contour de f est représenté dans la figure 1.1. Il est aisé de vérifier que $x^* = (-1, 0)^T$ est l'optimal global de f et que ce point se trouve sur la demi-droite $(z_1 \leq 0, z_2 = 0)$ où f est non différentiable. De plus la fonction f est convexe. Si maintenant on choisit un point quelconque dans la région $z_1 > |z_2| > (9/16)^2|z_1|$, le chemin parcouru par la méthode de plus grande pente converge vers $x = (0, 0)$ qui n'est pas l'optimum.

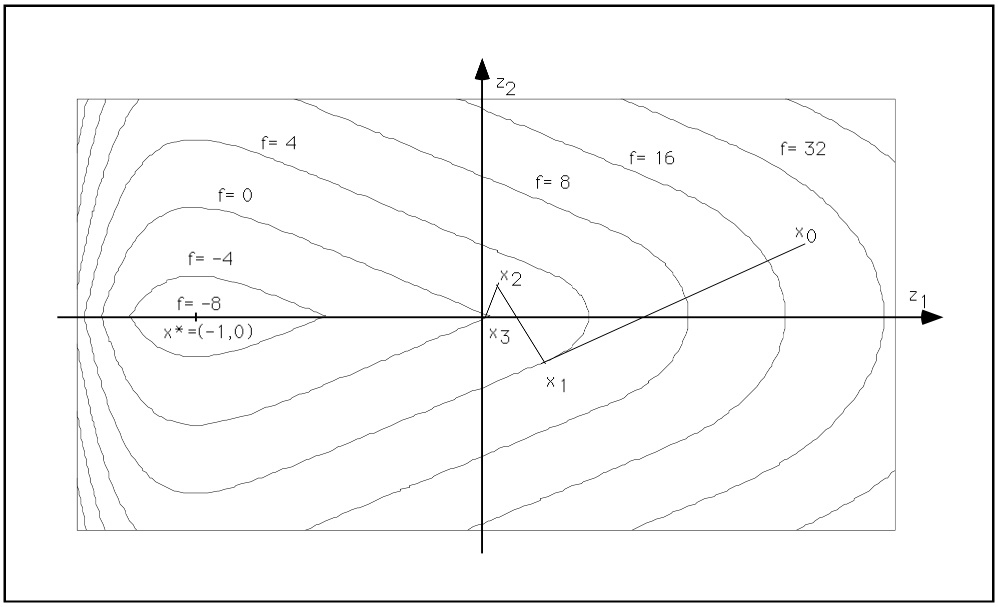

Figure 1.1: Contour de f et itérations avec la méthode de plus grande pente.

Enfin, même pour les fonctions avec lesquelles les méthodes classiques ne convergent pas vers un mauvais optimum, la question du test d'optimalité reste entière. Pour les fonctions différentiables, il suffit de constater que la norme du gradient converge vers zéro quand x converge vers l'optimum x^*. Le test d'arrêt peut alors s'écrire :

$$\|\nabla f(x_k)\| < \varepsilon,$$

où $\varepsilon > 0$ est la tolérance (ou précision) demandée. Cependant, dans le cas non différentiable, ce critère n'a plus de sens, même si le gradient est défini à chaque itération. Prenons par exemple le cas de la fonction $f(x) = |x|$. Pour tout $x_k \neq 0$, $\|\nabla f(x_k)\| = 1$. Le point x_k peut être aussi proche de l'optimum $x^* = 0$ que l'on veut, le test sur la norme du gradient n'apportera aucune information.

Il résulte de l'analyse précédente que les méthodes classiques, pour les problèmes non différentiables, risquent d'une part de converger vers un point non optimal et d'autre part de ne plus offrir de test d'arrêt valide. D'autres approches doivent donc être envisagées pour résoudre ces problèmes.

Bibliographie

[1] C. G. Broyden, “The convergence of a class of double-rank minimization algorithms 2 : the new algorithm”, *Journal Institute of Math. and its Appl.* **6** (1970) 222-231.

[2] A.L. Cauchy, “Méthode générale pour la résolution des systèmes d'équations simultanées”, *Comptes Rendus Acad. Sc. Paris, XXV* (1847) 536-538.

[3] H. Curry, “The method of steepest descent for nonlinear minimization problems”, *Quart. Appl. Math.* **2** (1944) 258-261.

[4] W. C. Davidon, “Variable metric method for minimization”, Research and Development Report ANL-5990 (ref.) U.S. Atomic Energy Commission Argonne National Laboratories (1959).

[5] Yu. M. Ermoliev, “Methods of solution of nonlinear extremal problems”, *Cybernetics 2* **4** 1-16.

[6] R. Fletcher, “A new approach to variable metric algorithms”, *The Computer Journal* **3** (1970) 317-322.

[7] R. Fletcher, M. J. D. Powell, “A rapidly convergent descent method for minimization”, *Computer Journal* **6** (1963) 163-168.

[8] R. Fletcher, C. M. Reeves, “Function minimization by conjugate gradients”, *Computer Journal* **7** (1964) 149-154.

[9] G. E. Forsythe, “On the asymptotic directions of the s-dimensional optimum gradient method”, *Numerische Mathematik* **11** (1968) 57-76.

[10] D. Goldfarb, “A family of variable metric methods derived by variational means”, *Mathematics of Computation* **24** (1970) 23-26.

[11] C. Lemaréchal, A. Nemirovskii et Y. Nesterov, “New variants of bundle methods”, Technical Report 1508, Institut National de Recherche en Informatique

et en Automatique, Domaine de Voluceau, Rocquencourt, B.P. 105, 78153 La Chesnay Cedex, France (September 1991).

[12] M. Minoux, *Programmation mathématique : théorie et algorithmes*, **2** Dunod (Bordas), Paris (1983).

[13] A. Nemirovskii *unpublished Lecture notes* Technion, The Israel Institute of Technology, Faculty of Industrial Engineering & Management (1994).

[14] B.-T. Polyak, “A general method of solving extremum problems”, *Soviet Mathematics* **8** (1966) 593-597.

[15] M. J. D. Powell, “Recent advances in unconstrained optimisation”, *Mathematical Programming* **1** (1971) 26-57.

[16] R.-T. Rockafellar, *Convex analysis*, Princeton University Press, (1972).

[17] D. F. Shanno, “Conditioning of quasi-Newton methods for function minimization”, *Mathematics of Computation* **24** (1970) 641-656.

[18] P. Wolfe, “A method of conjugate subgradients for minimizing nondifferentiable convex functions”, *Mathematical Programming* **3** (1975) 145-173.

[19] J. Zowe, *Nondifferentiable optmization*, NATO ASI Series F15, Computational Mathematical Programming, ed. K. Schittkowski, Springer-Verlag Berlin Heidelberg (1985) 323-356.

Chapitre 2

Optimisation non différentiable : algorithmes

Nous avons vu au chapitre précédent les différentes méthodes que l'on pouvait utiliser dans le cadre de l'optimisation différentiable. Nous avons vu de plus les limites de ces méthodes lorsqu'on les applique à des problèmes non différentiables. Nous allons maintenant faire un rapide tour d'horizon sur les méthodes utilisées pour contourner ces difficultés.

De manière générale, il existe deux façons de procéder : l'approche indirecte et l'approche directe.

- L'approche indirecte élimine le caractère non différentiable du problème en résolvant un problème voisin mais différentiable. Après transformation du problème original en problème différentiable les méthodes du chapitre précédent peuvent alors être utilisées. Cependant, il n'est pas toujours possible de trouver un problème différentiable ayant la même solution que le problème original. Seuls les problèmes minmax et les problèmes de norme l_1 peuvent être abordés ainsi.

- L'approche directe résoud "directement" le problème original. Elle ne nécessite donc pas une "récriture différentiable" du problème. De plus elle reste indifférente au fait que f soit différentiable ou non. Les seules exigences portent sur le fait que la fonction doit être convexe et que les hypothèses (1.1) et (1.2) doivent être vérifiées (Connaissance de la valeur de f et d'un de ses sous-gradients en tout point x. Existence d'une solution optimale). Les méthodes développées dans cette optique présentent donc l'avantage d'être plus générales que les méthodes classiques puisqu'elles peuvent aussi résoudre les problèmes différentiables. Elles ont par contre l'inconvénient majeur de ne pas pouvoir être considérées comme des méthodes de descente et possèdent donc un taux de convergence médiocre. Nous verrons dans le chapitre qui va suivre la méthode du sous-gradients (et quelques variantes) et la méthode des ellipsoïdes.

Il existe aussi d'autres méthodes directes appelées les “two stage methods” (voir Conn [3], Hornung [11], Fletcher [5] et Womersley [23]). Cependant, comme pour l'approche indirecte, ces méthodes sont limitées à la résolution des problèmes minmax et des problèmes relatifs à la norme l_1. Ces méthodes ne seront donc pas étudiées.

2.1 Réduction à un problème différentiable

2.1.1 Lissage de la fonction

Considérons le problème l_1 :

$$\min_x \sum_{i=1}^{m} |f_i(x)|, \tag{2.1}$$

composé de fonctions f_i que nous supposerons différentiables. La fonction $f(x) = \sum_{i=1}^{m} |f_i(x)|$ est non différentiable lorsque pour un x la valeur d'une fonction $f_i(x)$ devient nulle. El Attar et al. [2] ont alors proposé de perturber le problème de la manière suivante :

$$\min_x \sum_{i=1}^{m} (f_i(x)^2 + \varepsilon)^{1/2},$$

où $\varepsilon > 0$.

On vérifie aisément que la fonction perturbée $f_\varepsilon(x) = \sum_{i=1}^{m}(f_i(x)^2 + \varepsilon)^{1/2}$ est différentiable. Elle peut donc être minimisée par une méthode classique. En théorie, lorsque ε tend vers 0 la solution du problème perturbé converge vers la solution du problème original. Dans la pratique des difficultés apparaissent. Comme $f_i(x)^2 + \varepsilon$ est pratiquement égal à zéro chaque fois que $f_i(x) = 0$, il faut calculer la racine-carrée d'un terme quasiment égal à zéro ce qui conduit à des instabilités numériques. De plus, le choix de ε est délicat. Certaines valeurs, excellentes pour une fonction f, peuvent être désastreuses pour une autre.

Une transformation similaire peut être trouvée pour les problèmes minmax :

$$\min_x \max_{1\le i\le m} f_i(x). \tag{2.2}$$

2.1.2 Transformation des problèmes l_1 et l_∞

Une autre transformation possible est de modifier le problème pour qu'il devienne un problème de minimisation sous contraintes. Ceci se fait par l'adjonction de nouvelles variables α_i. Les problèmes (2.1) et (2.2) se récrivent alors :

$$\begin{aligned} &\text{Minimiser} \quad \textstyle\sum_i \alpha_i \\ &\qquad \text{s.c.} \quad -\alpha_i \le f_i(x) \le \alpha_i, \qquad\qquad i = 1, 2, \ldots, m, \end{aligned}$$

et

$$\begin{aligned} &\text{Minimiser} \quad \alpha \\ &\qquad \text{s.c.} \quad f_i(x) \le \alpha, \qquad\qquad i = 1, 2, \ldots, m. \end{aligned}$$

L'équivalence de ces problèmes peut être aisément vérifiée. Ces nouveaux problèmes sont alors différentiables et peuvent donc être résolus par les méthodes classiques. Le prix à payer est un plus grand nombre de variables et le passage d'un problème d'optimisation sans contrainte à un problème d'optimisation avec contraintes (voir §1.1.4). Bien que peu esthétique cette méthode fonctionne parfois très bien en pratique.

2.2 Méthode des sous-gradients

2.2.1 Principes généraux

Cette approche directe est une généralisation de la méthode du gradient vue pour l'optimisation différentiable. Supposons tout d'abord que f est C^1 et que x_k est le point courant. Dans les méthodes faisant une approximation du premier ordre de la fonction f, à chaque itération, on fait un pas positif dans la direction opposée du gradient.

$$x_{k+1} := x_k - t_k \nabla f(x_k), \qquad\qquad t_k \in \mathbb{R}^{+*}.$$

La direction $-\nabla f(x_k)$ est une direction de descente. Donc, une recherche linéaire (unidimensionnelle) suivant cette direction va permettre de déterminer un t_k de façon à ce que $f(x_{k+1}) < f(x_k)$.

Dans le cas où f est non différentiable, il n'est pas possible de définir son gradient en un point. Cependant, l'hypothèse (1.1) indique que l'on connait au moins 1 sous-gradient en x_k. La méthode devient alors :

$$x_{k+1} := x_k - t_k \frac{g_k}{\| g_k \|}, \qquad \text{où} \quad g_k \in \partial f(x_k) \quad \text{et} \quad t_k \in \mathbb{R}^{+*}. \tag{2.3}$$

Deux problèmes restent à résoudre :

1. Le sous-gradient g_k n'étant pas nécessairement une direction de descente, comment choisir la valeur de t_k ?

2. Le sous-gradient g_k n'étant pas nécessairement nul à l'optimum, quel test d'arrêt doit être utilisé ?

Commençons par la deuxième question. Dans le cas différentiable nous avons vu que, le gradient devenant petit lorsque l'on s'approche de l'optimum, cela pouvait fournir un bon test d'arrêt. Malheureusement, ce n'est plus le cas pour les fonctions non différentiables. Rappelons l'exemple :

$$f(x) = |x| \Longleftrightarrow f(x) = \max\{x, -x\}.$$

Nous avons les sous-gradients -1 et $+1$ pour $x_k \neq 0$. Et ceci que l'on soit proche ou non de l'optimum. De plus, supposons que, par chance, on se trouve à l'optimum (c.à.d. $x_k = 0$ et $\partial f(x_k) =$ Enveloppe Convexe$\{-1, +1\}$), le fait que l'on ne possède pas toute l'information sur f ne nous permet pas de réaliser que l'on se trouve à l'optimum. On va alors faire un déplacement suivant l'un ou l'autre des sous-gradients ce qui va nous éloigner de l'optimum. Malheureusement la littérature reste très laconique sur les critères d'arrêt qui peuvent être utilisés. Cet aspect est un des handicaps principaux de cette méthode.

En ce qui concerne la valeur de t_k, considérons à nouveau la fonction (1.11) représentée à la figure 1.1. Au point $x_k = (0,0)^T$ elle est non différentiable et $g_k = (9,16)^T$ est un de ses sous-gradients en ce point. Il apparait clairement, en observant la figure, que $-g_k$ n'est pas une direction de descente. Il n'existe donc pas de $t_k > 0$ tel que $f(x_{k+1}) < f(x_k)$ pour ce sous-gradient g_k. Contrairement au cas différentiable le pas t_k ne peut pas être déterminé à l'aide d'une recherche unidimensionnelle, même si la direction est une direction de descente (voir pour cela les itérations effectuées sur la figure 1.1). Le cas non-différentiable nécessite donc que t_k soit choisi de manière externe. Lemaréchal [13] prouve alors le lemme suivant :

Lemme : 2.1 *Soit x_k le point courant non optimal et x^* un point optimal. Il existe $T_k = 2(f(x^k) - f(x^*))/\|g_k\| > 0$ tel que pour l'itération (2.3) :*

$$\| x_{k+1} - x^* \| \;<\; \| x_k - x^* \| \qquad\qquad \forall t_k \in [0, T^k[.$$

L'interprétation géométrique de ce lemme est la suivante (voir figure 2.1). Soit $|\beta|$ l'angle entre $-g_k$ et $x^* - x_k$. Le lemme dit que le nouveau point déterminé par (2.3) peut être plus près de l'optimum que x^k. Cet angle est donc inférieur à $\pi/2$. De plus il existe une valeur T_k (valeur maximum que peut prendre t_k) qui assure que l'on se rapproche effectivement de la solution.

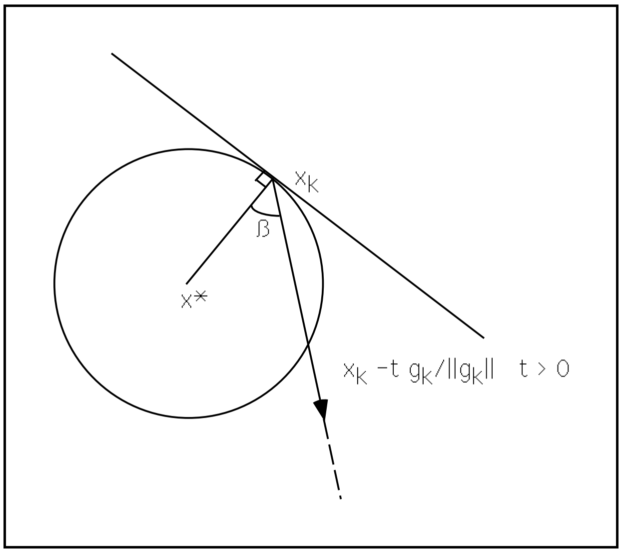

Figure 2.1: Interprétation géométrique du lemme.

Malheureusement, dans la plupart des cas la valeur de l'optimum n'est pas connue et, par conséquent, la valeur de T_k non plus. Nous savons seulement que la longueur du pas t_k doit être suffisamment petite pour que l'inégalité soit satisfaite. Ce qui conduit à choisir t_k parmi une séquence d'éléments convergents vers 0.

Une autre propriété doit être considérée. Soit x_0 le point de départ et soit $\Lambda = \sum_{j=0}^{\infty} t_j$. A chaque itération :

$$\begin{aligned} \| x_0 - x_k \| &\leq \| x_0 - x_1 \| + \| x_1 - x_2 \| + \ldots + \| x_{k-1} - x_k \| \\ &= t_0 + t_1 + \ldots + t_{k-1} \;\leq\; \Lambda. \end{aligned}$$

En d'autres termes, on reste durant toutes les itérations dans une boule de rayon Λ centrée en x_0. Ce qui suggère de choisir t_k petit mais tel que la somme Λ soit grande. Ainsi, même si le point de départ x_0 se situe loin de l'optimum x^* ce dernier sera atteint. Le calcul du nouveau point, à l'itération k, s'effectue alors de la façon suivante :

$$x_{k+1} := x_k - t_k \frac{g_k}{\|g_k\|} \quad \text{où} \quad g_k \in \partial f(x_k) \text{ et} \tag{2.4}$$
$$t_k \text{ tel que } \lim_{k\to\infty} t_k = 0 \text{ et } \sum_{j=0}^{\infty} t_j = \infty.$$

Lorsque l'on applique cet algorithme, il est alors possible de démontrer le théorème suivant (voir Ermoliev [4] et Polyak [18]) :

Théorème : 2.1 *Soit X^* l'ensemble des points optimaux que l'on suppose non vide et borné. Pour un point de départ x_0 quelconque, la séquence x_k générée par (2.4) converge vers un point $x^* \in X^*$*

Ce type d'algorithme possède l'avantage d'être très simple à mettre en œuvre. Aucune recherche unidimensionnelle ne doit être faite. En contrepartie, il se peut que la convergence soit très lente. Nous avons en effet le théorème suivant [13] :

Théorème : 2.2 *Le taux de convergence de l'algorithme dont les itérations sont définies par (2.4) n'est pas géométrique.*

Preuve. Supposons pour l'instant que le taux de convergence soit géométrique. Il existe alors M et $0 < q < 1$ tel que :

$$\| x_k - x^* \| \leq Mq^k, \qquad \forall k.$$

Alors, pour tout k nous avons l'inégalité suivante :

$$t_k = \| x_{k+1} - x_k \| \leq \| x_{k+1} - x^* \| + \| x^* - x_k \| \leq M(q+1)q^k.$$

En faisant la somme sur k on obtient :

$$\sum_{k=0}^{\infty} t_k \leq M(q+1)\sum_{k=0}^{\infty} q^k = M\frac{q+1}{1-q}.$$

Ce qui contredit le fait que cette somme est égale à l'infini. Le taux de convergence n'est donc pas géométrique. ∎

La figure 2.2 illustre le cas où, partant du même point de départ que dans la figure 1.1, on cherche à minimiser la fonction (1.11) en choisissant $\frac{2.175}{k+1}$ comme valeur pour le pas t_k. Les conditions $\lim_{k\to\infty} t_k = 0$ et $\sum_{k=0}^{\infty} t_k = \infty$ sont satisfaites. Le théorème (2.1) certifie que la séquence des x_k définie par (2.4) converge vers l'unique optimum $x^*(-1,0)$. Cependant, la vitesse de convergence est extrêmement lente. Apres deux itérations, on se trouve relativement proche de l'optimum, mais ensuite les points générés oscillent autour. Au bout de 80 itérations on obtient seulement une précision de de 10^{-1} [25].

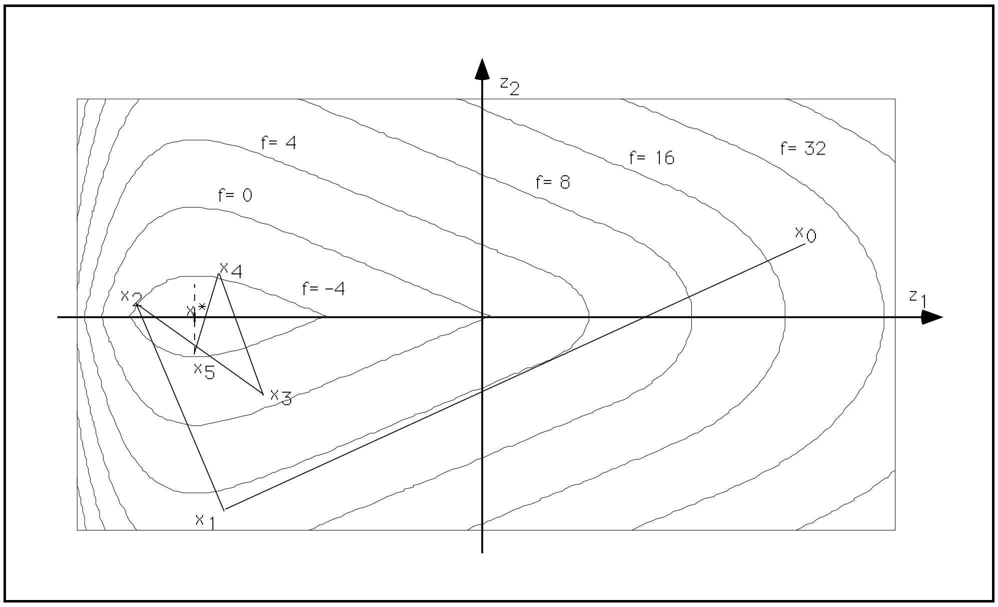

Figure 2.2: Contour de f et itérations avec la méthode des sous-gradients.

Pour pallier ce type de problème, différents types de calcul du pas t_k ont été envisagés. En voici les principaux.

2.2.2 Méthode de la série convergente

Pour que la convergence soit géométrique, il est nécessaire que la série $\{t_k\}$ soit géométrique. Shor [20] et Goffin [6] ont proposé la série :

$$t_k = M\rho^k, \qquad \text{avec } M > 0 \text{ et } 0 < \rho < 1.$$

Comme $\Lambda = \sum_{k=0}^{\infty} t_k = M/(1-\rho)$ n'est plus infini, le théorème (2.1) ne peut plus être appliqué. L'algorithme peut donc converger vers un point qui n'est pas optimal. Cependant, toutes les itérations demeurant dans une boule centrée en x_0 et de rayon

Λ, si on prend la précaution de s'assurer que l'optimum soit contenu dans cette boule, l'algorithme converge géométriquement vers celui-ci.

Cette algorithme présente non seulement l'avantage de converger géométriquement vers un point $\tilde{x}$ mais de plus, il ne nécessite pas de connaître la valeur $f(x^*)$ de l'optimum. Malheureusement, pour s'assurer que $\tilde{x}$ soit optimal, il est souvent nécessaire que M soit très grand et ρ proche de l'unité. La convergence peut donc être très lente.

2.2.3 Méthode de relaxation

Si la valeur de l'optimum $f(x^*)$ est connue, il est possible de prendre :

$$t_k = \lambda \frac{f(x) - f(x^*)}{\| g_k \|}, \qquad \text{avec } 0 < \lambda < 2. \tag{2.5}$$

Le lemme (2.1) assure la convergence de l'algorithme. De plus, si l'optimum est unique Polyak [19] a montré que la convergence est géométrique. Dans tous les cas le lemme (2.1) garantit une décroissance monotone de $\| x_k - x^* \|$.

Le problème de réalisabilité est précisément un cas où l'optimum est connu ($f(x^*) = 0$) et donc où ce calcul de t_k peut être fait.

Comme la valeur de l'optimum n'est pas toujours connue, Agmon [1], Motzkin et Schoenberg [15] puis Held, Wolfe et Crowder [10] ont proposé des variantes où $f(x^*)$ est approximé par $\tilde{f}$ dans (2.5). Si $\tilde{f} > f(x^*)$ on peut montrer [14] que l'algorithme converge linéairement vers un point $\tilde{x}$ tel que $f(\tilde{x}) \leq \tilde{f}$. Par contre, si $\tilde{f} < f(x^*)$ l'algorithme ne converge généralement pas vers f^*. Held, Wolfe et Crowder [10] ont donné des heuristiques permettant de choisir λ dans (2.5). Cependant, malgré des résultats satisfaisants obtenus dans la pratique, ces heuristiques n'ont pas pu être justifiées sur le plan théorique.

2.2.4 Accélérations de la méthode des sous-gradients

Si la valeur de l'optimum $f(x^*)$ n'est pas connue et que l'on veut être certain de trouver l'optimum, il existe un moyen d'accélérer la convergence de la méthode des sous-gradients. On utilise pour cela l'information des itérations antérieures. Il existe plusieurs approches. Nous ne détaillerons que la principale, due à Shor [21], qui consiste, à chaque itération, à faire une dilatation dans l'espace des sous-gradients.

Supposons que l'algorithme trouve le point x_1 en se déplaçant de x_0 suivant la direction $-g_0$. Soit $-g_1$ la direction de déplacement en x_1, pour éliminer la nature oscillatoire des itérations, une possibilité consiste à supprimer la partie de $-g_1$ collinéaire à $-g_0$. Cette dilatation de l'espace est obtenue en multipliant par une matrice H_1 le vecteur g_1. Si l'on cumule les dilatations à chaque itération dans une matrice H_k on obtient les formules suivantes :

$$\begin{aligned} d_k &= \frac{H_k g_k}{\sqrt{g_k^T H_k g_k}}, \qquad \text{où} \quad g_k \in \partial f(x_k), \\ H_{k+1} &:= \alpha_k (H_k - \beta_k d_k d_k^T), \\ x_{k+1} &:= x_k - t_k d_k. \end{aligned} \tag{2.6}$$

où α, α_k, β_k et t_k sont des paramètres positifs et où la matrice H_0 est définie positive, par exemple $H_0 = \alpha I$.

On remarquera :

- Si $\alpha_k = 1$ et $\beta_k = 0$ on retrouve la méthode des sous-gradients classique.
- Le cas $\beta = 1$ correspond à une projection : g_k appartient à l'espace nul de H_{k+1}.
- Si la fonction est continûment différentiable, cette méthode peut être considérée comme une variante de la méthode des gradients conjugués décrite au chapitre précédent.
- Cette méthode n'est autre qu'une variante de la méthode des ellipsoïdes que nous verrons en détail plus loin.

Si n est la dimension du problème, il paraît intéressant de donner aux paramètres les valeurs suivantes [13] :

$$\alpha_k = \frac{n^2}{n^2-1}, \quad \beta_k = \frac{2}{n+1} \quad \text{et} \quad t_k = \frac{1}{n+1}.$$

Shor [22], Goffin [7] et Nemirovskii-Yudin [17] ont alors montré le résultat de convergence suivant :

Théorème : 2.3 *Si x_0 et H_0 sont tels que $\| x_o - x^* \|^2 \leq \alpha$, et $H_o = xI$, alors il existe $M > 0$ et $q < 1$ tel que la séquence de points x_k générée par (2.6) satisfait :*

$$\min_{0 \leq j \leq k} \{f(x_j) - f(x^*)\} \leq Mq^k, \qquad \forall k.$$

Ce théorème garantit qu'une sous-suite de points x_k converge géométriquement vers l'optimum x^*. Une analyse plus précise montre que $q = [n^2/(n^2-1)]^{1/2}[1-2/(n+1)]^{1/2n} \approx 1 - 1/2n^2$. Le paramètre q est donc malheureusement d'autant plus proche de 1 que la dimension n du problème est grande.

2.2.5 Généralisation à l'optimisation sous contraintes

Les méthodes de sous-gradients se généralisent sans difficulté au problème d'optimisation sous contraintes (1.6) :

$$\begin{aligned} \text{Minimiser} \quad & f(x) \\ \text{s.c.} \quad & x \in X, \end{aligned}$$

où f est une fonction convexe, X est un sous-ensemble convexe et fermé de $\mathbb{R}^n$ et tel qu'il existe une solution optimale $x^* \in X$.

En notant $P_X x$ la projection de x sur X, la formule itérative des méthodes de sous-gradients (2.3) se récrit :

$$x_{k+1} := P_X\left(x_k - t_k \frac{g_k}{\| g_k \|}\right), \qquad \text{où} \quad g_k \in \partial f(x_k) \quad \text{et} \quad t_k \in \mathbb{R}^{+*}.$$

Le théorème (2.1) du à Ermoliev [4] et Polyak [18] est encore valable.

2.2.6 Conclusions

Nemirovskii [16] montre qu'il est possible de choisir t_k de tel sorte que la méthode des sous-gradients ait une complexité optimale. Il montre en effet que la minimisation par cette méthode d'une fonction convexe, continue au sens de Lipschitz[1], sur un

[1]Une fonction est continue au sens de Lipschitz s'il existe une constante L telle que $\partial f(x) \leq L$ pour tout x sur l'ensemble de définition de f.

ensemble convexe fermé X, possède la propriété suivante :

$$\varepsilon_k = \min_{1\le i\le k}\{f(x^i) - f(x^*)\} \le O(1)L(f)D(X)/\sqrt{N},$$

où $O(1)$ est une constante, $L(f)$ est la constante de Lipschitz de f et $D(X)$ est le diamètre de X.

Ce taux de convergence est optimal puisqu'il montre de même qu'il faut au moins :

$$\min\left\{n; O(1)\left(\frac{L(f)D(X)}{\varepsilon}\right)^2\right\}$$

itérations à tout algorithme pour avoir une précision relative de ε de la valeur de la solution. Dans le cas où la dimension du problème est grande,

$$n \ge \left(\frac{L(f)D(X)}{\varepsilon}\right)^2.$$

Bien qu'en théorie cet algorithme soit optimal, il s'avère très moyen en pratique. Comme nous sommes plus intéressés par le comportement moyen en pratique que par le comportement dans le pire cas, il est souvent préférable d'employer d'autres algorithmes comme, par exemple, la méthode de faisceaux ou l'algorithme ACCPM.

2.3 Méthode des ellipsoïdes

En février 1979, une note de Khachiyan [12] indiqua comment une méthode basée sur des ellipsoïdes pouvait résoudre un programme linéaire en un temps polynomial. Ce résultat, basé sur les travaux de Shor [22] et Nemirovskii-Yudin [17], motiva une recherche importante. Par la suite, cette méthode a été généralisée pour résoudre des problèmes d'optimisation convexe non-différentiable.

L'idée de base de cette méthode consiste à construire une série d'ellipsoïdes $E_0, E_1, \ldots, E_k$ contenant chacun la solution du problème (1.6). A l'itération $k+1$, si le centre x^k de l'ellipsoïde courant E_k n'est pas optimal, un plan séparateur est calculé ($g^T(x - x^k) \le 0$) et l'agorithme détermine l'ellipsoïde de volume minimum contenant ce demi-ellipsoïde :

$$\{x \in E_k : \quad g^T(x - x^k) \le 0\}. \tag{2.7}$$

Par définition, un plan séparateur est un hyperplan qui sépare le point courant x^k de la solution optimale x^*. Comme E_0 contenait la solution du problème (1.6), l'ellipsoïde défini en (2.7) contient donc nécessairement l'optimum. Le nouvel ellipsoïde et son centre sont notés E_{k+1} et x^{k+1}. La démarche est alors poursuivie jusqu'à ce que le centre x_k de l'ellipsoïde courant puisse être considéré comme optimal. La figure 2.3 illustre une itération de l'algotithme.

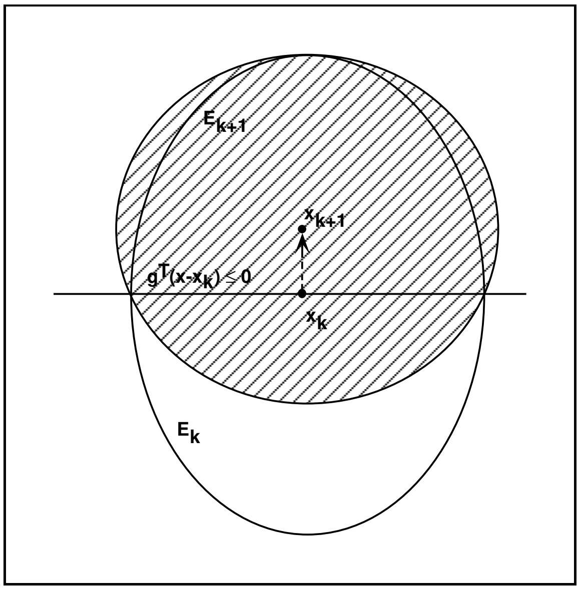

Figure 2.3: Méthode des ellipsoïdes.

De nombreuses possibilités peuvent être envisagées pour choisir cet hyperplan. Par exemple, dans le cas non contraint, on peut le définir par un sous-gradient de f en x^k : $g^T(x - x^k) \leq 0$ avec $g \in \partial f(x^k)$. En utilisant ce type de plan séparateur on fait aisément le parallèle entre la méthode des ellipsoïdes et la méthode des sous-gradients avec dilatation de l'espace. Cependant, pour rester dans un cadre plus général nous continuerons de définir le plan séparateur par $g^T(x - x^k) \leq 0$ sans préciser g.

Lorsque l'on développe les calculs permettant de déterminer l'ellipsoïde de volume minimum on obtient :

$$x^{k+1} := x^k - \tau \frac{H_k g}{(g^T H_k g)^{1/2}}, \tag{2.8}$$

$$H^{k+1} := \delta \left(H_k - \sigma \frac{H_k g g^T H_k}{g^T H_k g} \right), \tag{2.9}$$

où

$$\tau = \frac{1}{n+1}, \quad \delta = \frac{n^2}{n^2 - 1}, \quad \text{et} \quad \sigma = \frac{2}{(n+1)}.$$

L'analogie avec la méthode des sous-gradients est frappante mais la méthode des ellipsoïdes doit cependant être considérée comme une généralisation de cette première.

De nombreuses études ont été faites sur cette méthode (voir [9], [8], [24], [16]). Goffin [8] a montré que si on applique une version de cet algorithme à la minimisation d'une fonction convexe, pour peu que l'ellipsoïde initial contienne une solution, le nombre d'itérations est fini. De plus, le volume de l'ellipsoïde décroît géometriquement d'un ratio

$$c_n = \frac{n}{n+1}(\frac{n^2}{n^2-1})^{(n-1)/2} < e^{-1/2(n+1)},$$

ne dépendant que de la dimension de l'espace.

Nemirovskii [16] donne un algorithme complet dans le cas de l'optimisation non différentiable sous contraintes. Il montre que l'algorithme trouve une solution avec une précision relative de ε en moins de :

$$\bar{k} = \rfloor \frac{\ln(1/\varepsilon)}{(1/\kappa(n))} \lfloor \;\leq\; \rfloor 2n(n-1)\ln(1/\varepsilon) \lfloor$$

itérations avec $\kappa(n) = \frac{n^2}{n^2-1}\sqrt{\frac{n-1}{n+1}} \leq e^{-1/(2(n-1))}$.

La précision est donc relative au volume de l'ellipsoïde initial. Nemirovskii indique donc simplement que pour un ellipsoïde initial E_0 donné, après au plus $\bar{k}$ itérations, son volume $\text{Vol}(E_k)$ vérifie :

$$\text{Vol}(E_k) \leq \varepsilon \text{Vol}(E_0).$$

L'algorithme proposé par Nemirovskii [16] est différent de celui que l'on pourrait construire en utilisant les idées du paragraphe de la méthode des sous-gradients sous contraintes. Dans les deux cas cependant, le test d'arrêt reste un problème délicat. De plus les conclusions pour cet algorithme sont les mêmes que pour la méthode des sous-gradients : bon en théorie, moyen en pratique.

Nous terminons ici l'étude des méthodes "classiques" utilisées pour minimiser les fonctions non différentiables. Dans ce chapitre nous avons omis une autre grande famille de méthodes, celle qui utilise des techniques de plans coupants. Ces méthodes vont faire l'objet du chapitre suivant.

Bibliographie

[1] S. Agmon, "The relaxation method for linear inequalities", *Canadien Journal Math.* **6** (1954) 382-392.

[2] El-Attar, R. Vidyasagar and S.-R. Dutta, "An algorithm for l_1-norm minimization with application to nonlinear l_1-approximation", *SIAM Journal on Numerical Analysis* **16** (1979) 70-86.

[3] A. R. Conn, "An efficient second order method to solve the (constrained) minimax problem", Technical Report CoRR-79-5 University of Waterloo.

[4] Y. M. Ermoliev, "Methods of solution of nonlinear extremal problems", *Cybernetics 2* **4** (1966) 1-17.

[5] R. Fletcher, *Practical methods of optimization II*, John Wiley & Sons, New York (1981).

[6] J. L. Goffin, "On convergence rate of subgradient optimization methods", *Mathematical Programming* **13** (1977) 329-347.

[7] J. L. Goffin, *Convergence results in a class of variable metric subgradient methods*, O.L. Mangasarian, R. R. Meyer and S. M. Robinson (eds.), Nonlinear programming 4 (Academic Press, New York, 1981).

[8] J. L. Goffin, "convergence rates of the ellipsoïd method on general convex functions", *Mathematics of Operations Research* **8**, 1, (1983) 135-150.

[9] D. Goldfarb, M.-J. Todd, "Modification and implementation of the ellipsoid algorithm for linear programming", *Mathematical Programming* **23** (1982) 1-19.

[10] M. Held, P. Wolfe et H. P. Crowder, "Validation of subgradient optimization", *Mathematical Programming* **6** (1974) 62-88.

[11] R. Hornung, "An efficient algorithm for solving the discrete minisum problems", *Operations Research Letters* **1** (1983) 115-118.

[12] L.-G. Khachiyan, "A polynomial algorithm in linear programming", *Doklady Akademiia Nauk SSSR* **224** (1979) 1093-1096 (traduit dans *Soviet Mathematics Doklady* **20** (1979) 191-194).

[13] C. Lemaréchal, "Nondifferentiable optimisation", G. L. Nemhauser, A. H. G. Rinnnooy Kan et M.J. Todd, *Handbooks in operations research and management science : Optimization*, **1** Elsevier Science Publishers B. V. (North-Holland) (1989) 529-572.

[14] M. Minoux, *Programmation mathématique : théorie et algorithmes*, **2** Dunod (Bordas), Paris (1983).

[15] T. Motzkin, I. J. Schoenberg, "The relaxation method for linear inequalities", *Canadien Journal Math.* **6** (1954) 393-404.

[16] A. Nemirovskii *unpublished Lecture notes* Technion,The Israel Institute of Technology, Faculty of Industrial Engineering & Management (1994).

[17] A. Nemirovskii, D.B Yudin, *Problem complexity and method efficiency in optimization*, Wiley-Interscience, 1983.

[18] B. T. Polyak, "A general method of solving extremum problems", *Soviet Mathematics* **8** (1966) 593-597.

[19] B. T. Polyak, "Minimization of unsmooth functionals", *USSR Computational Mathematics and Mathematical Physics* **9** 14-29.

[20] N. Z. Shor, "On the rate of convergence of the generalized gradient descent method", *Cybernetics 4* **3** (1968) 79-80.

[21] N. Z. Shor, "Convergence rate of the gradient descent method method with dilatation of the space", *Cybernetics 6* **2** (1970) 102-108.

[22] N. Z. Shor, "Cut-off method with space extension in convex programming problems", *Cybernetics 13* **1** (1977) 94-96.

[23] R. S. Womersley, *Numerical methods for structured problems in nonsmooth optimization*, University of Dundee (1981).

[24] D.-B. Yudin, A.-S. Nemirovskii, "Informational complexity and effective methodes of solution for convex external problems", *Ekonomika i Matematicheskie Metody* **12** (1976) 357-369 (traduit dans *Matekon* (Spring 1977) (M.E. Sharpe, Inc., White Plains, N.Y.)).

[25] J. Zowe, *Nondifferentiable optmization*, NATO ASI Series F15, Computational Mathematical Programming, ed. K. Schittkowski, Springer-Verlag Berlin Heidelberg (1985) 323-356.

Chapitre 3

Méthodes de plans coupants

Au chapitre précédent, nous avons vu quelques méthodes permettant de résoudre les problèmes d'optimisation non partout différentiables. Certaines d'entres elles (méthode des sous-gradients) sont des méthodes optimales dans le sens que l'on peut montrer qu'il n'est pas possible de trouver un algorithme ayant un meilleur taux de convergence. Dans la pratique, ces méthodes s'avèrent assez moyennes comparées à d'autres appelées méthodes de plans coupants que nous allons voir plus en détail maintenant.

Après avoir donné les principes de base des méthodes de plans coupants, nous allons détailler dans ce chapitre les plus classiques d'entre elles. Nous commencerons tous d'abord par la méthode de Kelley, puis nous verrons quelques variantes des méthodes de faisceaux et nous terminerons par les méthodes des centres (méthode d'Elzinga et Moore, méthode du centre de gravité et méthode volumétrique). l'algorithme ACCPM que nous décrirons dans les chapitres suivants fait partie de la famille des méthodes des centres.

3.1 Principe

Les algorithmes de plans coupants pour la programmation convexe sont relativement simples dans leur principe. Ils procèdent par amélioration successive de l'approximation linéaire du problème. Les différentes méthodes ne diffèrent que par la manière dont l'amélioration de l'approximation est construite à chaque itération.

Le problème original est le suivant :

$$\begin{aligned} \text{Minimiser} \quad & c^T x \\ \text{s.c.} \quad & x \in X, \end{aligned} \tag{3.1}$$

où $X \subset \mathbb{R}^n$ est un ensemble convexe compact. Il est à noter que le problème (1.6) de minimisation d'une fonction convexe sur un ensemble convexe défini comme :

$$\begin{aligned} \text{Minimiser} \quad & f(y) \\ \text{s.c.} \quad & y \in Y, \end{aligned} \tag{3.2}$$

où $Y \subset \mathbb{R}^{n-1}$ est un ensemble convexe et f une fonction convexe, peut être reformulé comme le problème (3.1). En effet, le problème (3.2) est équivalent à :

$$\begin{aligned} \text{Minimiser} \quad & r \\ \text{s.c.} \quad & f(y) - r \le 0 \\ & y \in Y. \end{aligned} \tag{3.3}$$

En prenant $x = (r, y)$ on retrouve ainsi le problème (3.1).

Nous sommes donc bien dans le cadre général de la minimisation d'une fonction convexe sur un ensemble convexe. Soit P_0 une approximation linéaire initiale polyédrique de X et x^0 un point de P_0, la formulation générale d'un algorithme de plans coupants pour résoudre le problème (3.1) est :

Méthode de plans coupants

Initialisation :

k := 0 ;
Définir $P_0 \supset X$;
Choisir $x^0 \in P_0$;

Tant que x^k n'est pas optimal, **faire**

Si $x \notin X$, **alors** définir un hyperplan H_k séparant le point x^k de X ;
Sinon $H_k = \{x : c^T x = c^T x^k\}$ **fsi** ;
Mettre à jour P_k en ajoutant H_k (c.à.d. $P_{k+1} := P_k \cap \{a_k^T x \le b_k\}$) ;
Sélectionner un point x^{k+1} appartenant à P_{k+1} ;
k := k+1 ;

Fin.

Définir un hyperplan séparant x^k de X consiste à trouver $a_k \in \mathbb{R}^n$, $b_k \in \mathbb{R}$ tel que $X \subset \{x : a_k^T x \leq b_k\}$ et $x^k \in \{x : a_k^T x > b_k\}$. Cette procédure itérative affine à chaque itération la connaissance qu'elle a du problème en ajoutant au modèle une forme linéaire.

La figure 3.1 illustre ce processus lorsque le point x^{k+1} minimise la fonction $c^T x$ sur l'ensemble P_{k+1}. On remarquera que dans ce cas le test d'arrêt est :

Tant que $x^k \notin X$ **faire**.

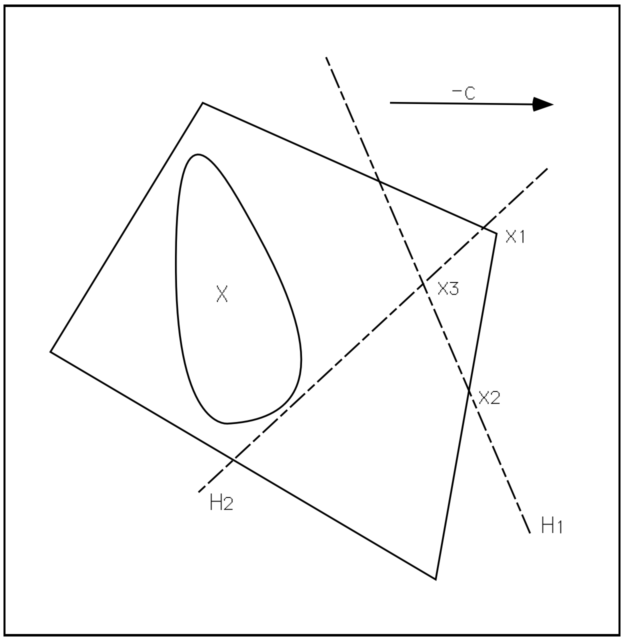

Figure 3.1: Méthode de plans coupants.

Les algorithmes de plans coupants diffèrent principalement dans la manière dont est sélectionné le nouveau point x^{k+1} dans P_{k+1}. C'est l'aspect le plus important. En effet, plus la coupe est profonde, c'est-à-dire plus le volume de P_{k+1} est petit par rapport à P_k, plus l'algorithme converge rapidement.

Il est à noter que dans certaines versions des méthodes de réduction sont implantées. Elles ont pour objectif d'éliminer les hyperplans qui n'interviendront plus dans l'élaboration de la solution. Elles présentent l'avantage de réduire la taille du modèle ce qui limite les calculs.

Maintenant que les idées de base ont été présentées, étudions quelques méthodes classiques de plans coupants.

3.2 Algorithme de Kelley

3.2.1 Généralités

L'algorithme proposé par Kelley [2] en 1960, a été originellement développé pour résoudre des problèmes de programmation convexe de la forme :

$$\begin{aligned} &\text{Minimiser} \quad f(x) \\ &\text{s.c.} \quad g_i(x) \leq 0, \qquad i = 1, 2, \ldots, p, \end{aligned} \tag{3.4}$$

où $x \in \mathbb{R}^n$ et f et g_i sont des fonctions convexes et différentiables. Cependant, il peut être aisément généralisé au cas convexe non différentiable. Comme nous avons vu précédemment, ce problème est équivalent au problème suivant, dont l'objectif est linéaire :

$$\begin{aligned} &\text{Minimiser} \quad c^T x \\ &\text{s.c.} \quad g(x) \leq 0, \end{aligned} \tag{3.5}$$

où $x \in \mathbb{R}^n$ et où $g(x) \in \mathbb{R}^{p+1}$ est convexe et différentiable. Comme g est convexe et différentiable nous avons l'inégalité fondamentale :

$$g(x) \geq g(x^k) + \nabla^T g(x^k)(x - x^k), \tag{3.6}$$

pour tout x et $x^k \in \mathbb{R}^n$. Cette inégalité est utilisée pour déterminer l'hyperplan de séparation. L'algorithme est le suivant :

Méthode de Kelley

Initialisation :

k := 0 ;

Définir $P_0 \supset X = \{x : g(x) \leq 0\}$;

Choisir $x_0 := \text{argmin}\{c^T x : x \in P_0\}$;

Tant que $g(x^k) > 0$, **faire**

Soit i l'indice maximisant $g_i(x^k)$. Par définition $g_i(x^k) > 0$. **Former** P_{k+1} en faisant l'intersection de P_k et du demi-espace défini par :

$$\{x : g_i(x^k) + \nabla^T g_i(x^k)(x - x^k) \leq 0\}; \tag{3.7}$$

$x^{k+1} := \text{argmin}\{c^T x : x \in P_{k+1}\}$;

k := k+1 ;

Fin.

L'équation $g_i(x^k) + \nabla^T g_i(x^k)(x - x^k) = 0$ définit bien un hyperplan de séparation. En effet, si $\nabla g_i(x^k) \neq 0$ on a pour tout $x \in X$:

$$0 \geq g_i(x) \geq g_i(x^k) + \nabla^T g_i(x^k)(x - x^k),$$

d'où $\{x : g_i(x^k) + \nabla^T g_i(x^k)(x - x^k) \leq 0\} \supset X^*$. Par ailleurs, le demi-espace ne contient pas x^k puisque $g(x^k) > 0$.

Le cas $\nabla g_i = 0$ implique que g_i est minimal en x^k. Par suite, $g(x) \geq g(x^k) > 0$ et donc $X = \emptyset$. Le problème n'admet alors pas de solution.

Une illustration de la méthode est donnée figure 3.2 dans le cas où l'on se trouve dans l'espace à une dimension ($x \in \mathbb{R}$).

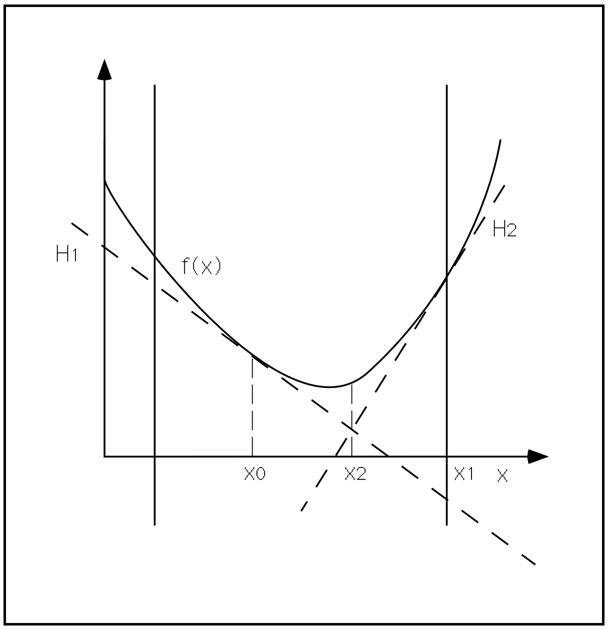

Figure 3.2: Méthode de Kelley.

La méthode de Kelley est donc une méthode de plans coupants pour laquelle les plans de séparation sont générés au minimum de la relaxation linéaire courante.

3.2.2 Convergence

Sous certaines conditions sur les fonctions convexes, la méthode de Kelley est globalement convergente. Il est possible d'appliquer le théorème de la convergence générale pour le prouver [22] mais la preuve suivante est plus directe et aussi plus simple.

Théorème : 3.1 [13] *Soit les fonctions convexes $g_i, i = 1, 2, \ldots, p$, continûment différentiables et $\{x^k\}$ la séquence des points générés par l'algorithme. Toute limite de cette séquence est une solution du problème (3.5).*

Preuve. Définissons $\{x^k\}$, $k \in \mathcal{K}$ une sous-séquence de $\{x^k\}$ convergent vers $x^{\bar{k}}$. En prenant, si nécessaire, une sous-séquence de $\mathcal{K}$, il est possible de supposer que l'indice i maximisant $g_i(x^k)$ au point 1 de l'algorithme est toujours le même. Si $k \in \mathcal{K}$ et $\tilde{k} \in \mathcal{K}$ sont tels que $\tilde{k} > k$, nous devons avoir :

$$g_i(x^k) + \nabla^T g_i(x^k)(x^{\tilde{k}} - x^k) \leq 0,$$

puisque les points générés satisfont les contraintes antérieures. Ceci implique alors que :

$$g_i(x^k) \leq \|\nabla^T g_i(x^k)\| \quad \|x^{\tilde{k}} - x^k\|. \tag{3.8}$$

Si l'on suppose que la fonction g_i est continue au sens de Lipschitz alors $\|\nabla^T g_i(x^k)\|$ est bornée pour tout $k \in \mathcal{K}$. Le membre de droite de (3.8) tend donc vers zéro quand k et $\tilde{k}$ tendent vers l'infini alors que le membre de gauche tend vers $g_i(x^{\bar{k}})$ par continuité. On obtient alors $g_i(x^{\bar{k}}) \leq 0$ ce qui prouve que $x^{\bar{k}}$ est réalisable pour le problème (3.5).

D'autre part, si f^* est la valeur de la solution optimale du problème (3.5) alors $c^T x^k \leq f^*$ pour tout k puisque x^k est obtenu en minimisant $c^T x^k$ sur l'ensemble $P_k \supset X$. Par continuité, $c^T x^{\bar{k}} \leq f^*$ et donc $x^{\bar{k}}$ est une solution optimale. ∎

Nemirovskii [16] indique que, dans le pire cas, pour résoudre le problème (3.5) avec une précision relative de ε, le nombre d'itérations nécessaires est au moins :

$$O(1)(1/\varepsilon)^{(n-1)/2}.$$

La convergence théorique de cet algorithme est donc désastreuse. Nous sommes très loin de la convergence polynomiale convoitée. En effet, avec $\varepsilon = 2^{-L}$ où L est relatif à la dimension des données du problème, l'algorithme converge en $O(2^{L(n-1)/2})$ itérations.

3.2.3 Accélération de la méthode de Kelley

Dans une variante de l'algorithme de Kelley les plans coupants sont généres sur la frontière de l'ensemble X. Les restrictions sur la convexité sont alors moins importantes. Il n'est pas nécessaire que les fonctions g_i soient convexes. Par contre, l'ensemble X doit l'être. De plus, on doit faire les hypothèses suivantes :

1. On connait un point $\hat{x}$ à l'intérieur de X.

2. Pour tout point x non optimum sur la frontière de X, $\max_i \nabla g_i(x) \neq 0$.

L'algorithme est le même que celui de Kelley exepté qu'à chaque itération le plan de support n'est pas généré au minimum de la relaxation continue (c.à.d. en $x^{k+1} := \text{argmin}\{c^T x : x \in P_{k+1}\}$) mais en un point $\tilde{x}$ se trouvant à l'intersection de la demi-droite $[\hat{x}, x_{k+1}]$ et de la frontière de l'ensemble X. La figure 3.3 illustre deux itérations de cet algorithme lorsque $x \in \mathbb{R}^2$.

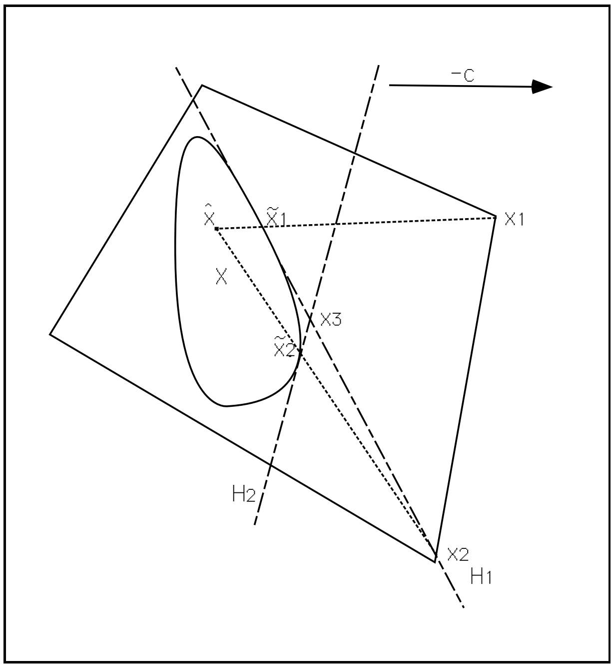

Figure 3.3: Méthode des plans de supports.

Le prix à payer par rapport à l'algorithme de Kelley est une interpolation linéaire le long de la demi-droite $[\hat{x}, x_{k+1}]$ pour trouver le point $\tilde{x}$. Si cette interpolation n'est pas trop coûteuse, cette méthode offre alors l'intérêt d'introduire un élement régulateur $\hat{x}$ qui va probablement atténuer les effets oscillatoires caractéristiques de l'algorithme de Kelley.

3.2.4 Conclusion sur la méthode de Kelley

La méthode de Kelley dans sa version standard est donc la méthode la plus intuitive que l'on puisse imaginer. Elle consiste à construire une approximation linéaire du problème et en calcule le minimum en faisant l'hypothèse que celle-ci dépeint correctement le problème. Si l'optimum n'est pas atteint, elle enrichit en ce point son modèle en y introduisant une nouvelle forme linéaire et poursuit sa recherche.

Malheureusement, son taux de convergence est désastreux. En pratique, la méthode fonctionne mieux que ne le laisse prévoir ce résultat. Surtout si la fonction à minimiser est linéaire par morceaux. Il n'en demeure pas moins que cette méthode reste trop lente pour beaucoup de problèmes.

Une façon naturelle d'améliorer la méthode de Kelley est la suivante : tout ce que l'on peut espérer c'est que l'approximation linéaire dépeint correctement la fonction aux alentours des points où l'approximation linéaire a été faite (c.à.d. aux alentours des points x_k). Il parait donc raisonnable de forcer le point suivant x_{k+1} à rester proche du précédent. Plus précisément du meilleur des points précédents. C'est dans cet optique qu'ont été développées les méthodes de faisceaux.

3.3 Méthodes de faisceaux

Les méthodes de faisceaux ont été initialement introduites par Lemaréchal [8] [9] et Wolfe [21] dans le cas convexe non contraint et par Mifflin [14] pour le cas non convexe avec contraintes. Ces études ont été complétées et répertoriées par Kiwiel en 1985 [3] puis en 1989 [4] [5] et en 1994 [7].

Comme la méthode de Kelley, elles construisent une approximation linéaire (polyédrique) du problème. Cependant, elles diffèrent par rapport à cette première dans la définition du point x_{k+1} où le plan de séparation est généré. Pour déterminer ce point, les méthodes de faisceaux cherchent le point qui minimise un programme quadratique tandis que la méthode de Kelley cherche le point qui minimise l'approximation linéaire courante.

Nous allons voir maintenant plusieurs variantes. Nous verrons tout d'abord le cas simple de la minimisation d'une fonction convexe non contrainte, puis nous généraliserons l'algorithme au cas contraint en donnant deux de ces variantes.

3.3.1 Fonction convexe non contrainte

Commençons par le cas simple de la minimisation d'une fonction convexe f non contrainte. Pour résoudre ce problème, la méthode de faisceaux génère une séquence de points $\{x^k\}$ convergeant vers l'optimum de f et une séquence de points $\{y^k\}$ auxquels sont générés les plans de support de f :

$$\bar{f}^k(x) = f(y^k) + (x - y^k)^T \nabla f(y^k), \qquad \forall x \in \mathbb{R}^n. \tag{3.9}$$

La démarche de cette méthode se rapproche de celle d'une personne qui, marchant dans le noir, sonde d'abord l'endroit où elle va poser son prochain pas (point y^{k+1}) et qui, en fonction de l'information, décide ou non de le faire (c.à.d. $x^{k+1} := y^{k+1}$ ou $x^{k+1} := x^k$). L'algorithme démarre avec $x^0 = y^0$ un point de départ. A l'itération k, l'approximation polyédrique de f est donnée par :

$$\hat{f}^k(x) = \max\{\bar{f}^j(x) : j \in K\}, \qquad \forall x \in \mathbb{R}^n, \tag{3.10}$$

où $K = \{0, 1, \ldots, k\}$.

Par définition, nous avons $\hat{f}^k(x) \leq f(x)$ avec plus particulièrement l'égalité : $\hat{f}^k(y^j) = f(y^j) \quad \forall j \in K$. Le point y^{k+1} suivant est généré comme le minimum de la fonction suivante :

$$y^{k+1} := \operatorname{argmin}\left\{\hat{f}^k(x) + \frac{u^k\|x - x^k\|^2}{2} : x \in \mathbb{R}^n\right\}, \tag{3.11}$$

où $u^k > 0$ est choisi de sorte que y^{k+1} reste dans une région où l'approximation $\hat{f}^k(x)$ est proche de f. Ensuite, si

$$f(y^{k+1}) \leq f(x^k) + \mu v^k, \qquad \mu \in [0; 0,5], \tag{3.12}$$

où $v^k = \hat{f}^k(y^{k+1}) - f(x^k)$, alors $x^{k+1} := y^{k+1}$. Appelons ceci un "pas réel". Cela revient à considérer que la fonction objectif a suffisamment décru pour mémoriser l'information. Sinon, c'est-à-dire si l'égalité (3.12) n'est pas vérifiée, on prend $x^{k+1} := x^k$. Appelons ceci un "pas nul". Il est à noter cependant que si le pas est nul, l'itération n'a pas pour autant été inutile. Elle a permis en effet d'introduire un nouveau plan de support en y^{k+1} et ainsi, a enrichi le modèle du problème original. L'itération suivante cherchera alors le minimum d'une fonction dans laquelle seule la relaxation de f aura évolué, c'est-à-dire où seule la connaissance sur f se sera enrichie, le point de référence x^k restant le même.

De façon plus précise, en définissant $d^k = y^{k+1} - x^k$, le vecteur d^k et le scalaire v^k doivent minimiser le problème quadratique suivant[1] :

$$\begin{aligned} \text{Minimiser} \quad & \tfrac{1}{2}u\|d\|^2 + v \\ \text{s.c.} \quad & -\alpha^j + d^T\nabla f(y^j) \leq v, \qquad \forall j \in K \\ & (d, v) \in \mathbb{R}^{n+1}. \end{aligned} \tag{3.13}$$

où $\alpha^j = f(x^k) - \bar{f}^j(x^k)$ est l'erreur due à la linéarisation de f en x^k par rapport à la $j^{\text{ème}}$ forme linéaire.

En résolvant ce problème, on va chercher à minimiser l'approximation linéaire de la fonction f en restant dans le voisinage du meilleur point trouvé. Le concept développé par Kelley est donc amélioré. Ici on ne se contente plus de supposer que le modèle dans son ensemble approxime correctement la fonction que l'on cherche à minimiser. On suppose simplement que cela est vrai aux alentours des points où l'approximation linéaire a été faite. On vérifiera que les problèmes (3.13) et (3.11) sont équivalents.

Le principal problème réside dans le choix du poids u^k pour que celui-ci reste le plus insensible possible à la valeur de f. En théorie u^k peut être fixé à une quelconque valeur constante $u^k = \bar{u}$, tout en assurant la convergence globale de l'algorithme [3]. Cependant, si $\bar{u}$ est trop grand par rapport à f alors les grandeurs $\|v^k\|$ et $\|d^k\|$ sont petites : nous avons alors toujours des "pas réels", mais la convergence est lente. D'autre part, si $\bar{u}$ est petit, les grandeurs $\|v^k\|$ et $\|d^k\|$ sont grandes et chaque "pas réel" est suivi d'une multitude de "pas nuls" conduisant une fois encore à une convergence lente.

Le cas où u^k est trop élevé peut être détecté par l'inégalité suivante :

$$f(y^{k+1}) \leq f(x^k) + \bar{\mu}v^k, \qquad \bar{\mu} \in [\mu, 1]. \tag{3.14}$$

Cela signifie que u^k devrait être diminué si l'approximation $\hat{f}_k$ de f est proche de f au point y^{k+1}. De nombreuses règles ont été élaborées pour mettre à jour u^k. De manière générale, il parait intéressant de choisir u^{k+1} de façon à conserver une interpolation quadratique de la courbure de f entre x^k et y^{k+1}. En procédant ainsi, les expérimentations numériques montrent que le nombre d'itérations décroît de façon significative [5].

[1] Pour simplifier les notations l'indice k a été omis (c.à.d. , d et v doivent être lus d^k et v^k respectivement).

3.3.2 Fonction convexe sous contraintes convexes

Considérons maintenant le problème suivant :

$$\begin{aligned} \text{Minimiser} \quad & f(x) \\ \text{s.c.} \quad & g(x) \leq 0, \end{aligned} \tag{3.15}$$

où f et g sont convexes et $g = \max\{g_i(x) : \quad i \in I = \{1 \dots m\}\}$.

Pour résoudre ce problème, la méthode consiste à minimiser la fonction suivante :

$$\phi(x, \eta) = f(x) + \eta \max\{g(x), 0\}, \tag{3.16}$$

où $\eta > 0$ est au moins aussi grand que le multiplicateur de Lagrange associé au problème (3.15). Comme la valeur de η n'est que rarement connue à l'avance, ce paramètre va être évolutif (c.à.d. $\eta = \eta^k$). Il va croître durant la minimisation.

Comme on peut déterminer des sous-gradients de f et g séparément, en définissant de façon similaire à f, $\bar{g}_i^k(x) = g_i(y^k) + (x - y^k)^T \nabla g_i(y^k)$ et $\hat{g}_i^k = \max\{\bar{g}_i^j(x) : j \in K\}$, la relaxation de (3.16) est la suivante :

$$\hat{\phi}^k(x, \eta^k) = \hat{f}(x) + \eta^k \max\{\hat{g}^k(x), 0\}. \tag{3.17}$$

Il est alors possible d'utiliser l'algorithme du cas non contraint. Dans ce cas, le point y^{k+1} est défini comme suit :

$$y^{k+1} := \operatorname{argmin} \left\{ \hat{\phi}^k(x, \eta^k) + \frac{u^k \|x - x^k\|^2}{2} : x \in \mathbb{R}^n \right\}, \tag{3.18}$$

avec $v^k = \hat{\phi}^k(y^{k+1}, \eta^k) - \phi(x^k, \eta^k)$ dans l'inégalité (3.12) comme indicateur permettant de discerner si l'on a affaire à un "pas réel" ou à un "pas nul".

L'objectif de la méthode est donc de choisir η^k de sorte que le point soit réalisable et que la fonction ϕ voit ainsi son second terme disparaître. Kiwiel [3] propose d'appliquer la règle : "Si $\eta^k < \lambda^k$ alors $\eta^{k+1} = \max\{\lambda_k, 2\eta^k\}$ sinon $\eta^{k+1} = \eta^k$" où les λ^k sont les multiplicateurs de Lagrange associés aux contraintes dans le problème suivant :

$$y^{k+1} := \operatorname{argmin} \left\{ \hat{f}^k(x) + \frac{u^k \|x - x^k\|^2}{2} : \quad \hat{g}^k(x) \leq 0, \quad x \in \mathbb{R}^n \right\}. \tag{3.19}$$

Comme $\eta^{k+1} \geq \lambda$ la solution du problème (3.19) est aussi solution de (3.18). Bien que l'on puisse directement résoudre le problème (3.18), l'algorithme utilisé pour résoudre le problème initial (3.15) consiste à resoudre le problème plus stable (3.19) pour utiliser dans un second temps les concepts relatifs au problème (3.18), c'est-à-dire choix ou non de y^{k+1} comme nouveau point de référence x^{k+1}.

Une amélioration de cette méthode consiste à ne pas agréger la fonction g mais à considérer chacune de ces composantes. La fonction ϕ s'écrit alors :

$$\phi(x,\eta) = f(x) + \sum_{i=1}^{m} \eta_i g_i(x), \tag{3.20}$$

et la fonction $\hat{\phi}$:

$$\hat{\phi}^k(x,\eta^k) = \hat{f}^k(x) + \sum_{i=1}^{m} \eta_i^k \max\{\hat{g}_i^k(x), 0\}. \tag{3.21}$$

Le point y^{k+1} est :

$$y^{k+1} := \operatorname{argmin}\left\{\hat{\phi}^k(x,\eta^k) + \frac{u^k\|x-x^k\|^2}{2} : x \in \mathbb{R}^n\right\}, \tag{3.22}$$

et le critère pour déterminer si on exécute un “pas réel” ou un pas nul devient :

$$\phi(y^{k+1},\eta^k) \leq \phi(x^k,\eta^k) + \mu v^k, \tag{3.23}$$

où $\mu \in [0; 0.5]$ et, comme nous venons de le voir,

$$v^k = \hat{\phi}^k(y^{k+1};\eta^k) - \phi(x^k;\eta^k).$$

De nombreuses politiques [5] existent pour mettre à jour le coefficient η^k. Nous n'en verrons qu'une ici. De manière générale, plus il est élevé et plus l'algorithme devra trouver les points y^k proche de la frontière de réalisabilité $X = \{x : g_i(x) \leq 0 \quad \forall i \in I\}$. Cependant, s'il est trop élevé, la convergence sera lente. Une règle consiste à augmenter η_i^k de façon à assurer une décroissance de la violation de la contrainte, c'est-à-dire afin que :

$$\hat{g}_i^k(y_{k+1}) \leq \kappa_i^k \max\{g_i(x_k), 0\}, \qquad i \in I, \tag{3.24}$$

où $\kappa_i^k \in [0,1)$ est un facteur de contraction. Reste le problème du choix de κ_i^k. Choisir $\kappa^k = 0$ peut générer des valeurs de η^k inutilement grandes dans les itérations initiales. Il paraît donc plus judicieux de choisir pour commencer des valeurs relativement grandes (par exemple $\kappa = 0.99$ ou 0.999) et de les diminuer jusqu'à les annuler lorsque l'on s'approche de l'optimum.

En étendant les définitions vues dans le cas non contraint :

$$\begin{array}{lrcl}
 & d & = & y^{k+1} - x^k, \\
 & \alpha_{f_j} & = & f(x^k) - \bar{f}^j(x^k), \\
 & \alpha_{g_{ij}} & = & \max\{g_i(x^k), 0\} - \bar{g}_i^j(x^k), \\
 & v & = & v_f + \sum_{i=1}^{m} \eta_i v_{g_i}, \\
\text{où} & v_f & = & \hat{f}^k(y^{k+1}) - f(x^k), \\
\text{et} & v_{g_i} & = & \hat{g}_i^k(y^{k+1}) - g_i(x^k),
\end{array}$$

de manière analogue, le point y^{k+1} peut alors être trouvé en résolvant le système suivant :

$$\begin{array}{rlll}
\text{Minimiser} & \frac{1}{2}u\|d\|^2 + v_f + \sum_{i=1}^{m} \mu_i v_{g_i}, & & \\
\text{s.c.} & -\alpha_{f_j} + d^T \nabla f(y^j) \leq v_f, & \forall j \in K_f & \\
 & -\alpha_{g_{ij}} + d^T \nabla g_i(y^j) \leq v_{g_i}, & \forall j \in K_{g_i}, & \forall i \in I \\
 & (d, v_f, v_{g_1}, \ldots, v_{g_m}) \in \mathbb{R}^{n+1+m}, & &
\end{array} \tag{3.25}$$

où K_f et K_{g_i} sont des sous-ensembles de k où les approximations de f et g_i ont été introduites dans le modèle.

Reste enfin à définir la manière dont évolue le coefficient u^k. Il a été montré [6] que pour évaluer le paramètre u^k, on peut estimer la courbure de la fonction $\phi(., \mu^k)$ entre y^{k+1} et x^k de la manière suivante :

$$u_{int}^{k+1} = 2u^k(1 - [\phi(y^{k+1}, \mu^k) - \phi(x^k, \mu^k)]/v^k). \tag{3.26}$$

Pour éviter des changements drastiques de u^k, Kiwiel [5] conseille de prendre :

$$u^{k+1} = \min[\max\{u_{int}^{k+1}, \frac{1}{10}u^k, u_{min}\}, 10u^k, u_{max}], \tag{3.27}$$

où $0 < u_{min} \ll 1 \ll u_{max}$ sont des constantes.

L'algorithme peut maintenant être défini. Il faut cependant garder à l'esprit qu'il s'agit d'une famille de méthodes. Nous en présentons ici trois, celles qui semblent être les plus utilisées. Voici donc l'algorithme correspondant à la méthode que nous avons décrite ci-dessus.

Méthodes de faisceaux

Initialisation :

k := 0 ;

Sélectionner un point initial x^0 et égaliser $y^0 := x^0$;

Donner un niveau de tolérance pour l'objectif : $\varepsilon_f \geq 0$;

Donner un niveau de tolérance pour la réalisabilité : $\varepsilon_g \geq 0$;

Donner le paramètre $\mu \in [0; 0,5]$;

Définir un poids initial u^0 et des bornes $0 < u_{min} < u_{max}$;

Définir une facteur de contraction $\kappa_{max} \in [0, 1)$;

Définir un coefficient initial de pénalité $\eta^0 > 0$;

Linéarisation de f et de g_i en y^0 ;

Direction de recherche :

Résoudre le problème (3.25) pour déterminer $(d, v_f, v_{g_1}, \ldots, v_{g_m})$;

$y^1 := x^0 + d\,;\;\; v := v_f + \sum_{i=1}^{m} \eta_i^0 v_{g_i}$;

Tant que $v < -\varepsilon_f$ ou $g_i(x^k) > \varepsilon_g \quad \forall i \in I$, **faire**

Mise à jour de la pénalité ;

Pour i de 1 à m, **faire**

Choisir κ_i dans $[0, \kappa_{max}]$;

Si $\hat{g}_i^k(y^{k+1}) > \kappa_i^k \max\{g_i(x), 0\}$, **alors** remplacer μ_i^k par $2\mu_i^k$ **fsi** ;

Fait ;

Si μ^k a changé, **alors**

Direction de recherche :

Résoudre le problème (3.25) pour déterminer $(d, v_f, v_{g_1}, \ldots, v_{g_m})$;

$y^{k+1} := x^k + d\,;\;\; v := v_f + \sum_{i=1}^{m} \eta_i^k v_{g_i}$;

Sinon

Si $\phi(y^{k+1}, \eta^k) \leq \phi(x^k, \eta^k) + \mu^k v$, **alors** $x^{k+1} := y^{k+1}$;

Sinon $x^{k+1} := x^k$ **fsi** ;

Linéarisation de f et de g_i en y^{k+1} ;

Si $x^{k+1} \neq x^k$, **alors** sélectionner u^{k+1} dans $[u_{min}, u^k]$ ou selon (3.27) ;

Sinon $u^{k+1} := u^k$ **fsi** ;

Fsi ;

$\mu^{k+1} := \mu^k\,;\;\; k := k + 1\,;\;\; y^{k+1} := y^k$;

Fin.

On peut montrer [5] que cet algorithme converge en un nombre fini d'itérations si les tolérances ε_f et ε_g sont strictement positives.

3.3.3 Nouvelles variantes

Lemaréchal, Nemirovskii et Nesterov [10] ont présenté de nouvelles variantes pour la méthode de faisceaux. Dans la méthode décrite ci-dessus, en plus de la difficulté de régler les paramètres de réalisabilité η_i, il faut parvenir à définir ce qu'est un "pas nul" et un "pas réel" (c.à.d. définir μ^k) et la pénalité liée à l'approximation linéaire (c.à.d. définir u^k). Si des règles heuristiques efficaces ont pu être trouvées pour μ^k, il n'en est malheureusement pas de même pour u^k.

D'autres alternatives peuvent être imaginées pour déterminer le point y^{k+1}. L'une d'elle serait de définir une région dans laquelle l'approximation linéaire est considérée comme raisonnable. On pourrait définir cette zone comme une boule de centre x^k et de rayon τ^k. Plus τ^k est grand, plus important est le domaine sur lequel on accepte le modèle. Dans le cas non contraint, (3.11) se récrirait alors :

$$y^{k+1} := \text{argmin}\{\hat{f}^k(x) : \quad \|x - x^k\| \leq \tau^k, \quad x \in \mathbb{R}^n\}. \tag{3.28}$$

Cette méthode mentionnée dans [10] ne semble pas avoir été étudiée.

Une autre approche [11] se caractérise par un détour dans l'espace dual. Le paramètre de contrôle est alors ε^k.

Dans l'approche de Lemaréchal, Nemirovskii et Nesterov, le paramètre de contrôle n'est plus u^k, τ^k ou ε^k mais un niveau l^k. Dans le cas non contraint, (3.11) se récrit :

$$y^{k+1} := \text{argmin}\{\frac{1}{2}\|x - x^k\|^2 : \quad \hat{f}^k(x) \leq l^k, \quad x \in \mathbb{R}^n\}. \tag{3.29}$$

La sélection du niveau l^k conduisant à des comportements relativement stables, des règles extrêmement simples peuvent être utilisées. Cette propriété a permis de supprimer le concept de "pas nul". x^{k+1} peut donc être toujours égalisé à y^{k+1} à chaque itération. Nous ne mentionnerons donc plus par la suite la série y^{k+1} et dans (3.29) y sera remplacé par x.

Le concept de base, pour déterminer l^k, est le suivant : à l'itération k, on calcule une borne inférieure $\underline{f}^k$ de f en recherchant la valeur du minimum de la relaxation

$\hat{f}^k(x)$. On détermine aussi une borne supérieure :

$$\overline{f}^k = \min\{f(x^k) : \quad \forall k \in K\}. \tag{3.30}$$

Le saut de dualité est donc défini de la manière suivante :

$$\Delta^k = \overline{f}^k - \underline{f}^k. \tag{3.31}$$

En prenant $\mu \in [0,1]$, l'objectif est de résoudre (3.29) en prenant pour l^k la valeur :

$$l^k = \mu\overline{f}^k + (1-\mu)\underline{f}^k = \underline{f}^k + \mu\Delta^k. \tag{3.32}$$

On remarquera que si l'on prend $\mu = 1$ dans (3.32) alors $l^k = \overline{f}^k$ et $x^{k+1} = x^k$. En effet, comme $\hat{f}^k(x) \leq \overline{f}^k$, le minimum de (3.29) est atteint pour $x^{k+1} = x^k$. Pour des valeurs de μ proches de 1, la méthode agit comme une pure méthode de sous-gradients avec de très petits pas. En effet, puisque $x^k \approx x^{k+1}$ la direction $x^{k+1} - x^k$ va être donnée par l'approximation de f en x^k, c'est-à-dire, son sous-gradient en x^k. Par opposition, si $\mu = 0$, la méthode sera alors équivalente à la méthode de plans coupants de Kelley. Cette remarque suggère qu'il est préférable de choisir de petites valeurs pour μ.

Dans le cas du problème (3.15) de minimisation d'une fonction convexe sur un ensemble convexe, on considère le problème équivalent :

$$\min_x\{\max\{f(x) - f^*, g(x)\} : \quad x \in \mathbb{R}^n\}, \tag{3.33}$$

où f^* est la valeur optimale. En général cette valeur étant inconnue, on est confronté à une nouvelle difficulté : comment calculer une approximation linéaire de (3.33) en un point x^k. Pour résoudre ce problème, f^* est sous-estimée par le minimum de la relaxation linéaire $\underline{f}^k$. Deux approches sont ensuite envisageables :

- Le problème (3.33) est équivalent à :

  $$\min_{x\in\mathbb{R}^n}\{\max_{0\leq\eta\leq 1}\{\eta(f(x) - f^*) + (1-\eta)g(x)\}\}. \tag{3.34}$$

  Les conditions du théorème de la dualité [18] (ou théorème minmax [19]) étant satisfaites[2], le problème (3.34) est équivalent à :

  $$\max_\eta\{\phi(\eta) - \eta f^* : \quad 0 \leq \eta \leq 1\}, \tag{3.35}$$

[2] Le problème (3.34) est continue, convexe en x et concave en η.

où

$$\phi(\eta) = \min_x \{\eta f(x) + (1-\eta) g(x) : \quad x \in \mathbb{R}^n\}.$$

La fonction $\phi(\eta)$ peut être sur-estimée par :

$$\phi^k(\eta) = \min_{x_j} \{\eta f(x^j) + (1-\eta) g(x^j) : \quad j \in K\}.$$

La mesure sur la qualité de la solution $\bar{\Delta}^k$ est alors obtenue par :

$$\bar{\Delta}^k = \max\{\phi^k(\eta) - \eta \overline{f}^k : \quad 0 \leq \eta \leq 1\}.$$

En définissant $P_X x$ comme la projection orthogonale de x sur X, l'algorithme est le suivant :

1ère méthode de faisceaux avec niveaux

Paramètres :

$\mu \in [0,1]$;

$\theta \in [0,1]$;

Définir un niveau de tolérance : $\varepsilon_f \geq 0$;

Initialisation :

$k := 0$;

$x^0 \in \mathbb{R}^n$;

$\eta^0_{min} := 0$, $\eta^0_{max} := 1$;

$\eta^0 := \frac{1}{2}$;

Linéarisation de f et de g en x^0 ;

Calcul de $\underline{f}^k$ et $\bar{\Delta}^k$;

Tant que $\bar{\Delta}^k > \varepsilon_f$, **faire**

Définir η^k_{min} et η^k_{max} tel que $\phi^k(\eta) \geq 0$;

Si $\frac{\eta^k - \eta^k_{min}}{\eta^k_{max} - \eta^k_{min}} \in [\frac{\theta}{2}, 1 - \frac{\theta}{2}]$, **alors** $\eta^{k+1} := \frac{\eta^k_{min} + \eta^k_{max}}{2}$;

Sinon $\eta^{k+1} := \eta^k$ **fsi** ;

$w^k := \eta^k \underline{f}^k$;

$W^k := \min_{j \in K} \{\eta^k f(x^j) + (1-\eta^k) g(x^j)\}$;

$l^k := w^k + \mu(W^k - w^k)$;

$x^{k+1} := P_{\{x:\ \eta^k \hat{f}^k(x) + (1-\eta^k)\hat{g}^k(x) \leq l^k\}} x^k$;

$k := k+1$;

Linéarisation de f et de g en x^k ;

Calcul de $\underline{f}^k$ et $\bar{\Delta}^k$;

Fin.

Lemaréchal, Nemirovskii et Nesterov [10] montrent que $\Delta^k < \bar{\Delta}^k$ et, si $k > c(\mu,\theta)(V/\varepsilon)^2 \log(2V/\varepsilon)$ alors $\Delta^k \leq \varepsilon$, où :

$$c(\mu,\theta) = \frac{2(1+1/\theta)^2}{\log 2 \log(2/(1+\theta))(1-\mu)^2(2-\mu)\mu}$$

et $V = DL$ avec D le diamètre dans la norme euclidienne de l'ensemble sur lequel est défini le problème et L la plus grande constante de Lipschitz des fonctions $f, g_1, \ldots, g_m$. On notera que $\min c(.,.) = c(0.29289\ldots, 0.53247\ldots) \leq 360$.

Si cet estimateur de la complexité de l'algorithme est bien meilleur que celui trouvé pour la méthode de Kelley, il n'est malheureusement pas celui d'un algorithme polynomial, mais pseudo-polynomial.

- Dans une seconde approche, la valeur de l'optimum f^* est remplacée par un paramètre η et le problème à résoudre est alors :

$$\phi(\eta) = \min\{\max\{f(x) - \eta, g(x)\} : \quad x \in \mathbb{R}^n\} = 0.$$

La résolution de ce problème se fait aussi par une approximation linéaire basée sur la génération de plans coupants. Le paramètre η est mis à la valeur de $\underline{f}^k$ chaque fois que la mesure sur la qualité de la solution décroit d'une quantité suffisante.

Définissons deux fonctions, l'une majorant et l'autre minorant ϕ :

$$\overline{\phi}^k(\eta) = \min\{\max\{f(x^i) - \eta, g(x^i)\} : \quad i \in K\};$$
$$\underline{\phi}^k(\eta) = \min\{\max\{\hat{f}^k(x) - \eta, \hat{g}^k(x)\} : \quad x \in \mathbb{R}^n\}.$$

La mesure sur la qualité de la solution est donnée par l'expression :

$$\bar{\Delta}^k = \overline{\phi}^k(\eta^k).$$

L'algorithme est le suivant :

$2^{\text{ème}}$ méthode de faisceaux avec niveaux

Paramètres :
- $\mu \in (0,1)$;
- $\theta \in (\frac{1}{2},1)$;

Initialisation :
- $k := 0$;
- $x^0 \in \mathbb{R}^n$;

Linéarisation de f et de g en x^0 ;
Calcul de $\underline{f}^0$, $\overline{\phi}^0(\eta^0)$, $\underline{\phi}^0(\eta^0)$ et $\bar{\Delta}^0$;

Tant que $\bar{\Delta}^k > \varepsilon$, **faire**
- **Si** $\underline{\phi}^k(\eta^k) > \theta\overline{\phi}^k(\eta^k)$, **alors** $\eta^{k+1} := \underline{f}^k$;
- **Sinon** $\eta^{k+1} := \eta^k$ **fsi** ;
- $w^k := \underline{\phi}^k(\eta^{k+1})$;
- $W^k := \overline{\phi}^k(\eta^{k+1})$;
- $l^k := w^k + \mu(W^k - w^k)$;
- $x^{k+1} := P_{\{x:\ \max_{x\in\mathbb{R}^n}\{\hat{f}^k(x)-\eta,\hat{g}^k(x)\}\leq l^k\}}x^k$;
- $k := k+1$;
- **Linéarisation** de f et de g en x^k ;
- **Calcul** de $\underline{f}^k$, $\overline{\phi}^k(\eta^k)$, $\underline{\phi}^k(\eta^k)$ et $\bar{\Delta}^k$;

Fin.

Lemaréchal, Nemirovskii et Nesterov [10] montrent que $\Delta^k < \bar{\Delta}^k$ et, si $k > c(\mu,\theta)(V/\varepsilon)^2\log(18V/\varepsilon)$ alors $\Delta^k \leq \varepsilon$, où :

$$c(\mu,\theta) = \frac{2}{\log(2\theta)(1-\theta)^2(1-\mu)^2(2-\mu)\mu}$$

et $V = DL$ avec D le diamètre dans la norme euclidienne de l'ensemble sur lequel est défini le problème et L la plus grande constante de Lipschitz des fonctions $f, g_1, \ldots, g_m$. On notera que $\min c(.,.) = c(0.29289\ldots, 0.65252\ldots) \leq 249$.

Pour les mêmes raisons que pour la $1^{\text{ère}}$ méthode de faiseaux avec niveaux, cet algorithme, bien que d'une meilleure complexité que celui de Kelley, n'est pas polynomial, mais pseudo-polynomial.

3.4 Méthodes des centres

La première méthode de plans coupants faisant appel à un concept central a été proposée par Elzinga et Moore en 1974 [1]. Elle a été l'impulsion d'une direction de recherche qui a donné naissance à de nombreux algorithmes. Ces méthodes consistent non pas, comme dans la méthode de Kelley ou les méthodes de faisceaux, à déterminer le point x^{k+1} sur la frontière définie par l'approximation linéaire courante du problème mais à le choisir comme "centre" d'un ensemble borné convexe appelé ensemble de localisation $\mathcal{L}$.

L'ensemble de localisation $\mathcal{L}$ est défini par l'intersection des demi-espaces générés par les formes linéaires de l'approximation courante et par une borne supérieure $\overline{f}$ de la solution (voir figure 3.4). Par définition il contient une solution optimale.

$$\mathcal{L} = \{x : Ax \leq b, c^T x \leq \overline{f}\}.$$

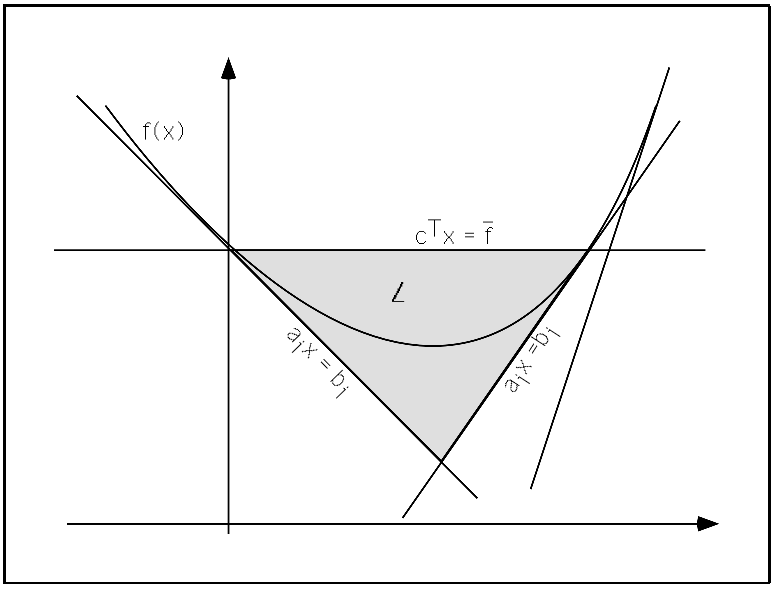

Figure 3.4: Ensemble de localisation $\mathcal{L}$.

Les méthodes des centres ne diffèrent que dans la manière dont elles définissent un "centre" dans cet ensemble $\mathcal{L}$. Nous verrons ici les principales idées qui ont été élaborées : la méthode d'Elzinga et Moore [1], la méthode du centre de gravité [12] et la méthode volumétrique [20]. Les chapitres suivants décriront de manière plus approfondie l'algorithme ACCPM qui est à l'origine de cette thèse et qui est aussi basé sur un concept de "centre".

3.4.1 Méthode d'Elzinga et Moore

La méthode d'Elzinga et Moore [1], définit le “centre” x^{k+1} comme le centre de la plus grande hypersphère comprise dans l'ensemble de localisation $\mathcal{L}$. La figure 3.5 représente une itération en indiquant comment a été déterminé le point x^k et x^{k+1}.

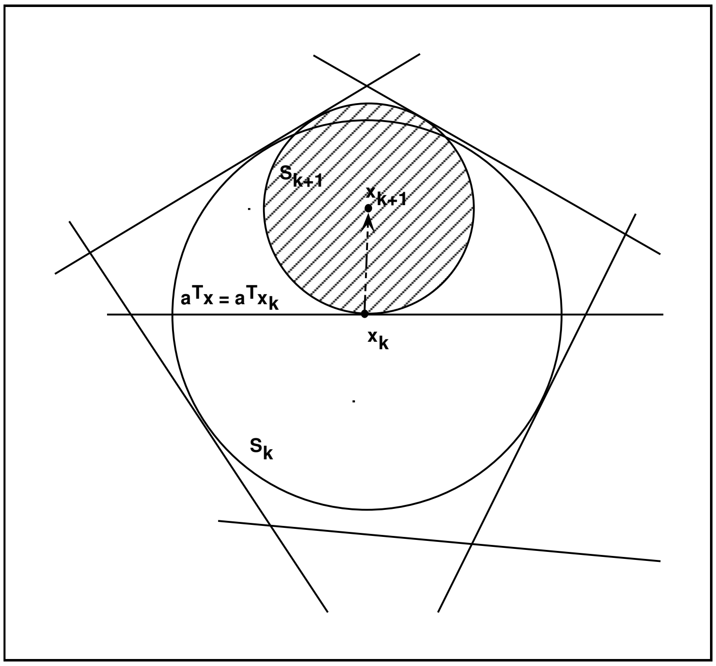

Figure 3.5: Méthode d'Elzinga et Moore.

L'origine de la méthode est due aux travaux de Nemhauser et Widhelm [17] qui ont démontré que si un ensemble d'inégalités définit un polyèdre borné non vide :

$$\{x : a_i^T x \leq b_i, \quad i \in I\},$$

où $I = \{1 \ldots m\}$, alors le problème :

$$\begin{array}{ll} \text{Minimiser} & \sigma \\ \text{s.c.} & a_i^T x + \|a_i^T\|\sigma \leq b_i, \qquad\qquad i \in I, \end{array}$$

permet de trouver le rayon σ et le centre x de la plus large hypersphère qui peut être incluse dans ce polyèdre.

L'algorithme de résolution du problème de base (3.4) peut s'exprimer de la manière suivante :

Méthode d'Elzinga et Moore

Paramètres :

Donner un niveau de tolérance pour l'objectif : $\varepsilon \geq 0$;

Soit B un ensemble polyédrique convexe et compact de $\mathbb{R}^n$;

Initialisation :

$k := 0$;

Déterminer une borne supérieure de la valeur de la solution : $\overline{f}$;

$P_0 := \{x : c^T x + \sigma \leq \overline{f},\ x \in B\}$;

$x^0 := \text{argmax}\{\sigma :\ (x, \sigma) \in P_0\}$;

Tant que $\sigma \geq \varepsilon$, **faire**

Si $g_i(x_k) \leq 0 \quad \forall i \in \{1 \dots p\}$, **alors**

Ajouter à P_k la nouvelle contrainte sur l'objectif : $c^T x + \sigma \leq c^T x^k = \overline{f}$;

Sinon

Ajouter à P_k la contrainte : $g_i(x_k) + \nabla^T g_i(x_k)(x - x_k) + \|\nabla g_i(x_k)\|\sigma \leq 0$;

Fsi ;

$x^{k+1} := \text{argmax}\{\sigma :\ (x, \sigma) \in P_{k+1}\}$;

$k := k + 1$;

Fin.

Elzinga et Moore [1] ont montré que, toute sous-séquence convergente $\{x^{\tilde{k}}\}_{\tilde{k}=0}^{\infty}$ converge vers une solution optimale. De plus, ils ont montré que la sous-suite composée des points réalisables, converge linéairement vers l'optimum.

En conclusion, bien que les calculs à chaque itération soient relativement simples (résolution d'un Simplexe), rien dans l'étude sur la convergence ne prouve que l'algorithme va trouver une solution rapidement.

3.4.2 Méthode du centre de gravité.

Levin [12] en 1965 a proposé une méthode où le plan coupant est généré au centre de gravité de l'ensemble de localisation. Cette méthode a été motivée par la remarque théorique [15] qui indique que l'hyperplan passant par le centre de gravité coupe le volume v de l'ensemble de localisation en deux parties de volume au moins égal à $(1-1/(n+1))^n v$. Après k itérations, le volume v_k de l'ensemble de localisation est alors borné par $v_k < (1-e^{-1})^k v_0$. Il est important de noter que ce volume ne dépend pas de la dimension du problème (c.à.d. il est indépendant de n) et qu'il décroît géométriquement. Nemirovskii [16] montre que cet algorithme résoud le problème (1.6) avec une précision relative de ε en moins de

$$O(1)k \ln\left(\frac{1}{\varepsilon}\right),$$

itération. Il montre, par ailleurs, que c'est le meilleur taux de convergence à un facteur constant près que l'on puisse obtenir. Malheureusement, le calcul du centre de gravité est un problème probablement plus compliqué que la résolution du problème original. C'est pourquoi, bien que dotée d'un taux de convergence optimal, cette méthode n'a pas débouché sur des résultats concrets dans la pratique.

3.4.3 Méthode volumétrique

Cette méthode proposée par Vaidya [20] a initialement été conçue pour trouver un point réalisable, c'est-à-dire, un point appartenant à l'ensemble convexe X. Cependant, cette méthode peut aisément s'adapter aux problèmes de la minimisation d'une fonction convexe sur un ensemble convexe.

Dans cette méthode des centres, le point x^{k+1}, appelé centre volumétrique, est déterminé ainsi :

$$x^{k+1} = \min\{F(x) = \frac{1}{2}\log(\det(H(x)) : \quad x \in \mathcal{L}\}, \tag{3.36}$$

où

$$H(x) = \sum_{i=0}^{m} \frac{a_i a_i^T}{(b_i - a_i^T x)^2},$$

est le Hessien évalué en x de la fonction barrière $\sum_{i=0}^{m} \log(b_i - a_i^T x)$ et où a_i^T est la $i^{\text{ème}}$ ligne de la matrice A de l'ensemble de localisation $\mathcal{L}$ si $i \neq 0$, et où $a_0 = -c$ et $b_0 = -\overline{f}$.

Vaidya [20] montre que la fonction $F(x)$ est strictement convexe. Une approximation du point x^{k+1} peut donc être calculée par la méthode de Newton.

L'intérêt de choisir x^{k+1} ainsi est le suivant : à chaque $x \in \mathcal{L}$ est associée un ellipsoïde $E(x) = \{y : (y-x)^T H(x)(y-x) \leq 1\}$ contenu dans $\mathcal{L}$. L'ellipsoïde $E(x)$ peut être vu comme l'approximation quadratique locale de $\mathcal{L}$. Si x^{k+1} est définie par (3.36), $E(x^{k+1})$ est l'ellipsoïde inscrite de volume maximum. Un plan passant par x^{k+1} divise $E(x^{k+1})$ en deux parties d'égal volume ce qui laisse espérer que $\mathcal{L}$ sera divisé en deux parties de volumes voisins. La figure 3.6 illustre une itération de cette méthode.

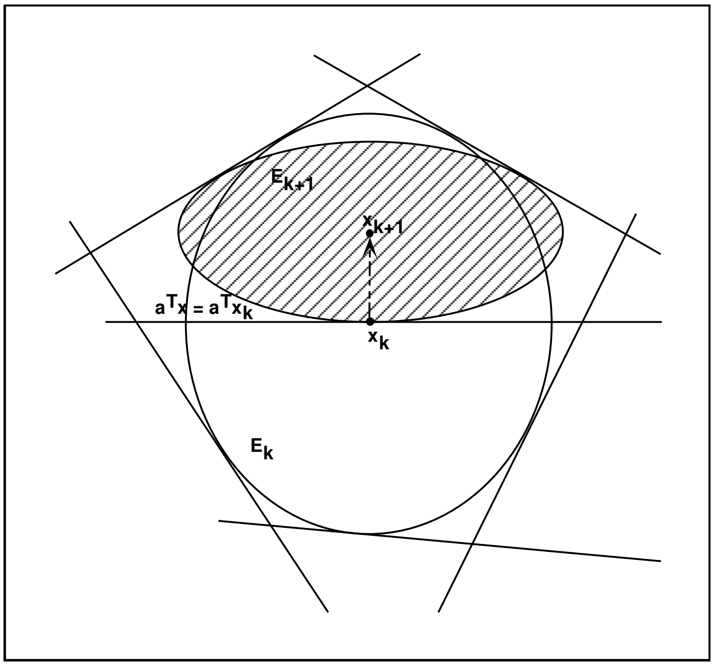

Figure 3.6: Méthode volumétrique.

Vaidya [20] montre aussi que le volume de $\mathcal{L}$ décroît en moyenne d'un facteur constant à chaque itération, facteur qui est indépendant de n. L'algorithme s'arrête alors avec un point dans X en $O(nL)$ itérations où L est tel que 2^L est le rayon d'une boule contenant l'ensemble X et 2^{-L} le rayon d'une boule contenue dans l'ensemble X. Il est important de noter que pour obtenir ce résultat sur la convergence, l'algorithme doit obligatoirement intégrer un test permettant d'éliminer de $\mathcal{L}$ des formes linéaires de sorte que le nombre de plans le décrivant reste proportionnel à la dimension n de l'espace de x. Si ce type de test n'est pas incorporé, l'algorithme s'arrête après $O(n^2L^2)$ itérations et non $O(nL)$.

Dans la méthode volumétrique, le centre volumétrique est donc défini comme le minimum de la fonction $F(x)$ avec $x \in \mathcal{L}$. Une approximation de ce point étant déterminée par la méthode de Newton, il nous faut calculer son gradient et son Hessien en x. En définissant

$$\sigma_i(x) = \frac{a_i^T H(x)^{-1} a_i}{(b_i - a_i^T x)^2}, \qquad\qquad 0 \leq i \leq m,$$

le gradient s'écrit simplement :

$$\nabla F(x) = -\sum_{i=0}^{m} \sigma_i(x) \frac{a_i}{b_i - a_i^T x}.$$

Le Hessien est approximé par (voir [20]) :

$$\nabla^2 F(x) \approx Q(x) = \sum_{i=0}^{m} \sigma_i(x) \frac{a_i a_i^T}{(b_i - a_i^T x)^2},$$

Pour satisfaire la qualité de l'estimation de la convergence, l'algorithme doit intégrer un dispositif premettant de garder le nombre de formes linéraires dans $\mathcal{L}$ relativement constant. L'algorithme est alors divisé en deux parties. Une partie intégrant dans le modèle les formes linéaires et une partie supprimant de ce modèle celles jugées inutiles dans la définition de l'optimum. Après chaque modification du modèle, un nouveau centre volumétrique doit être trouvé. Comme son calcul exact serait inutile et trop coûteux, seule une approximation est calculée. Sans entrer dans le détail, si le point précédent x^k satisfait certaines conditions quant à sa proximité du centre volumétrique, il peut être montré [20] que le nombre d'itérations nécessaires pour trouver un nouveau point x^{k+1} satisfaisant ces mêmes conditions est majoré par $\lceil 30 \ln(2\varepsilon^{-4.5}) \rceil$ si une forme linéaire a été ajoutée et $\lceil 30 \ln(4\varepsilon^{-3}) \rceil$ si une forme linéaire a été supprimée. L'algorithme est alors le suivant :

Méthode Volumétrique

Paramètres :

Donner $\alpha \leq 10^{-4}$;

Donner $\varepsilon \leq 10^{-3}\alpha$;

Initialisation :

$k := 0$;

$P_0 := \{x : \sum_{j=1}^{n} x_j \leq n2^L \quad , x_i \geq -2^L \quad , 1 \leq j \leq n\}$;

Calcul de x^0 ;

Linéarisation en x^0 ;

Tant que $x^k \notin X$, **faire**

Ajouter au modèle le plan séparateur ;

Pour j de 1 à $\lceil 30 \ln(2\varepsilon^{-4.5}) \rceil$, **faire**

$x^k := x^k - 0.18Q(x^k)^{-1}\nabla F(x^k)$;

Fait ;

Tant que $\min_{1 \leq i \leq m} \{\sigma_i(x^k)\} < \varepsilon$, **faire**

$\sigma_j(x^k) := \min_{1 \leq i \leq m} \{\sigma_i(x^k)\}$;

Supprimer le plan j de P_k ;

Pour j de 1 à $\lceil 30 \ln(4\varepsilon^{-3}) \rceil$, **faire**

$x^k := x^k - 0.18Q(x^k)^{-1}\nabla F(x^k)$;

Fait ;

Fait ;

Linéarisation en x^k ;

$k = k + 1$;

Fin.

Bibliographie

[1] J. Elzinga, T. G. Moore, “A central cutting plane algorithm for the convex programming problem”, *Mathematical Programming* **8** (1973) 134-145.

[2] J. E. Kelley, “The cutting-plane method for solving convex programs”, *Journal of the SIAM* **8** (1960) 703-712.

[3] K. C. Kiwiel, *Methods of descent for nondifferrentiable optimization*, Lectures Notes in Economics and Mathematics 1133, Springer, Berlin (1985).

[4] K. C. Kiwiel, “A survey of bundle methods for nondifferentiable optimization”, M. Iri et K. Tanabe, eds.,*Mathematical Programming : Recent Developments and Applications* KTT/Kluwer, Tokyo, (1989) 263-282.

[5] K. C. Kiwiel, “Exact penalty functions in proximal bundle methods for constrained convex nondifferentiable minimisation”, Technical Report of Systems Research Institute, Newelska 6, 01-447 Warsaw, Poland (1989).

[6] K. C. Kiwiel, “Proximity control in bundle methods for convex nondifferentiable minimization”, *Mathematical Programming* **46** (1990) 105-122.

[7] K. C. Kiwiel, “A bundle method for minimizing a sum of convex functions with smooth weights”, Working paper of IIASA’, A-2361 Laxenburg, Austria (1994).

[8] C. Lemaréchal, “An algorithm for minimizing convex functions”, Proceeding IFIP ’74 Congress J. L. Rosenfeld ed., North-Holland, Amsterdam (1974) 552-556.

[9] C. Lemaréchal, “An extension of Davidon methods to nondifferentiable problems”, *Mathematical Programming* **3** (1975) 95-109.

[10] C. Lemaréchal, A. Nemirovskii et Y. Nesterov, “New variants of bundle methods”, Technical Report 1508, Institut National de Recherche en Informatique

et en Automatique, Domaine de Voluceau, Rocquencourt, B.P. 105, 78153 La Chesnay Cedex, France (September 1991).

[11] C. Lemaréchal, J. J. Strodiot et A. Bihain, *On a bundle algorithm for nonsmooth optimization*, Nonlinear Programming **4** O. L. Mangasarian, R. R. Meyer, S. M. Robinson (eds.), Academic Press (1981) 245-282.

[12] A. Y. Levin, “On an algorithm for the minimization of convex functions over convex functions”, *Soviet Mathematical Doklady* **6** (1965) 286-290.

[13] D. G. Luenberger *Linear and nonlinear programming*, Addison Wesley (1984).

[14] R. Mifflin, “An algorithm for constrained optimization with semismooth functions”, *Mathematics of Operations Research* **2** (1977) 191-207.

[15] H. Minkowski, Uspehi Mat. Nauk **2** (1936).

[16] A. Nemirovskii *unpublished Lecture notes* Technion,The Israel Institute of Technology, Faculty of Industrial Engineering & Management (1994).

[17] G. L. Nemhauser, W. B. Widhelm, “A modified linear program for columnar methods in mathematical programming”, *Operations Research* **19** (1971) 1051-1060.

[18] M. Minoux, *Programmation mathématique : théorie et algorithmes*, **2** Dunod (Bordas), Paris (1983).

[19] R. T. Rockafellar, *Convex analysis*, Princeton University Press, (1972).

[20] P. Vaidya, “A new algorithm for minimizing convex functions over convex sets”, AT&T Bell Laboratories, Murray Hill, NJ 07974.

[21] P. Wolfe, “A method of conjugate subgradients for minimizing nondifferentiable convex functions”, *Mathematical Programming* **3** (1975) 145-173.

[22] W. I. Zangwill, *Nonlinear programming : a unified approach*, Prentice-Hall, Englewoog Cliffs, N.J., (1969).

Partie II

THEORIE SUR L'ALGORITHME ACCPM

Chapitre 4

Algorithme ACCPM : étude théorique

Dans la partie précédente nous avons présenté la problématique liée à l'optimisation convexe non différentiable. Les parties suivantes décrivent l'algorithme ACCPM. Nous verrons qu'il s'agit d'une méthode de plans coupants qui peut être mise dans la famille des méthodes des centres.

Dans ce chapitre nous verrons tout d'abord les principes de base de l'algorithme ACCPM (rappel de la méthode des plans coupants), puis nous définirons la relaxation linéaire du problème original, l'oracle (procédure qui génère les coupes), l'ensemble de localisation et le centre analytique. Nous donnerons ensuite l'algorithme ACCPM de haut niveau et terminerons par un tour d'horizon sur les études de convergence qui ont été faites.

4.1 Principes

Comme nous l'avons vu au chapitre précédent, il existe un grand nombre d'algorithmes basés sur la génération de plans coupants. Ils résolvent le problème suivant :

$$\begin{aligned} \text{Minimiser} \quad & f(y) \\ \text{s.c.} \quad & y \in Y \\ & l \leq y \leq h, \end{aligned} \tag{4.1}$$

où la fonction $f : \mathbb{R}^n \to \mathbb{R}$ et l'ensemble Y sont convexes et où l et h sont des bornes sur y. Par la suite nous appellerons les inégalités $l \leq y \leq h$, contraintes de boîtes.

Ces algorithmes sont constitués de deux procédures. La première, généralement appellée **programme-maître**, travaille sur une relaxation linéaire du problème original (4.1). Cette relaxation s'affine à chaque itération par adjonction d'une ou plusieurs coupes. A une itération donnée, le programme-maître, au vu de la relaxation courante envoie une proposition y à la deuxième procédure appelée **oracle**. L'oracle reçoit le vecteur y et renvoie au programme-maître une information sur le problème (4.1) en ce point afin d'enrichir l'approximation linéaire de ce dernier. Plus précisement, si y est réalisable, $y \in Y$, l'oracle renvoie un plan de support de la fonction f en y. Sinon, il renvoie un plan séparant y de Y. Dantzig [4] appelle ces coupes, **coupe d'optimalité** dans le premier cas et **coupe de réalisabilité** dans le second. Nous adopterons cette terminologie. Le mécanisme est illustré par la figure 4.1.

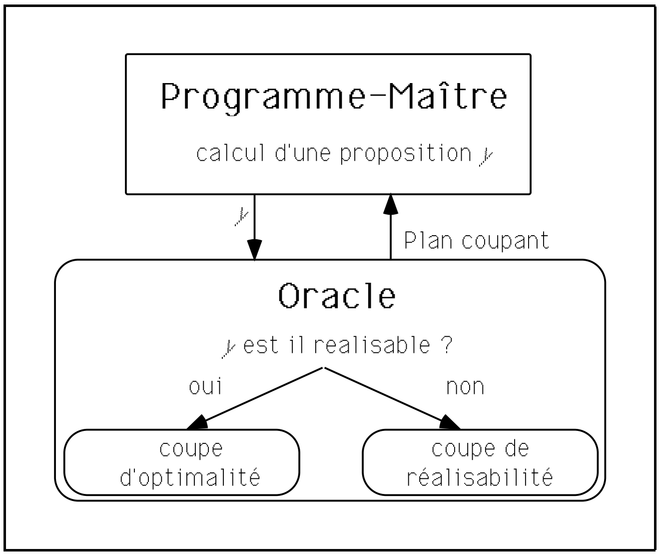

Figure 4.1: Méthode de plans coupants.

Cette procédure prend fin lorsque le programme-maître considère qu'il a atteint l'optimum.

Bien qu'étant tous semblables dans leur fonctionnement, ces algorithmes diffèrent sur la politique à adopter lors du calcul de la proposition y à envoyer à l'oracle. Kelley-Golstein et Dantzig-Wolfe considèrent qu'elle doit être le minimum de la relaxation courante. Pour les méthodes de faisceaux, elle fait l'objet d'un arbitrage entre le minimum de la relaxation courante et la distance à l'ancienne proposition (c.à.d. recherche du minimum dans un voisinage). Ces méthodes déterminent le nouveau point sur la frontière définie par la relaxation linéaire courante. D'autres méthodes, dites centrales, définissent un ensemble de localisation. Cet ensemble contient une solution optimale du problème. Elles calculent ensuite un point y dans cet ensemble. Dans ces méthodes on trouve : l'algorithme d'Elzinga-Moore, la méthode du centre de gravité et la méthode volumétrique.

Dans l'algorithme que nous allons maintenant étudier, ce point est défini comme le centre analytique de l'ensemble de localisation. C'est pourquoi, nous avons appellé cet algorithme : **l'algorithme ACCPM** (Analytic Center Cutting Plane Method).

4.2 Relaxation linéaire

L'algorithme consiste à approximer la fonction f et l'ensemble Y par des applications linéaires. Nous avons vu qu'à chaque itération, le programme-maître reçoit de l'oracle, une ou plusieurs coupes qui peuvent être de deux types : des coupes d'optimalité ou des coupes de réalisabilité.

4.2.1 Coupe d'optimalité

De par la convexité de f, un plan de support en y^i vérifie l'inégalité :

$$f(y) \geq f(y^i) + \xi(y - y^i), \qquad \xi \in \partial f(y^i) \quad \forall y \in Y,\ l \leq y \leq h. \tag{4.2}$$

En définissant la fonction :

$$\hat{f}(y) = \max_{y^i \in I}\{f(y^i) + \xi(y - y^i) : \quad \xi \in \partial f(y^i)\},$$

où $I \subseteq \{y : y \in Y, l \leq y \leq h\}$, on déduit de (4.2) les propriétés suivantes :

- Si I est l'ensemble $\{y : y \in Y, l \leq y \leq h\}$, alors $\hat{f}(y) = f(y)$. En effet, on a $y^i = y$.

- Si I est un-sous ensemble fini de $\{y : y \in Y, l \leq y \leq h\}$ alors $\hat{f}(y) \leq f(y)$. La fonction $\hat{f}(y)$, suprémum d'un ensemble d'applications linéaires, est donc une relaxation linéaire de f.

Si l'on omet momentanément les contraintes, le problème (4.1) s'écrit :

$$\begin{aligned} \text{Minimiser} \quad & f(y) \\ \text{s.c.} \quad & l \leq y \leq h. \end{aligned} \tag{4.3}$$

En posant :

$$f(y; y^i) = f(y^i) + \xi(y - y^i), \qquad \xi \in \partial f(y^i),$$

on peut alors définir le programme linéaire (4.4), relaxation du problème (4.3) :

$$\begin{aligned} \text{Minimiser} \quad & z \\ \text{s.c.} \quad & z \geq f(y; y^i), \qquad y^i \in I \\ & l \leq y \leq h. \end{aligned} \tag{4.4}$$

où I est l'ensemble composé de tous les points y envoyés à l'oracle. Une coupe d'optimalité est un hyperplan de support de f en y^i :

coupe d'optimalité : $\qquad z \geq f(y; y^i).$

4.2.2 Coupe de réalisabilité

Une coupe de réalisabilité est un hyperplan séparant le point y^j envoyé à l'oracle de l'ensemble Y des points admissibles. Cela consiste à trouver un vecteur $\omega \in \mathbb{R}^n$ et un scalaire $\gamma \in \mathbb{R}$ tel que $Y \subseteq \{y : \omega^T y \leq \gamma\}$ et $y^j \in \{y : \omega^T y > \gamma\}$.

Soit $g(y, y^j) = \omega^T y - \gamma$, de par la convexité de Y, il est possible de définir une approximation polyédrique $\hat{Y}$ de Y :

$$Y \subseteq \hat{Y} = \{y : g(y; y^j) \leq 0, \quad y^j \in J\},$$

où $J \subseteq \{y : y \notin Y, l \leq y \leq h\}$. Tout comme pour l'approximation de f on déduit de la convexité de Y les propriétés suivantes :

- Si J est l'ensemble $\{y : y \notin Y, l \leq y \leq h\}$, alors $\hat{Y} = Y$.
- Si J est un sous-ensemble fini de $\{y : y \notin Y, l \leq y \leq h\}$ alors $\hat{Y} \supset Y$.

Une coupe de réalisabilité en y^j est donc l'hyperplan :

coupe de réalisabilité : $\qquad g(y; y^j) \leq 0.$

Prenons par exemple $Y = \{y : g(y) \leq 0\}$ avec $g : \mathbb{R}^n \to \mathbb{R}$ une fonction convexe. Comme pour f on peut définir l'approximation linéaire de $g(y)$:

$$\hat{g}(y) = \max_{y^j \in J}\{g(y^j) + \xi(y - y^j) : \quad \xi \in \partial g(y^j)\},$$

et la relaxation polyèdrique de Y :

$$\hat{Y} = \{y : \hat{g}(y) \leq 0\}.$$

En posant :

$$g(y; y^j) = g(y^j) + \xi(y - y^j), \qquad\qquad \xi \in \partial g(y^j),$$

on définit la coupe de réalisabilité en y^j : $g(y; y^j) \leq 0$.

4.2.3 Relaxation linéaire : définition

On peut maintenant définir le programme linéaire (4.5), relaxation du problème original (4.1) :

$$\begin{aligned} \text{Minimiser} \quad & \hat{f}(y) \\ \text{s.c.} \quad & y \in \hat{Y} \\ & l \leq y \leq h, \end{aligned} \tag{4.5}$$

ou bien, si I est l'ensemble des points y^i envoyés à l'oracle tel que $y^i \in Y$ et J est l'ensemble des points y^j envoyés à l'oracle tel que $y^j \notin Y$,

$$\begin{aligned} \text{Minimiser} \quad & z \\ \text{s.c.} \quad & z \geq f(y; y^i), && y^i \in I \\ & g(y; y^j) \leq 0, && y^j \in J \\ & l \leq y \leq h. \end{aligned} \tag{4.6}$$

Nous minimisons alors, une approximation extérieure de la fonction f sur un domaine élargi de Y.

Il est intéressant de noter que la fonction f et l'ensemble Y n'ont nullement besoin d'être connus explicitement. Seule une approximation linéaire en un point donné doit pouvoir être fournie.

4.3 Oracle

Le programme-maître travaille sur l'approximation linéaire (4.6) du problème original (4.1). A l'itération k les ensembles I et J sont constitués de tous les points y^{k-1} envoyés par le programme-maître depuis la première itération.

A une itération donnée k, le programme-maître envoie un point $y^k \in \{y : l \leq y \leq h\}$ à l'oracle et celui-ci renvoie une ou plusieurs coupes. On remarque que de nombreuses politiques peuvent être envisagées. Pour un point y^k l'oracle pourrait renvoyer toutes les coupes d'optimalité et toutes les coupes de réalisabilité. Cette politique génère un grand nombre de coupes. Inversement, il pourrait ne renvoyer qu'une coupe d'optimalité si $y^k \in Y$ et qu'une coupe de réalisabilité si $y^k \notin Y$. Entre ces deux politiques extrêmes de nombreuses autres peuvent être envisagées. Dans un premier temps, par esprit de simplification, nous ferons l'hypothèse que la deuxième politique est appliquée.

Dans l'algorithme ACCPM, l'oracle est traité comme une boîte noire qui renvoie l'une ou l'autre coupe. Le programme-maître traite ces coupes sans aucune référence à la façon dont elles ont été générées. Pour renforcer le caractère général de la méthode, nous nous concentrerons uniquement sur le programme-maître dans les sections qui suivent. Les oracles seront décrits dans le dernier chapitre lors de la description de problèmes spécifiques.

4.4 Ensemble de localisation et centre analytique

Le programme-maître envoie à l'oracle un point y^k appartenant à un ensemble appelé, **ensemble de localisation**. Pour que l'algorithme puisse converger, cet ensemble doit contenir une solution optimale. Dans les sections qui vont suivre nous allons voir comment définir cet ensemble et comment calculer un point y appartenant.

4.4.1 Ensemble de localisation : définition

Tout d'abord, remarquons qu'il est possible de calculer une borne supérieure $\overline{z}$ de la solution optimale z^* du problème original (4.1). Il suffit de prendre pour $\overline{z}$ la plus petite des valeurs de f aux différents points $\{y^i\}_{i\in I}$, c'est-à-dire[1] :

$$\overline{z} = \min_{i\in I} f(y^i).$$

L'ensemble de localisation $\mathcal{L}_{\overline{z}}$ est alors défini ainsi :

$$\begin{aligned}\mathcal{L}_{\overline{z}} = \quad & \{(y,z) \in \mathbb{R}^{n+1} : \quad z \leq \overline{z}, \quad z \geq f(y, y^i), \quad g(y, y^j) \leq 0, \\ & \qquad\qquad\qquad\qquad l \leq y \leq h, \quad \forall i \in I, j \in J\}.\end{aligned}$$

Remarquons que $\mathcal{L}_{\overline{z}}$ est un sous-ensemble borné de l'épigraphe de $\hat{f}(y)$. De plus, pour toute solution optimale z^* du problème (4.1) nous avons l'inégalité :

$$z^* = f(y^*) \geq \min\{\hat{f}(y) : y \in \hat{Y}, l \leq y \leq h\}.$$

Le polytope $\mathcal{L}_{\overline{z}}$ contient donc nécessairement la solution optimale (y^*, z^*).

4.4.2 Centre analytique : définition

L'algorithme ACCPM repose sur la détermination d'un point "central" y^k, appelé **centre analytique**, qui doit appartenir à l'ensemble de localisation $\mathcal{L}_{\overline{z}}$.

Définissons tout d'abord la matrice Ω_J^T et le vecteur γ_J obtenus par les coupes de réalisabilité. La matrice Ω_J^T est composée de l'ensemble des vecteurs $-\omega^T$ et le vecteur γ_J des scalaires $-\gamma$.

Définissons de même, la matrice Ω_I^T et le vecteur γ_I obtenus par les coupes d'optimalité. La matrice Ω_I^T est composée des sous-gradients $-\xi^T$ et le vecteur γ_I des scalaires $f(y^i) - \xi^T f(y^i)$.

[1] Afin d'alléger les notations, on écrit ici et par la suite : $i \in I$ pour $y_i \in I$. De même on écrira $j \in J$ pour $y_j \in J$.

Le problème (4.6) peut alors se récrire :

$$\begin{aligned} \text{Minimiser} \quad & z \\ \text{s.c.} \quad & \Omega_J^T y \geq \gamma_J \\ & \Omega_I^T y + z\mathbf{1} \geq \gamma_I \\ & l \leq y \leq h, \end{aligned} \tag{4.7}$$

où $\mathbf{1}$ est le vecteur unitaire de dimension appropriée.

Le dual de ce problème est :

$$\begin{aligned} \text{Maximiser} \quad & l^T\zeta - h^T\eta + \gamma_J^T\varrho + \gamma_I^T\varsigma \\ \text{s.c.} \quad & \zeta - \eta + \Omega_J\varrho + \Omega_I\varsigma = 0 \\ & \mathbf{1}^T\varsigma = 1 \\ & \zeta, \eta, \varrho, \varsigma \geq 0. \end{aligned} \tag{4.8}$$

Si l'on introduit les notations,

$$x = \begin{pmatrix} \zeta \\ \eta \\ \varrho \\ \varsigma \end{pmatrix}, \quad c = \begin{pmatrix} l \\ -h \\ \gamma_J \\ \gamma_I \end{pmatrix}, \quad b = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix}, \quad G = \begin{pmatrix} I_d & -I_d & \Omega_J & \Omega_I \\ 0 & 0 & 0 & \mathbf{1}^T \end{pmatrix},$$

où I_d est la matrice unitaire, en définissant le vecteur $u = \binom{y}{z}$ et $s = G^T u - c$, le problème (4.8) peut être alors reformulé de la façon suivante :

$$\begin{aligned} \text{Maximiser} \quad & c^T x \\ \text{s.c.} \quad & Gx = b \\ & x \geq 0. \end{aligned} \tag{4.9}$$

Son dual s'écrit alors :

$$\begin{aligned} \text{Minimiser} \quad & b^T u \\ \text{s.c.} \quad & G^T u - s = c \\ & s \geq 0. \end{aligned} \tag{4.10}$$

Dans la littérature sur les méthodes de points intérieurs le problème (4.9) est identifié comme le problème primal et (4.10) comme le problème dual. Par la suite nous ferons souvent référence à (4.9) comme le problème $\mathcal{P}$ et à (4.10) comme le problème $\mathcal{D}$.

Enfin, définissons les domaines suivants :

$$\begin{aligned} S_{\mathcal{P}} &= \{x : Gx = b, \quad x > 0\}, \\ S_{\bar{\mathcal{D}}} &= \{u : G^T u > c\}, \\ S_{\mathcal{D}} &= \{u : G^T u > c, \quad b^T u < \overline{z}\}, \end{aligned}$$

en posant l'hypothèse qu'ils ne sont pas vides. Cette hypothèse est classique dans la littérature sur les points intérieurs [9]. On fait de plus l'hypothèse que l'optimum n'a pas été déterminé et donc que $\overline{z}$ est une borne supérieure stricte.

L'ensemble de localisation s'écrit alors :

$$\mathcal{L}_{\overline{z}} = \{u : \quad b^T u \leq \overline{z}, \quad G^T u \geq c\},$$

On remarquera que si on n'est pas à l'optimum, il existe un point u appartenant à l'intérieur de $\mathcal{L}_{\overline{z}}$. En effet, comme $S_{\bar{\mathcal{D}}} \neq \emptyset$, s'il n'existait pas de point intérieur, $b^T u = z \geq \overline{z}$ et $\overline{z}$ ne serait pas une borne supérieure stricte.

Soit κ le nombre de colonnes de la matrice G, le centre analytique de $\mathcal{L}_{\overline{z}}$ est défini comme le point qui minimise la fonction potentielle suggérée par Renegar [7] et Sonnevend [10] :

$$\begin{aligned} \text{Minimiser} \quad & \psi(u,s) = -\ln(\overline{z} - b^T u) - \sum_{j=1}^{\kappa} \ln s_j \\ \text{s.c.} \quad & G^T u - s = c, \qquad s > 0 \\ & \overline{z} - b^T u > 0. \end{aligned} \tag{4.11}$$

En effet, de par les contraintes de ce problème, le point va appartenir à l'intérieur de l'ensemble de localisation $\mathcal{L}_{\overline{z}}$. De plus, en minimisant la fonction objectif, le point va chercher à s'éloigner des frontières de ce domaine. L'écart avec les contraintes étant mesuré par la valeur de la variable d'écart associée, ce point n'est pas un centre de l'ensemble de localisation au sens géométrique du terme. C'est pourquoi il a été dénommé centre analytique.

En définissant s_0, la variable mesurant l'écart avec la borne supérieure, $\tilde{s} = (s_0, s)^T$ et $\tilde{\lambda} = (\lambda_0, \lambda)^T$, le Lagrangien du problème (4.11) s'écrit :

$$\mathcal{L}(u, \tilde{s}, \tilde{\lambda}) = -\ln s_0 - \sum_{j=1}^{\kappa} \ln s_j + \lambda_0(s_0 + b^T u - \overline{z}) + \lambda^T(s - G^T u + c).$$

Ses dérivées partielles sont :

$$
\begin{aligned}
\frac{\partial \mathcal{L}(u, \tilde{s}, \tilde{\lambda})}{\partial u} &= \lambda_0 b - G\lambda, \\
\frac{\partial \mathcal{L}(u, \tilde{s}, \tilde{\lambda})}{\partial s_0} &= -s_0^{-1} + \lambda_0, \\
\frac{\partial \mathcal{L}(u, \tilde{s}, \tilde{\lambda})}{\partial s} &= -S^{-1}\mathbf{1} + \lambda, \\
\frac{\partial \mathcal{L}(u, \tilde{s}, \tilde{\lambda})}{\partial \lambda_0} &= s_0 + b^T u - \overline{z}, \\
\frac{\partial \mathcal{L}(u, \tilde{s}, \tilde{\lambda})}{\partial \lambda} &= -G^T u + s + c,
\end{aligned}
$$

où S est la matrice diagonale dont les éléments sont les composantes du vecteur s.

Le problème (4.11) étant composé de la somme de fonctions logarithmiques affectées d'un signe négatif, l'objectif est donc convexe. De plus, il est défini en tout point de son ensemble de définition. Les conditions de Karush-Kuhn-Tucker sont donc des conditions nécessaires et suffisantes d'optimalité [7], [10] et [8].

En définissant $x = \lambda/\lambda_0$, les **conditions d'optimalité du premier ordre** sont :

$$
\begin{aligned}
Gx &= b, & x &> 0 \\
G^T u - s &= c, & s &> 0 \\
b^T u + s_0 &= \overline{z}, & s_0 &> 0 \\
Xs &= s_0 \mathbf{1}, & &
\end{aligned}
\tag{4.12}
$$

où X est la matrice diagonale dont les éléments sont les composantes du vecteur x.

Sous l'hypothèse qu'il existe un point appartenant à l'intérieur de $\mathcal{L}_{\overline{z}}$, (4.12) admet toujours une solution et cette solution est unique. Nous verrons dans le chapitre suivant comment calculer ce point.

4.5 Algorithme ACCPM théorique

Comme nous l'avons vu, l'algorithme est divisé en deux parties. La première dénommée oracle, qui génère les coupes, et la seconde nommée programme-maître qui calcule un centre analytique $u^c = (y^c, z^c)$ de l'ensemble de localisation courant $\mathcal{L}_{\overline{z}}$ en résolvant (4.11).

Nous verrons par la suite, qu'il est possible au programme-maître de calculer une borne inférieure $\underline{z}$. Comme d'autre part nous possédons une borne supérieure, on peut définir un saut de dualité $\overline{z} - \underline{z}$, critère d'arrêt de cette procédure itérative.

L'algorithme ACCPM de génération de plans coupants aux centres analytiques peut être formulé de la manière suivante :

Algorithme ACCPM

Paramètres :

Définir un niveau de tolérance pour l'objectif : $\varepsilon \geq 0$;

Définir des bornes sur y : l et $h \in \mathbb{R}^n$;

Initialisation :

Borne supérieure de la valeur de la solution : $\overline{z}$;

Borne inférieure de la valeur de la solution : $\underline{z}$;

Ensemble de localisation : $\mathcal{L}_{\overline{z}} = \{(y, z) \in \mathbb{R}^{n+1} : \ z \leq \overline{z}, \ z \geq \underline{z}, \ \ l \leq y \leq h\}$;

Centre analytique de $\mathcal{L}_{\overline{z}}$: $(y^c, z^c) = \left(\frac{l+h}{2}, \frac{\underline{z}+\overline{z}}{2}\right)$;

Tant que $\overline{z} - \underline{z} > \varepsilon$, **faire**

oracle :

Si $y^c \in Y$, **alors**

Envoyer la coupe d'optimalié : $z \geq f(y; y^c)$;

$\overline{z} = \min\{\overline{z}, f(y^c)\}$;

Sinon

Envoyer la coupe de réalisabilité : $g(y; y^c) \leq 0$;

Fsi ;

programme-maître :

Mettre à jour l'ensemble de localisation ;

Déterminer le nouveau centre analytique : (y^c, z^c) ;

Calculer d'une borne inférieure θ ;

$\underline{z} = \max\{\underline{z}, \theta\}$;

Fin.

4.6 Résultats de convergence

L'étude de la convergence de algorithme ACCPM tel qu'il a été implanté n'est pas complète. Bien que, récemment, quelques résultats aient été obtenus, ils ne peuvent être utilisés que si l'on fait quelques restrictions.

Les études qui vont suivre se basent sur les travaux de Atkinson et Vaiday [1] et de Nesterov [6]. Elles les complètent, c'est pourquoi nous ne les mensionnerons pas plus en détail et laisserons le lecteur s'y référer pour plus d'information.

Pour pouvoir cerner la convergence de l'algorithme, les récents travaux font quelques hypothèses restrictives. Par exemple, Kiwiel [3] et Altman et Kiwiel [2] se limitent à l'étude de ACCPM lorsqu'il minimise une fonction convexe non contrainte. Goffin, Luo et Ye [5] et Ye [11] s'intéressent au comportement d'un algorithme voisin lorsqu'il est appliqué au problème de réalisabilité convexe. On notera que si la fonction est continue au sens de Lipschitz ce dernier problème est équivalent à celui de la minimisation d'une fonction convexe sur un ensemble convexe. Pour cela, il suffit de prendre comme oracle dans le problème de réalisabilité : tant que la borne inférieure n'est pas comprise dans une boule de rayon ε, envoyer une coupe.

Dans l'étude théorique que nous avons développée jusqu'a présent, nous n'avons pas abordé les calculs permettant de déterminer un centre analytique. Nous verrons dans le chapitre suivant que son calcul exact n'est pas nécessaire et donc que seul un centre analytique "approché" sera envoyé à l'oracle. D'un point de vue pratique la différence est importante puisqu'elle permet de diminuer de manière drastique l'effort de calcul pour déterminer ce centre. Malheureusement, ceci fait aussi l'objet d'une restriction dans les études théoriques. Par exemple, Kiwiel [3] et Altman et Kiwiel [2] exigent qu'un centre analytique exact soit calculé. De même, les résultats de Ye [11] se basent sur le calcul d'un centre analytique exact. Ces derniers cependant peuvent être généralisés lorsqu'un centre analytique approché est calculé. Seuls Goffin, Luo et Ye [5] restent dans un cadre général avec le calcul d'un centre analytique approché.

Au chapitre 7 nous verrons que des tests d'élimination peuvent être implantés. Ces tests permettent de supprimer des coupes. Ils diminuent alors la taille de la matrice G et facilitent ainsi les calculs. Ces tests ne sont, cependant, jamais mentionnés dans les études théoriques.

En conclusion, aucune preuve de convergence n'a été trouvée lorsque, pour le problème de base (4.1), l'algorithme ACCPM dans son cadre le plus général est utilisé, c'est-à-dire avec le calcul d'un centre analytique approché et en appliquant des tests d'élimination.

Avant de donner les résultats obtenus rappelons en quoi consistent les différents problèmes traités. Le problème de la minimisation d'une fonction convexe est équivalent au problème de base (4.1) lorsque $Y = \mathbb{R}^n$. Le problème de réalisabilité convexe consiste à touver un point appartenant à l'intérieur d'un ensemble convexe Y. Si la valeur de l'optimum z^* est connu, le problème de base (4.1) se transforme aisément en un problème de réalisabilité convexe. En effet, le but est alors de trouver un point y appartenant à l'ensemble $\tilde{Y} = \{y : f(y) \leq z^*, y \in Y\}$.

Les résultats de convergence sont les suivants :

- Tout d'abord, Goffin, Luo et Ye [5] ont montré qu'une version voisine de l'algorithme ACCPM tel que nous l'avons implanté, est pseudo-polynomiale lorsqu'elle est appliquée au problème de réalisabilité convexe où $Y \subseteq \{y : 0 \leq y \leq \mathbf{1}\}$. La complexité en nombre d'appels à l'oracle est :

$$O^*\left(\frac{n^2}{\varepsilon^2}\right),$$

  où $\varepsilon < 1/2$ est le rayon d'une boule contenue dans Y et où O^* est une notation des auteurs qui signifie que seule une estimation du nombre d'itérations a pu être donnée, les termes d'ordre inférieur ayant été négligés.

  Ils montrent de plus que sous certaines conditions de centrage[2] quatre itérations sont nécessaires pour déterminer un nouveau centre analytique. La complexité globale de l'algorithme est donc $O^*(n^2/\varepsilon^2)$.

  On remarquera tout d'abord que le problème plus général où $Y \supseteq \{y : l \leq y \leq h\}$ peut se récrire aisément comme le problème traité. Nous avons vu de plus que ce problème est équivalent au problème original (4.1). Analysons en quoi la version des auteurs diffère de la nôtre.

  1. d'une part elle introduit les coupes au centre analytique approché (coupes moins profondes). Les coupes sont donc du type $\omega^T y \leq \omega^T y^c$.
  2. d'autre part elle travaille directement sur la matrice $\hat{G} = \begin{pmatrix} \overline{z} & -c^T \\ -b & G \end{pmatrix}$. Bien que cela soit strictement équivalent pour le calcul du centre analytique approché nous verrons que, lorsque l'on ajoute une contrainte, cela assure que le produit $\overline{z} - c^T x$ reste constant.

  En résumé cette version de l'algorithme ACCPM diffère assez peu de la nôtre.

[2] La condition s'écrit $\theta \leq 0.09$, avec θ le paramètre de centrage.

- Ye [11] généralise ce résultat lorsque η coupes sont générées à chaque itération. Il restreint cependant son étude au cas où l'algorithme calcule un centre analytique exact. Il montre alors que, pour le même problème de réalisabilité que ci-dessus, la complexité globale de l'algorithme est :

$$O^*\left(\frac{\eta^2 n^2}{\varepsilon^2}\right).$$

- Kiwiel [3] traite la minimisation d'une fonction convexe par l'algorithme ACCPM lorsqu'un centre analytique exact est calculé. Plus précisément il cherche à résoudre le problème suivant :

$$z^* = \min\{f(y) : 0 \leq y \leq \mathbf{1}\},$$

  où $f : \mathbb{R}^n \to \mathbb{R}$ est convexe, $z^* \leq 0$ et $f(\frac{1}{2}\mathbf{1}) = 1$. Il montre que tout problème de minimisation d'une fonction convexe peut se récrire ainsi.

  De plus il apporte un variante à l'algorithme. Il considère que si $\overline{z} > f(y^c)$ aucune coupe n'est générée seule la valeur de $\overline{z}$ diminue. En s'appuyant sur des résultats de Goffin, Luo et Ye [5], il montre alors que la complexité en nombre de coupes introduites pour une précision de ε est :

$$O^*\left(\frac{nL^2}{\varepsilon^2}\right),$$

  où L est la constante de Lipschitz de la fonction f. Malheureusement ce résultat ne peut pas se généraliser si une coupe est introduite lorsque $\overline{z} > f(y^c)$.

Nous terminons ici ce rapide tour d'horizon sur les différentes estimations de la complexité de l'algorithme ACCPM.

Bibliographie

[1] D. S. Atkinson, P. M. Vaidya, A cutting plane algorithm that uses analytic centers, University of Illinois at Urbana–Champaign, Urbana, IL 61820, USA, June 1992, submitted to "Nondifferentiable and Large Scale Optimization", *Mathematical Programming, Series B*, J.-L. Goffin and J.-P. Vial, eds.

[2] A. Altman, K. C. Kiwiel, "A note on some analytic center cutting plane methods for convex feasibility and minimization problems", Technical Report (1994).

[3] K. C. Kewiel, "Complexity of some cutting plane methods that use analytic centers", Technical Report (1994).

[4] G. B. Dantzig, "Planning under uncertainty using parallel computing", *Annals of Operations Research* **14** (1988) 1-16.

[5] J. L. Goffin, Z. Luo et Y.Ye, "Complexity analysis of an interior cutting plane method for convex feasibility problems", manuscrit, McGill University, Canada (1993), To appear in *SIAM J. Optimization.*

[6] Y. Nesterov, Cutting plane algorithms from analytic centers : efficiency estimates, (1992) submitted to "Nondifferentiable and Large Scale Optimization", *Mathematical Programming, Series B*, J.-L. Goffin and J.-P. Vial, eds.

[7] J. Renegar, "A polynomial-time algorithm based on Newton's method for linear programming", *Mathematical Programming* **40** (1988) 59-94.

[8] C. Roos et J. P. Vial, "Analytic centers in linear programming", Report 88-68, Faculty of Technical Mathematics and Informatics, Technische Universiteit Delft, Holland (1988).

[9] C. Roos et J. P. Vial, "Interior-point methods for linear programming", In J. E. Beasley, editor, *Advances in Linear and Integer Programming* Oxford University Press, Oxford, Great Britain, 1995.

[10] G. Sonnevend, "New algorithms in convex programming based on a notion of "centre" (for systems of analytic inequalities) and on rational extrapolation", in K. H. Hoffmann, J. B. Hiriat-Urruty, C. Lemaréchal, and J. Zowe, editors, *Trends in Mathematical Optimization : Proceedings of the 4th French–German Conference on Optimization in Irsee, West-Germany, April 1986*, volume 84 of *International Series of Numerical Mathematics*, pages 311-327. Birkhäuser Verlag, Basel, Switzerland, 1988.

[11] Y. Ye, "Complexity analysis of the analytic center cutting plane methods that uses multiple cuts", Technical Report (1994).

Chapitre 5

Algorithme projectif : calcul du centre analytique

Dans le chapitre précédent nous avons vu que l'algorithme ACCPM est un algorithme de plans coupants se rattachant à la famille des méthodes des centres. Nous avons vu aussi qu'à chaque itération il est possible de définir un ensemble de localisation $\mathcal{L}_{\overline{z}}$ contenant une solution optimale. L'algorithme ACCPM calcule un point dans cet ensemble et l'envoie à l'oracle. Ce point, appelé centre analytique, a été défini au chapitre précédent comme solution du problème (4.11).

Bien qu'il soit possible de résoudre directement le problème (4.11), pour des raisons qui apparaîtront plus tard, nous avons procédé autrement. Dans la littérature sur les méthodes de points intérieurs, l'algorithme qui résoud directement ce problème est appellé algorithme projectif dual [4]. Dans ACCPM nous avons utilisé un algorithme qui est référencé comme l'algorithme projectif primal [1].

Dans ce chapitre nous allons décrire comment calculer le centre analytique et nous allons voir quelques unes de ces propriétés. Tout d'abord nous verrons l'approche primale et nous définirons un problème homogène équivalent. Ensuite nous verrons comment résoudre ce dernier par l'algorithme projectif primal (calcul de la direction de Newton, test d'arrêt pour les itérations et enfin algorithme projectif), en donnant un aperçu de la convergence de cet algorithme. Enfin, nous verrons comment définir des ellipsoïdes centrés au centre analytique et incluant ou englobant l'ensemble de localisation, et pour finir, nous verrons comment calculer un centre analytique pondéré, c'est-à-dire un centre analytique qui donne une importance inégale aux différentes coupes.

5.1 Approche primale

Le centre analytique du polytope $\mathcal{L}_{\overline{z}}$ peut, de manière équivalente, être défini comme le minimum, sur l'ensemble $S_{\mathcal{P}}$, de la **fonction potentielle de Karmarkar** [6]. La fonction à minimiser s'écrit :

$$\begin{aligned} \text{Minimiser} \quad & \varphi(x) = (\kappa+1)\ln(\overline{z} - c^T x) - \sum_{j=1}^{\kappa} \ln x_j \\ \text{s.c.} \quad & Gx = b \\ & x > 0. \end{aligned} \tag{5.1}$$

Théorème : 5.1 [5] [7] et [2]. *Les problèmes* $\min\{\psi(u,s) : \; u \in S_{\mathcal{D}}\}$ *et* $\min\{\varphi(x) : x \in S_{\mathcal{P}}\}$ *ont les mêmes conditions d'optimalité.*

Preuve. Le Lagrangien du problème (5.1) s'écrit de la manière suivante :

$$\mathcal{L}(x,\lambda) = (\kappa+1)\ln(\overline{z} - c^T x) - \sum_{j=1}^{\kappa} \ln x_j + \lambda^T (Gx - b).$$

Ses dérivées partielles sont :

$$\begin{aligned} \frac{\partial \mathcal{L}(x,\lambda)}{\partial x} &= \frac{-(\kappa+1)c}{\overline{z} - c^T x} - X^{-1}\mathbf{1} + G^T\lambda, \\ \frac{\partial \mathcal{L}(x,\lambda)}{\partial \lambda} &= Gx - b. \end{aligned}$$

Les conditions d'optimalité du premier ordre du problème (5.1) ci-dessus sont donc :

$$\begin{aligned} G^T\lambda &= \frac{\kappa+1}{\overline{z} - c^T x} c + X^{-1}\mathbf{1} \\ Gx &= b. \end{aligned}$$

En posant :

$$\begin{aligned} \lambda &= \frac{\kappa+1}{\overline{z} - c^T x} u \\ G^T u &= c + s, \qquad\qquad s > 0, \end{aligned}$$

la première condition d'optimalité se récrit :

$$\begin{aligned} \frac{\kappa+1}{\overline{z} - c^T x} c + X^{-1}\mathbf{1} &= \frac{\kappa+1}{\overline{z} - c^T x} G^T u \\ &= \frac{\kappa+1}{\overline{z} - c^T x} (c+s). \end{aligned}$$

On a donc :

$$X^{-1}\mathbf{1} = \frac{\kappa+1}{\overline{z}-c^Tx}s,$$

et par suite :

$$\begin{aligned} Xs &= \frac{\overline{z}-c^Tx}{\kappa+1}\mathbf{1} \\ &= \frac{\overline{z}-b^Tu+b^Tu-c^Tx}{\kappa+1}\mathbf{1}. \end{aligned} \tag{5.2}$$

Pour retrouver la dernière condition d'optimalité du problème (4.11), il reste à prouver que $b^Tu-c^Tx = \kappa(\overline{z}-b^Tu)$. Remarquons tout d'abord que $x^Ts = b^Tu-c^Tx$. En effet, comme $G^Tu-s=c$ et $G^Tx=b$, $c^Tx = (G^Tu-s)^Tx = u^TGx - s^Tx = u^Tb - s^Tx$. Nous avons alors :

$$\begin{aligned} Xs &= \frac{\overline{z}-b^Tu+b^Tu-c^Tx}{\kappa+1}\mathbf{1} \\ &= \frac{\overline{z}-b^Tu+x^Ts}{\kappa+1}\mathbf{1}. \end{aligned}$$

Par suite nous avons :

$$\begin{aligned} x^Ts &= \frac{\overline{z}-b^Tu+x^Ts}{\kappa+1}\mathbf{1}^T\mathbf{1} \\ &= \frac{\kappa}{\kappa+1}(\overline{z}-b^Tu+x^Ts). \end{aligned}$$

Soit :

$$x^Ts(1-\frac{\kappa}{\kappa+1}) = \frac{\kappa}{\kappa+1}(\overline{z}-b^Tu).$$

On en conclut :

$$x^Ts = \kappa(\overline{z}-b^Tu).$$

En utilisant cette dernière équation dans l'expression (5.2) de Xs on obtient :

$$\begin{aligned} Xs &= \frac{\overline{z}-b^Tu+b^Tu-c^Tx}{\kappa+1}\mathbf{1} \\ &= \frac{\overline{z}-b^Tu+x^Ts}{\kappa+1}\mathbf{1} \\ &= \frac{\overline{z}-b^Tu+\kappa(\overline{z}-b^Tu)}{\kappa+1}\mathbf{1} \\ &= (\overline{z}-b^Tu)\mathbf{1}. \end{aligned}$$

En posant $b^T u + s_0 = \overline{z}$ avec $s_0 > 0$, les conditions d'optimalité du problème (5.1) peuvent donc s'écrire :

$$\begin{aligned} Gx &= b, & x &> 0 \\ G^T u - s &= c, & s &> 0 \\ b^T u + s_0 &= \overline{z}, & s_0 &> 0 \\ Xs &= s_0 \mathbf{1}. \end{aligned}$$

On retrouve les conditions d'optimalité du problème (4.11). ■

5.2 Problèmes homogènes équivalents

Soit $\tilde{G} = (-b \; G)$, $\tilde{c} = \binom{\overline{z}}{-c}$, et $\tilde{x} = \binom{x_0}{x}$, définissons le **problème homogène** suivant opérant sur un espace élargi de dimension $\kappa + 1$:

$$\begin{aligned} \text{Minimiser} \quad & \tilde{\varphi}(\tilde{x}) = (\kappa+1) \ln \tilde{c}^T \tilde{x} - \sum_{j=0}^{\kappa} \ln \tilde{x}_j \\ \text{s.c.} \quad & \tilde{G}\tilde{x} = 0, \\ & \tilde{x} > 0. \end{aligned} \tag{5.3}$$

Théorème : 5.2 [3]. *Si $x_0 = 1$, les problèmes (5.1) $\min\{\varphi(x) : \; Gx = b, \; x > 0\}$ et (5.3) $\min\{\tilde{\varphi}(\tilde{x}) : \; \tilde{G}\tilde{x} = 0, \; \tilde{x} > 0\}$ sont équivalents.*

Preuve. Tout d'abord, remarquons que $\tilde{\varphi}(\tilde{x})$ est homogène de degré 0, c'est-à-dire $\forall \tilde{x} > 0$ et $t > 0$, $\tilde{\varphi}(t\tilde{x}) = \tilde{\varphi}(\tilde{x})$. En effet, $\forall \tilde{x} > 0$ et $t > 0$:

$$\begin{aligned} \tilde{\varphi}(t\tilde{x}) &= (\kappa+1) \ln(t\tilde{c}^T \tilde{x}) - \sum_{j=0}^{\kappa} \ln(t\tilde{x}_j) \\ &= (\kappa+1)[\ln t + \ln \tilde{c}^T \tilde{x}] - \sum_{j=0}^{\kappa} [\ln t + \ln \tilde{x}_j] \\ &= (\kappa+1) \ln \tilde{c}^T \tilde{x} - \sum_{j=0}^{\kappa} \ln \tilde{x}_j \\ &= \tilde{\varphi}(\tilde{x}). \end{aligned}$$

De plus, si $\tilde{x}$ satisfait les contraintes, pour tout $t > 0$, $t\tilde{x}$ les satisfait aussi. En effet, $\tilde{G}t\tilde{x} = t\tilde{G}\tilde{x} = 0$ et $t\tilde{x} > 0$. En prenant $t = 1/\tilde{x}_0$ les problèmes (5.3) et (5.1) sont équivalents. ■

Soit $\tilde{x}^k$ un point tel que $\tilde{G}\tilde{x}^k = 0$ et $\tilde{x}^k > 0$, définissons le problème suivant :

$$\min_{\tilde{p}\in\mathbb{R}^{\kappa+1}} \{\tilde{\varphi}(\tilde{x}^k + \tilde{p}) : \quad \tilde{G}\tilde{p} = 0, \quad \tilde{c}^T\tilde{p} = 0, \quad \tilde{x}^k + \tilde{p} > 0\} \tag{5.4}$$

Théorème : 5.3 [8]. *Les problèmes (5.3) et (5.4) sont équivalents.*

Preuve. Pour commencer, remarquons que si $\tilde{x}^k$ et $\tilde{p}$ satisfont les contraintes du problène (5.4), $\tilde{x} = \tilde{x}^k + \tilde{p}$ satisfait les contraintes du problème (5.3). Nous avons en effet : $\tilde{G}\tilde{x} = \tilde{G}(\tilde{x}^k + \tilde{p}) = \tilde{G}\tilde{x}^k + \tilde{G}\tilde{p} = 0 + 0 = 0$.

Montrons maintenant qu'à tout point $\tilde{x}$ solution réalisable du problème (5.3) on peut associer une solution réalisable pour (5.4). Par définition, le vecteur $\tilde{x}$ satisfait les équations $\tilde{G}\tilde{x} = 0$ et $\tilde{x} > 0$. Le vecteur $\tilde{x}/\tilde{x}_0$ vérifie aussi ces conditions mais il est de plus solution réalisable du problème (4.9) appelé problème $\mathcal{P}$. Comme $\overline{z}$ est une borne supérieure stricte[1] et $x_0 > 0$ on déduit :

$$\tilde{c}^T\tilde{x} = \overline{z}x_0 - c^T x > 0.$$

Comme $\tilde{x}^k$ satisfait aussi les hypothèses $\tilde{G}\tilde{x}^k = 0$ et $x^k > 0$, on obtient $\tilde{c}^T\tilde{x}^k > 0$.

Soit $\tilde{x}$ une solution de (5.3). Comme $\tilde{c}^T\tilde{x} > 0$ définissons $t = \frac{\tilde{c}^T\tilde{x}^k}{\tilde{c}^T\tilde{x}} > 0$. Le vecteur $\tilde{p}$ défini par $\tilde{p} = t\tilde{x} - \tilde{x}^k$ satisfait les contraintes du problème (5.4). En effet,

$$\begin{aligned}
\tilde{G}\tilde{p} &= t\tilde{G}\tilde{x} - \tilde{G}\tilde{x}^k = 0 \\
\tilde{c}^T\tilde{p} &= t\tilde{c}^T\tilde{x} - \tilde{c}^T\tilde{x}^k = \frac{\tilde{c}^T\tilde{x}^k}{\tilde{c}^T\tilde{x}}\tilde{c}^T\tilde{x} - \tilde{c}^T\tilde{x}^k = 0,
\end{aligned}$$

et

$$\tilde{x}^k + \tilde{p} = t\tilde{x} > 0.$$

Nous avons donc montré qu'à tout point $\tilde{x}$ solution du problème (5.3) peut être associé une solution réalisable pour (5.4) : $\tilde{p} = t\tilde{x} - \tilde{x}^k$.

Le problème (5.4) est plus contraint que le problème (5.3). L'ensemble sur lequel est défini $\tilde{\varphi}(.)$ est donc plus restreint pour ce nouveau problème. La fonction à minimiser étant la même, il reste à prouver qu'il est possible de trouver une direction $\tilde{p}$ donnant le même minimum. Soit $\tilde{x}^*$ le minimum du problème (5.3), nous avons vu que le vecteur $\tilde{p} = t\tilde{x}^* - \tilde{x}^k$ est réalisable pour le problème (5.4). Comme $\tilde{\varphi}(\tilde{x}^k + \tilde{p}) = \tilde{\varphi}(t\tilde{x}^*) = \tilde{\varphi}(\tilde{x}^*)$ la direction $\tilde{p}$ donne bien le même minimum pour (5.4). ■

[1] Nous avons montré que l'intérieur de $\mathcal{L}_{\overline{z}}$ est non vide (voir page 85).

Théorème : 5.4 [8]. *La fonction* $\tilde{\varphi}(\tilde{x}^k + \tilde{p})$ *est convexe en* $\tilde{p}$ *sur son ensemble de définition :* $\{\tilde{p} : \quad \tilde{G}\tilde{p} = 0, \quad \tilde{c}^T\tilde{p} = 0, \quad \tilde{x}^k + \tilde{p} > 0\}$.

Preuve. Comme $\tilde{c}^T\tilde{p} = 0$,

$$\begin{aligned}
\tilde{\varphi}(\tilde{x}^k + \tilde{p}) &= (\kappa+1)\ln(\tilde{c}^T(\tilde{x}^k + \tilde{p})) - \sum_{j=0}^{\kappa}\ln(\tilde{x}_j + \tilde{p}_j) \\
&= (\kappa+1)\ln(\tilde{c}^T\tilde{x}^k + \tilde{c}^T\tilde{p}) - \sum_{j=0}^{\kappa}\ln(\tilde{x}_j + \tilde{p}_j) \\
&= (\kappa+1)\ln(\tilde{c}^T\tilde{x}^k) - \sum_{j=0}^{\kappa}\ln(\tilde{x}_j + \tilde{p}_j).
\end{aligned}$$

Le point x^k étant fixé, $(\kappa+1)\ln(\tilde{c}^T\tilde{x}^k)$ est constant. Comme $-\sum_{j=0}^{\kappa}\ln(\tilde{x}_j + \tilde{p}_j)$ est convexe en $\tilde{p}$, $\tilde{\varphi}(\tilde{x}^k + \tilde{p})$ est convexe sur son ensemble de définition. ∎

En résumé, nous avons montré que le calcul du centre analytique défini initialement par le problème (4.11) peut se faire de manière équivalente dans l'espace primal par la résolution du problème (5.1). Ce problème n'est pas nécessairement convexe en x, cependant nous avons montré qu'il est équivalent au problème homogène (5.4) dans $\mathbb{R}^{\kappa+1}$ et que ce dernier est convexe en $\tilde{p}$. Par la suite, nous appellerons **espace projectif** l'espace $\mathbb{R}^{\kappa+1}$.

5.3 Direction de Newton projetée

Le problème (5.4) permet de déterminer le centre analytique recherché. En effet, nous avons montré dans le paragraphe précédent que ce problème est équivalent au problème initial (4.11) qui définit le centre analytique comme un point "central" de l'ensemble de localisation $\mathcal{L}_{\overline{z}}$.

Le problème (5.4) consiste à minimiser une fonction convexe $\tilde{\varphi}(\tilde{x}^k + \tilde{p})$ sur son ensemble de définition. Le point $\tilde{x}^k$ est fixé et satisfait $\tilde{G}\tilde{x}^k = 0$ et $\tilde{x}^k > 0$. L'optimisation du problème (5.4) se fait sur $\tilde{p}$ et non sur $\tilde{x}^k$. En faisant le changement de variable $\tilde{x} = \tilde{x}^k + \tilde{p}$ le problème (5.4) se récrit :

$$\min_{\tilde{x}\in\mathbb{R}^{\kappa+1}} \{\tilde{\varphi}(\tilde{x}) : \quad \tilde{G}\tilde{x} = 0, \tilde{c}^T\tilde{x} = \sigma, \ \tilde{x} > 0\}, \tag{5.5}$$

avec $\tilde{c}^T\tilde{x}^k = \sigma$.

Pour résoudre ce problème, nous allons utiliser la méthode de Newton dans l'espace projectif. Commençons par déterminer sa direction. L'algorithme de Newton consiste à minimiser l'approximation quadratique de la fonction $\tilde{\varphi}(\tilde{x})$ au point $\tilde{x}$. Le développement de Taylor du deuxième ordre de $\tilde{\varphi}(\tilde{x})$ est :

$$\tilde{\varphi}(\tilde{x}+\tilde{d}) = \tilde{\varphi}(\tilde{x}) + \tilde{d}^T \nabla\tilde{\varphi}(\tilde{x}) + \frac{1}{2}\tilde{d}^T \nabla^2\tilde{\varphi}(\tilde{x})\tilde{d} + O(\|d\|).$$

Comme les contraintes du problèmes deviennent :

$$\begin{aligned} \tilde{G}(\tilde{x}+\tilde{d}) &= \tilde{G}\tilde{x} + \tilde{G}\tilde{d} = \tilde{G}\tilde{d} \\ \tilde{c}^T(\tilde{x}+\tilde{d}) &= \tilde{c}^T\tilde{x} + \tilde{c}^T\tilde{d} = \sigma + \tilde{c}^T\tilde{d}, \end{aligned}$$

la direction de Newton $\tilde{d}$ est donc donnée par la résolution du problème suivant :

$$\begin{aligned} \text{Minimiser} \quad & \tilde{d}^T \nabla\tilde{\varphi}(\tilde{x}) + \tfrac{1}{2}\tilde{d}^T \nabla^2\tilde{\varphi}(\tilde{x})\tilde{d} \\ \text{s.c.} \quad & \tilde{G}\tilde{d} = 0 \\ & \tilde{c}^T\tilde{d} = 0 \\ & \tilde{x} + \tilde{d} > 0. \end{aligned} \tag{5.6}$$

En substituant $\nabla\tilde{\varphi}(\tilde{x})$ et $\nabla^2\tilde{\varphi}(\tilde{x})$ par leur valeur respective, à savoir :

$$\begin{aligned} \nabla\tilde{\varphi}(\tilde{x}) &= \frac{\kappa+1}{\tilde{c}^T\tilde{x}}\tilde{c} - \tilde{X}^{-1}\mathbf{1}, \\ \nabla^2\tilde{\varphi}(\tilde{x}) &= -\frac{\kappa+1}{(\tilde{c}^T\tilde{x})^2}\tilde{c}\tilde{c}^T + \tilde{X}^{-2}, \end{aligned}$$

le problème (5.6) s'écrit :

$$\begin{aligned} \text{Minimiser} \quad & \tilde{d}^T\left(\tfrac{\kappa+1}{\tilde{c}^T\tilde{x}}\tilde{c} - \tilde{X}^{-1}\mathbf{1}\right) + \tfrac{1}{2}\tilde{d}^T\left(-\tfrac{\kappa+1}{(\tilde{c}^T\tilde{x})^2}\tilde{c}\tilde{c}^T + \tilde{X}^{-2}\right)\tilde{d} \\ \text{s.c.} \quad & \tilde{G}\tilde{d} = 0 \\ & \tilde{c}^T\tilde{d} = 0 \\ & \tilde{x} + \tilde{d} > 0. \end{aligned}$$

Comme $\tilde{c}^T\tilde{d} = 0$, ce dernier problème se simplifie et s'écrit :

$$\begin{aligned} \text{Minimiser} \quad & -\tilde{d}^T\tilde{X}^{-1}\mathbf{1} + \tfrac{1}{2}\tilde{d}^T\tilde{X}^{-2}\tilde{d} \\ \text{s.c.} \quad & \tilde{G}\tilde{d} = 0 \\ & \tilde{c}^T\tilde{d} = 0 \\ & \tilde{x} + \tilde{d} > 0. \end{aligned}$$

Les conditions d'optimalité du premier ordre sont les suivantes :

$$\begin{aligned}\tilde{X}^{-2}\tilde{d} - \tilde{X}^{-1}\mathbf{1} + \lambda_0\tilde{c} + \tilde{G}^T\lambda &= 0\\ \tilde{G}\tilde{d} &= 0\\ \tilde{c}^T\tilde{d} &= 0,\end{aligned}$$

où (λ_0, λ) sont les multiplicateurs de Lagrange. En définissant le vecteur $\tilde{q}$ par $\tilde{q} = \tilde{X}^{-1}\tilde{d}$ et en prémultipliant par $\tilde{X}$ la première des conditions d'optimalité, ces équations se récrivent :

$$\begin{aligned}\mathbf{1} &= \tilde{q} + (\tilde{X}\tilde{c}, \tilde{X}\tilde{G}^T)\binom{\lambda_0}{\lambda}\\ \binom{\tilde{c}^T\tilde{X}}{\tilde{G}\tilde{X}}\tilde{q} &= 0.\end{aligned} \tag{5.7}$$

La deuxième équation indique que $\tilde{q}$ appartient au noyau de $\binom{\tilde{c}^T\tilde{X}}{\tilde{G}\tilde{X}}$ noté $\mathcal{N}\binom{\tilde{c}^T\tilde{X}}{\tilde{G}\tilde{X}}$. Par définition, il est orthogonal à l'espace image de $(\tilde{X}\tilde{c}, \tilde{X}\tilde{G}^T)$. Par contre, le vecteur $\tilde{r} = (\tilde{X}\tilde{c}, \tilde{X}\tilde{G}^T)\binom{\lambda_0}{\lambda}$ appartient à l'espace image de $(\tilde{X}\tilde{c}, \tilde{X}\tilde{G}^T)$. On remarque donc que $\mathbf{1}$ peut se décomposer en deux vecteurs complémentaires et orthogonaux :

$$\mathbf{1} = \tilde{q} + \tilde{r}. \tag{5.8}$$

Par conséquent, le vecteur $\tilde{q}$ peut être interprété comme la projection orthogonale de $\mathbf{1}$ dans l'espace nul de $\binom{\tilde{c}^T\tilde{X}}{\tilde{G}\tilde{X}}$ (voir figure 5.1). On le note :

$$\tilde{q} = P_{\mathcal{N}\binom{\tilde{c}^T\tilde{X}}{\tilde{G}\tilde{X}}}\mathbf{1}.$$

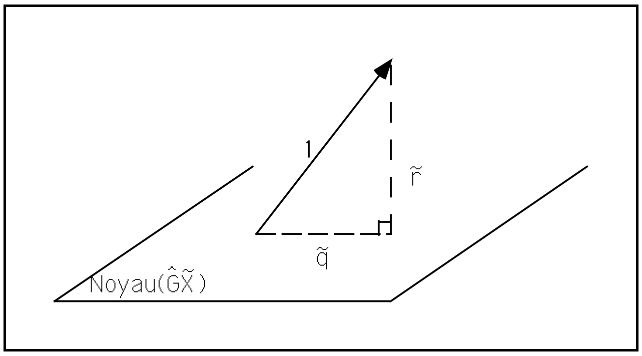

Figure 5.1: Projection orthogonale de $\mathbf{1}$ dans l'espace nul de $\hat{G}\tilde{X} = \binom{\tilde{c}^T\tilde{X}}{\tilde{G}\tilde{X}}$.

Le calcul de $\tilde{q}$ pourrait s'effectuer directement. En effet, en définissant la matrice $\hat{G} = \binom{\tilde{c}^T}{\tilde{G}}$, la projection de $\mathbf{1}$ dans l'espace nul de $\hat{G}\tilde{X}$ s'écrit :

$$\tilde{q} = \mathbf{1} - \tilde{X}\hat{G}^T(\hat{G}\tilde{X}^2\hat{G}^T)^{-1}\hat{G}\tilde{X}\mathbf{1}.$$

Dans la pratique, il paraît plus judicieux de procéder autrement. Pour calculer $\tilde{q}$ nous allons rester avec les composantes de la matrice sans les agréger. Comme $\tilde{x}$ et $\tilde{q}$ se trouvent dans les espaces nuls de $\tilde{G}$ et $\tilde{G}\tilde{X}$ respectivement, on peut remarquer que si l'on multiplie les deux membres de l'équation (5.8) par $\tilde{G}\tilde{X}$, on trouve :

$$\tilde{G}\tilde{X}\mathbf{1} = \tilde{G}\tilde{X}\tilde{q} + \tilde{G}\tilde{X}\tilde{r}.$$

Comme $\tilde{G}\tilde{X}\mathbf{1} = 0$ et $\tilde{G}\tilde{X}\tilde{q} = 0$ on a :

$$\tilde{G}\tilde{X}\tilde{r} = 0. \tag{5.9}$$

Etant donné que la matrice $\tilde{G}$ contient la matrice identité et que $\tilde{x} > 0$, la matrice $\tilde{G}\tilde{X}$ a plein rang et $\tilde{G}\tilde{X}^2\tilde{G}^T$ est symétrique définie positive. En développant (5.9) on peut obtenir l'expression de λ en fonction de λ_0 :

$$\lambda_0\tilde{G}\tilde{X}^2\tilde{c} + \tilde{G}\tilde{X}^2\tilde{G}^T\lambda = \tilde{G}\tilde{X}\tilde{r} = 0.$$

On en déduit :

$$\lambda = -\lambda_0(\tilde{G}\tilde{X}^2\tilde{G}^T)^{-1}\tilde{G}\tilde{X}^2\tilde{c}.$$

Pour obtenir λ_0, on procède de manière similaire. Comme $\tilde{c}^T\tilde{X}\tilde{q} = \tilde{c}^T\tilde{d} = 0$, en prémultipliant l'équation (5.8) par $\tilde{c}^T\tilde{X}$ on obtient :

$$\begin{aligned}\tilde{c}^T\tilde{x} &= \lambda_0\tilde{c}^T\tilde{X}^2\tilde{c} + \tilde{c}^T\tilde{X}^2\tilde{G}^T\lambda \\ &= \lambda_0\tilde{c}^T\tilde{X}^2\tilde{c} - \tilde{c}^T\tilde{X}^2\tilde{G}^T\lambda_0(\tilde{G}\tilde{X}^2\tilde{G}^T)^{-1}\tilde{G}\tilde{X}^2\tilde{c}.\end{aligned}$$

Comme[2] $\tilde{c}^T\tilde{x} = \sigma > 0$, il est possible d'extraire λ_0 :

$$\begin{aligned}\lambda_0 &= \frac{\tilde{c}^T\tilde{x}}{\tilde{c}^T\tilde{X}^2\tilde{c} - \tilde{c}^T\tilde{X}^2\tilde{G}^T(\tilde{G}\tilde{X}^2\tilde{G}^T)^{-1}\tilde{G}\tilde{X}^2\tilde{c}} \\ &= \frac{\tilde{c}^T\tilde{x}}{\tilde{c}^T\tilde{X}(\tilde{X}\tilde{c} - \tilde{X}\tilde{G}^T(\tilde{G}\tilde{X}^2\tilde{G}^T)^{-1}\tilde{G}\tilde{X}^2\tilde{c})}.\end{aligned}$$

On retrouve dans la seconde partie du dénominateur l'expression d'une projection. On remarque de plus qu'il exprime le produit scalaire entre un vecteur et sa projection. Par définition, ceci est égal à la norme de cette projection au carré. Le multiplicateur λ_0 s'écrit donc :

$$\lambda_0 = \frac{\tilde{c}^T\tilde{x}}{\tilde{c}^T\tilde{X}P_{\mathcal{N}(\tilde{G}\tilde{X})}\tilde{X}\tilde{c}} = \frac{\tilde{c}^T\tilde{x}}{\|P_{\mathcal{N}(\tilde{G}\tilde{X})}\tilde{X}\tilde{c}\|^2}.$$

[2]Nous avons vu que $\tilde{c}^T\tilde{x}^k > 0$.

On peut alors écrire $\tilde{q}$:

$$\begin{aligned}\tilde{q} &= \mathbf{1} - \lambda_0 \tilde{X}\tilde{c} - \tilde{X}\tilde{G}^T\lambda \\ &= \mathbf{1} - \lambda_0(\tilde{X}\tilde{c} - \tilde{X}\tilde{G}^T(\tilde{G}\tilde{X}^2\tilde{G}^T)^{-1}\tilde{G}\tilde{X}^2\tilde{c}) \\ &= \mathbf{1} - \lambda_0 P_{\mathcal{N}(\tilde{G}\tilde{X})}\tilde{X}\tilde{c}.\end{aligned}$$

On trouve ainsi l'expression finale de $\tilde{q}$:

$$\tilde{q} = \mathbf{1} - \frac{\tilde{c}^T\tilde{x}}{\|P_{\mathcal{N}(\tilde{G}\tilde{X})}\tilde{X}\tilde{c}\|^2} P_{\mathcal{N}(\tilde{G}\tilde{X})}\tilde{X}\tilde{c}, \tag{5.10}$$

ou, de manière équivalente :

$$\tilde{q} = \mathbf{1} - \frac{\tilde{c}^T\tilde{x}}{\|\tilde{X}\tilde{s}\|^2}\tilde{X}\tilde{s}, \tag{5.11}$$

avec

$$\tilde{s} = \tilde{c} + \tilde{G}^T u, \tag{5.12}$$

et

$$u = -(\tilde{G}\tilde{X}^2\tilde{G}^T)^{-1}\tilde{G}\tilde{X}^2\tilde{c}. \tag{5.13}$$

5.4 Test d'arrêt des itérations de Newton

Le centre analytique est calculé par la méthode de Newton. Nous venons de voir comment calculer la direction ; explicitons maintenant le critère d'arrêt de cette procédure itérative. Par définition, la direction de Newton $\tilde{X}\tilde{q}$ s'annule lorsque le point $\tilde{x}$ minimise (5.5). Comme $\tilde{x} > 0$, au centre analytique le vecteur $\tilde{q}$ est nul. Le test d'arrêt théorique est donc $\tilde{q} = 0$.

Théorème : 5.5 [2]. *Le point $\tilde{x}$ obtenu par la méthode de Newton lorsque le critère d'arrêt est $\tilde{q} = 0$ satisfait les conditions d'optimalité (4.12).*

Preuve. De (5.11) on obtient alors l'égalité suivante :

$$\tilde{X}\tilde{s} = \frac{\|\tilde{X}\tilde{s}\|^2}{\tilde{c}^T\tilde{x}}\mathbf{1}. \tag{5.14}$$

Comme $\tilde{c}^T\tilde{x} > 0$ on remarquera que $\tilde{s} > 0$. De plus, nous avons vu que le problème est homogène de degré 0, c'est-à-dire $\tilde{\varphi}(t\tilde{x}) = \tilde{\varphi}(\tilde{x})$. On peut donc faire l'hypothèse que $\tilde{x}_0 = 1$. La première équation de (5.14) s'écrit donc :

$$s_0 = \frac{\|\tilde{X}\tilde{s}\|^2}{\tilde{c}^T\tilde{x}}.$$

En la subtituant dans (5.14), on retrouve la dernière condition d'optimalité de (4.12) :

$$Xs = s_0\mathbf{1}.$$

En développant $\tilde{s} = \tilde{c} + \tilde{G}^T u$, on trouve :

$$\begin{pmatrix} s_0 \\ s \end{pmatrix} = \begin{pmatrix} \overline{z} \\ -c \end{pmatrix} + \begin{pmatrix} -b^T \\ G^T \end{pmatrix} u,$$

qui donne la deuxième et la troisième condition d'optimalité de (4.12) :

$$\begin{aligned} s &= -c + G^T u, & s &> 0 \\ s_0 &= \overline{z} - b^T u, & s_0 &> 0. \end{aligned}$$

Comme par ailleurs, $\tilde{G}\tilde{x} = 0$, on retrouve bien toutes les conditions d'optimalité (4.12). Le point $\tilde{x}$ déterminé par la méthode de Newton est bien le centre analytique recherché. ∎

Dans la pratique, le test d'arrêt $\tilde{q} = 0$ n'est pas opérationnel. On lui substitue le test $\|\tilde{q}\| < \varepsilon$. Ce critère permet de définir un **centre analytique approché**. La motivation est la suivante : pour améliorer les performances de l'algorithme ACCPM il faut réduire au maximum le temps passé dans le programme-maître. Comme le calcul de ce centre analytique u consomme la plupart du temps de calcul (les autres actions faites étant de durée beaucoup plus courte) en relâchant le critère d'arrêt, on améliore les performances de l'algorithme.

On remplace donc le test d'arrêt $\tilde{q} = 0$ par $\|q\| < \varepsilon$ qui définit un centre analytique approché. Il nous faut cependant déterminer la valeur de ε. Plus elle est petite, plus le nombre de pas de Newton pour satisfaire le critère d'arrêt risque d'être important. D'un autre coté, une faible valeur de ε fournira un point plus centré et donc aura des chances de générer des coupes plus profondes. De fait, l'expérience a montré que la qualité du centrage n'avait que peu d'incidence sur le nombre d'appels à l'oracle, c'est-à-dire sur la qualité des coupes. Le critère $\|q\| < \varepsilon$ a donc été adopté. Etudions maintenant quelles valeurs peut prendre ε.

L'objectif est de déterminer un point u à l'intérieur de l'ensemble de localisation $\mathcal{L}_{\bar{z}}$. Cependant, le calcul du centre analytique se fait dans le primal, c'est-à-dire qu'il permet de déterminer un vecteur $\tilde{x}$ et non un vecteur u. Heureusement, nous avons vu que des vecteurs $\tilde{s}$ et u peuvent être dérivés de ce calcul (voir (5.11)) et que lorsque $\tilde{q} = 0$, ils satisfont les conditions d'optimalité. Qu'en-est il si l'on s'arrête avant, avec $\|\tilde{q}\| < \varepsilon$?

On peut remarquer [1] que si $\|\tilde{q}\|_\infty < \mathbf{1}$, $\tilde{s}$ est positif. En effet, comme d'une part $\tilde{q} = \mathbf{1} - \frac{\tilde{c}^T\tilde{x}}{\|\tilde{X}\tilde{s}\|^2}\tilde{X}\tilde{s}$ et d'autre part $\tilde{x} > 0$ et $\tilde{c}^T\tilde{x} > 0$, il découle aisément que si $\|\tilde{q}\|_\infty < \mathbf{1}$ alors $\tilde{s}$ est positif. De plus, comme $\tilde{s} = \tilde{c} + \tilde{G}^T u$, le point u défini ainsi appartient à l'ensemble de localisation.

Dans la pratique, le test d'arrêt dans la méthode de Newton est $\|\tilde{q}\| < \theta \leq 1$. Le vecteur u obtenu appartient donc nécessairement à l'ensemble de localisation. Par la suite, nous nommerons u^c le centre analytique approché pour le problème $\mathcal{D}$ et $\tilde{x}^c$ le point $\tilde{x}$ associé dans le problème $\mathcal{P}$.

5.5 Algorithme projectif

Avant d'énoncer l'algorithme projectif complet, faisons quelques rappels et donnons quelques précisions.

Tout d'abord le calcul de la direction de Newton se base sur le fait qu'un point initial $\tilde{x} > 0$ tel que $\tilde{G}\tilde{x} = 0$ est connu. On le supposera dans l'immédiat et on verra plus tard comment satisfaire cette exigence.

Le calcul de la direction et le critère d'arrêt ont été développés dans les sections précédentes.

La direction de Newton a été calculée pour résoudre le problème homogène (5.4) dans l'espace projectif. Pour trouver le point x qui résoud le problème (5.1) dans $\mathbb{R}^n$, il suffit, comme $\tilde{x} > 0$, de diviser toutes les composantes de $\tilde{x}$ par $\tilde{x}_0$. Le pas de Newton est donc :

$$\tilde{x} := \frac{\tilde{x} + \bar{\alpha}\tilde{d}}{1 + \bar{\alpha}\tilde{d}_0},$$

où $\bar{\alpha}$ la valeur obtenue lors de la recherche unidimensionnelle (voir chapitre suivant).

Algorithme projectif

Paramètre :
Paramètre d'arrêt pour les itérations : $0 \leq \theta \leq 1$;
Initialisation :
Déterminer un vecteur $\tilde{x} > 0$ tel que $\tilde{G}\tilde{x} = 0$;
Calculer la direction de Newton $\tilde{d} := \tilde{X}\tilde{q}$ par (5.11) ;

Tant que $\|\tilde{q}\| > \theta$, **faire**
Recherche unidimensionnelle :
Si $\|q\| \geq 1$, **alors** $\bar{\alpha} := \text{argmin}_\alpha\{\tilde{\varphi}(\tilde{x} + \alpha\tilde{d}) : \quad \tilde{x} + \alpha\tilde{d} > 0\}$;
Sinon $\alpha := 1$ **fsi** ;
Pas de Newton :
$\tilde{x} := \frac{\tilde{x}+\bar{\alpha}\tilde{d}}{1+\bar{\alpha}\tilde{d}_0}$;
Calcul de la direction de Newton :
$\tilde{d} := \tilde{X}\tilde{q}$ par (5.11) ;
Fin.

5.6 Analyse de convergence

Dans cette section nous allons étudier la convergence de l'algorithme projectif. Tout d'abord, on peut remarquer que si $\|\tilde{q}\| < 1$, l'algorithme de Newton converge quadratiquement vers l'optimum $\tilde{q} = 0$. Ceci dérive du théorème suivant :

Théorème : 5.6 [3]. *Si* $\|\tilde{q}^k\| < 1$ *alors* $\|\tilde{q}^{k+1}\| \leq \|\tilde{q}^k\|_\infty \|\tilde{q}^k\| \leq \|\tilde{q}^k\|^2$.

Preuve. De par la définition de $\tilde{q}^k$, projection orthogonale de $\mathbf{1}$ dans l'espace nul de $\hat{G}\tilde{X}^k$, $\tilde{q}^k$ est le vecteur de plus petite norme de la forme $\tilde{q}^k = \mathbf{1} - \tilde{r}$ où $\tilde{r}$ appartient à l'espace image de $\tilde{X}^k\hat{G}^T$. Il peut s'écrire ainsi :

$$\tilde{q}^k = \mathbf{1} - \tilde{X}^k\hat{s}^k,$$

où $\hat{s}^k = \hat{G}^T(\hat{G}{\tilde{X}^k}^2\hat{G})^{-1}\hat{G}\tilde{X}^k\mathbf{1}$.

Comme $\|\tilde{q}^k\| < 1$, le pas $\tilde{x}^{k+1} = \tilde{X}^k(\mathbf{1}+\tilde{q}^k)$ est admissible (c.à.d. $\tilde{x}^{k+1} > 0$). De plus, remarquons que $\tilde{X}^{k+1}\hat{s}^k$ appartient à l'espace image de $\tilde{X}^{k+1}\hat{G}^T$. Comme $\tilde{q}^{k+1}$ est le vecteur de plus petite norme de la forme $\tilde{q}^{k+1} = \mathbf{1} - \tilde{r}$ où $\tilde{r}$ appartient à l'espace image de $\tilde{X}^{k+1}\hat{G}^T$:

$$\|\tilde{q}^{k+1}\| = \|\mathbf{1} - \tilde{X}^{k+1}\hat{s}^{k+1}\| \leq \|\mathbf{1} - \tilde{X}^{k+1}\hat{s}^k\|.$$

Comme :

$$\begin{aligned} \tilde{x}^{k+1} &= \tilde{X}^k(\mathbf{1} + \tilde{q}^k) \\ &= \tilde{x}^k + \tilde{X}^k(\mathbf{1} - \tilde{X}^k\hat{s}^k) \\ &= 2\tilde{x}^k - \tilde{X}^{k^2}\hat{s}^k, \end{aligned}$$

il résulte que

$$\begin{aligned} \|\tilde{q}^{k+1}\| &\leq \|\mathbf{1} - \hat{S}^k(2\tilde{x}^k - \tilde{X}^{k^2}\hat{s}^k)\| \\ &\leq \left(\sum_{i=0}^{\kappa}(1 - 2\hat{s}_i^k\tilde{x}_i^k + \hat{s}_i^{k^2}\tilde{x}_i^{k2})^2\right)^{\frac{1}{2}} \\ &\leq \left(\sum_{i=0}^{\kappa}(1 - \hat{s}_i^k\tilde{x}_i^k)^4\right)^{\frac{1}{2}} \\ &\leq \left(\max_i\{(1 - \hat{s}_i^k\tilde{x}_i^k)^2\}\sum_{i=0}^{\kappa}(1 - \hat{s}_i^k\tilde{x}_i^k)^2\right)^{\frac{1}{2}} \\ &\leq \|\tilde{q}^k\|_\infty\|\tilde{q}^k\| \leq \|\tilde{q}^k\|^2. \end{aligned}$$

■

Le théorème précédent indique comment l'algorithme projectif converge lorsque $\|q^k\| < 1$. Le théorème suivant montre comment décroît la fonction potentielle à chaque itération. Il implique donc la convergence globale de l'algorithme projectif.

Théorème : 5.7 [3]. *La fonction potentielle $\tilde{\varphi}(\tilde{x})$ décroît au moins d'une quantité :*

$$\Delta\tilde{\varphi} = \tilde{\varphi}(\tilde{x}^k) - \tilde{\varphi}(\tilde{x}^{k+1}) \geq \frac{\|\tilde{q}\|^2}{2(1 + \|q\|_\infty)}.$$

Preuve. Tout d'abord rappelons que pour tout $\lambda \in \mathbb{R}$ tel que $\lambda < 1$,

$$\ln(1 + \lambda) \geq \lambda - \frac{\lambda^2}{2(1 - |\lambda|)}. \tag{5.15}$$

En prenant $\alpha = (1 - \|\tilde{q}\|_\infty)^{-1}$, on a alors nécessairement :

$$\|\alpha\tilde{q}\|_\infty < 1.$$

De (5.15) on déduit

$$\ln(1 + \alpha\tilde{q}_i) \geq \alpha\tilde{q}_i - \frac{\alpha^2\tilde{q}_i^2}{2(1 - \alpha\tilde{q}_i)}.$$

En remarquant que $|\tilde{q}_i| \leq \|\tilde{q}\|_\infty$ et sommant sur i on obtient :

$$\sum_{i=0}^{\kappa} \ln(1 + \alpha\tilde{q}_i) \geq \mathbf{1}^T\tilde{q} - \frac{\alpha^2\|\tilde{q}\|^2}{2(1 - \alpha\|\tilde{q}\|_\infty)}. \tag{5.16}$$

Comme $\tilde{d}$ est dans le noyau de $\hat{G}$ et $\tilde{c}^T\tilde{d} = 0$ on a alors :

$$\Delta\tilde{\varphi} = \tilde{\varphi}(\tilde{x}^k) - \tilde{\varphi}(\tilde{x}^{k+1}) = \sum_{i=0}^{\kappa} \ln(1 + \alpha\tilde{q}_i).$$

En remarquant que $\mathbf{1}^T\tilde{q} = \|\tilde{q}\|^2$ et en remplacant α par sa valeur dans (5.16) on obtient

$$\Delta\tilde{\varphi} \geq \frac{\|\tilde{q}\|^2}{2(1 + \|q\|_\infty)}.$$

■

Pour prouver le théorème ci-dessus on a fixé le paramètre α à une valeur qui n'est pas nécessairement celle qu'aurait donnée la recherche unidimensionnelle. On remarquera cependant qu'il vérifie les contraintes, c'est-à-dire $\tilde{x}^k + \alpha\tilde{d} > 0$. De plus, la recherche unidimensionnelle ayant pour objectif de touver un meilleur α, c'est-à-dire une valeur pour α qui fait décroître au maximum le potentiel $\tilde{\varphi}(\tilde{x})$ dans la direction $\tilde{d}$, la variation $\Delta\tilde{\varphi}$ sera au moins aussi grande. Enfin $\tilde{\varphi}(\tilde{x})$ étant homogène de degré zéro, le théorème ci-dessus est donc valable pour l'algorithme projectif puisque $\tilde{\varphi}(\tilde{x})$ n'est pas affecté par $\tilde{x} := \tilde{x}/\tilde{x}_0$.

Pour finir, Goffin et Vial [3] estiment le nombre d'itérations pour déterminer un nouveau centre analytique à :

Théorème : 5.8 [3]. *La complexité de l'algorithme projectif pour déterminer un centre analytique approché est :*

$$O\left(\frac{\kappa L}{\theta^2}\right),$$

où L est le nombre de bits nécessaires pour stocker les données du problème, κ le nombre de colonnes de la matrice $\tilde{G}$ et θ le critère d'arrêt.

Preuve. La preuve de ce théorème se déduit de [3]. ■

Il est important de noter que derrière le nombre L se cache la valeur initiale $\tilde{\varphi}(\tilde{x}^0)$. Le choix de $\tilde{x}^0$ influence donc grandement le nombre d'itérations de l'algorithme projectif. Cette remarque justifie le soin que l'on prend pour introduire une nouvelle coupe[3]. En suivant la méthode proposée par Goffin, Luo et Ye [4], les auteurs montrent qu'il est possible de déterminer un nouveau centre analytique approché en quatre itérations.

Dans l'implantation de l'algorithme ACCPM nous introduisons des coupes plus profondes que dans [4], ce qui ne nous permet pas de déduire leur résultat. On verra cependant que, dans la pratique (dernier chapitre), très peu d'itérations dans le module projectif sont nécessaires.

5.7 Ellipsoïdes dans l'espace dual

Le centre analytique, qu'il soit exact ou approché, permet de définir deux ellipsoïdes. L'un $\mathcal{E}_{int}$ est inclus dans l'ensemble de localisation et l'autre $\mathcal{E}_{ext}$ englobe cet ensemble. Leur étude est intéressante puisqu'elle va nous permettre de définir un test pour supprimer des coupes et ainsi de réduire la dimension de la matrice $\tilde{G}$.

Soit $\tilde{x}$ un point primal réalisable (c.à.d. $\tilde{x} \in S_{\mathcal{P}}$) et $\bar{u}$ un point dans l'esapce dual défini par (5.13). Supposons de plus que $\bar{u} \in \mathcal{L}_{\overline{z}}$ l'ensemble de localisation[4]. On considère la famille d'ellipsoïdes $\mathcal{E}$ centrés en $\bar{u}$ défini par :

$$\mathcal{E} = \{u : \|\tilde{X}\tilde{G}^T(u - \bar{u})\| \leq \varepsilon\}.$$

Les théorèmes qui suivent montrent que pour des choix particuliers de ε on obtient les deux ellipsoïdes annoncés au paragraphe précédent.

Théorème : 5.9 [3]. *Soit $\tilde{x}$ un point primal réalisable (c.à.d. $\tilde{x} \in S_{\mathcal{P}}$) et $\bar{u}$ défini par (5.13). Supposons que $\bar{u} \in \mathcal{L}_{\overline{z}}$ l'ensemble de localisation. Si ε_e est défini par :*

$$\varepsilon_e^2 = \frac{\kappa}{\kappa + 1}(\tilde{c}^T\tilde{x})^2,$$

alors

$$\mathcal{E}_{ext} = \{u : \|\tilde{X}\tilde{G}^T(u - \bar{u})\| \leq \varepsilon_e\} \supseteq \mathcal{L}_{\overline{z}}.$$

[3]Nous verrons en effet que lors de l'introduction d'une coupe il nous faut déterminer un nouveau point $\tilde{x}$ tel que $\tilde{G}\tilde{x} = 0$.

[4]Dans un premier temps $\bar{u}$ n'est pas nécessairement ni le centre analytique ni un centre analytique approché.

Preuve. Pour tout u, un point quelconque de l'ensemble de localisation (c.à.d. $u \in \mathcal{L}_{\bar{z}}$) et $\bar{u}$, définissons $\tilde{s}$ et $\tilde{s}^c$ par (5.12). Par définition, si $u \in \mathcal{L}_{\bar{z}}$ alors le vecteur $\tilde{s}$ associé est strictement positif. De plus, le vecteur :

$$\tilde{X}(\tilde{s} - \tilde{s}^c) = \tilde{X}(\tilde{c} + \tilde{G}^T u - \tilde{c} - \tilde{G}^T \bar{u}) = \tilde{X}\tilde{G}^T(u - \bar{u}),$$

appartient à l'espace image de $\tilde{X}\tilde{G}^T$. Il est donc orthogonal à $\tilde{X}\tilde{s}^c$ qui lui, appartient à l'espace nul de $\tilde{G}\tilde{X}$. Nous avons alors l'égalité triangulaire :

$$\|\tilde{X}\tilde{s}\|^2 = \|\tilde{X}(\tilde{s} - \tilde{s}^c)\|^2 + \|\tilde{X}\tilde{s}^c\|^2.$$

Comme $\tilde{s} > 0$ et $\tilde{G}\tilde{x} = 0$:

$$\begin{aligned} \|\tilde{X}\tilde{s}\|^2 &\leq (\mathbf{1}^T\tilde{X}\tilde{s})^2 \\ &= (\tilde{x}^T(\tilde{c} + \tilde{G}^T u))^2 = (\tilde{c}^T\tilde{x})^2. \end{aligned}$$

De plus, comme $\tilde{q}$ et $\frac{\tilde{c}^T\tilde{x}}{\|\tilde{X}\tilde{s}\|^2}\tilde{X}\tilde{s}$ sont orthogonaux,

$$\|\mathbf{1}\|^2 = \kappa + 1 = \|\tilde{q}\|^2 + \frac{(\tilde{c}^T\tilde{x})^2}{\|\tilde{X}\tilde{s}\|^2}.$$

D'où :

$$\frac{(\tilde{c}^T\tilde{x})^2}{\|\tilde{X}\tilde{s}\|^2} \leq \kappa + 1. \tag{5.17}$$

Et plus particulièrement pour $\tilde{s}^c$:

$$-\|\tilde{X}\tilde{s}^c\|^2 \leq -\frac{(\tilde{c}^T\tilde{x})^2}{\kappa + 1}.$$

On obtient alors :

$$\|\tilde{X}\tilde{G}^T(u - \bar{u})\|^2 = \|\tilde{X}(\tilde{s} - \tilde{s}^c)\|^2 = \|\tilde{X}\tilde{s}\|^2 - \|\tilde{X}\tilde{s}^c\|^2 \leq \frac{\kappa}{\kappa + 1}(\tilde{c}^T\tilde{x})^2.$$

■

Tout comme nous venons de construire un ellipsoïde englobant l'ensemble de localisation, il est possible de construire un ellipsoïde inclus dans cet ensemble.

Théorème : 5.10 [3]. *Soit $\tilde{x}$ un point primal réalisable (c.à.d. $\tilde{x} \in S_{\mathcal{P}}$) et u^c défini par (5.13). Supposons de plus que u^c est un centre analytique approché*[5]*, c'est-à-dire $u^c \in \mathcal{L}_{\bar{z}}$ et $\|\tilde{q}\| < \theta$:*

$$\mathcal{E}_{int} = \left\{ u : \|\tilde{X}\tilde{G}^T(u - u^c)\| \leq \frac{1 - \theta}{\kappa + 1}\tilde{c}^T\tilde{x} \right\} \subseteq \mathcal{L}_{\bar{z}}.$$

[5]Le théorème est encore valable si l'on relache la condition de centrage en imposant simplement $\|\tilde{q}\|_\infty < \theta$.

Preuve. Rappelons la définition de $\tilde{q}$ donnée par (5.11) :

$$\tilde{q} = \mathbf{1} - \frac{\tilde{c}^T \tilde{x}}{\|\tilde{X}\tilde{s}\|^2}\tilde{X}\tilde{s}.$$

Comme $\|\tilde{q}\| < \theta$ alors :

$$(1-\theta) < \frac{\tilde{c}^T \tilde{x}}{\|\tilde{X}\tilde{s}\|^2}\tilde{x}_i\tilde{s}_i, \qquad i = 0, \ldots, \kappa.$$

De plus, comme $\tilde{c}^T\tilde{x} > 0$,

$$(1-\theta)\tilde{c}^T x < \frac{(\tilde{c}^T \tilde{x})^2}{\|\tilde{X}\tilde{s}\|^2}\tilde{x}_i\tilde{s}_i, \qquad i = 0, \ldots, \kappa. \tag{5.18}$$

Comme (5.17) est encore vrai, on peut alors écrire d'après (5.18) :

$$\frac{1-\theta}{1+\kappa}\tilde{c}^T\tilde{x} < \tilde{x}_i\tilde{s}_i, \qquad i = 0, \ldots, \kappa. \tag{5.19}$$

Vérifions maintenant que tout point u appartenant à $\mathcal{E}_{int}$ appartient à $\mathcal{L}_{\overline{z}}$. Soit $u \in \mathcal{E}_{int}$, définissons le vecteur $\tilde{s} := \tilde{c} + \tilde{G}^T u$. Comme u appartient à l'ellipsoïde intérieur,

$$\tilde{x}_i|\tilde{s}_i - \tilde{s}_i^c| \leq \frac{1-\theta}{\kappa+1}\tilde{c}^T\tilde{x}, \qquad i = 0, \ldots, \kappa. \tag{5.20}$$

De plus,

$$\tilde{x}_i|\tilde{s}_i - \tilde{s}_i^c| = \tilde{x}_i\tilde{s}_i^c\left|\frac{\tilde{s}_i}{\tilde{s}_i^c} - 1\right|, \qquad i = 0, \ldots, \kappa.$$

En utilisant cette dernière égalité avec (5.19) et (5.20) on obtient :

$$\left|\frac{\tilde{s}_i}{\tilde{s}_i^c} - 1\right| < 1, \qquad i = 0, \ldots, \kappa.$$

On vérifie bien que $\tilde{s} > 0$ ce qui prouve que tout point $u \in \mathcal{E}_{int}$ se trouve dans l'ensemble de localisation (c.à.d. $u \in \mathcal{L}_{\overline{z}}$). ■

On remarquera que les ellipsoïdes $\mathcal{E}_{int}$ et $\mathcal{E}_{ext}$ sont reliés par un facteur homothétique ρ :

$$\mathcal{E}_{ext} = \rho\mathcal{E}_{int} \quad \text{avec} \quad \rho = \frac{\sqrt{\kappa(1+\kappa)}}{1-\theta}.$$

Il est à noter que ρ ne dépend que de θ le facteur de centrage et κ le nombre de colonnes de $\tilde{G}$. Il ne dépend ni de $\tilde{x}$, u ou $\overline{z}$.

La figure 5.2 illustre l'ellispoïde extérieur et intérieur d'un ensemble de localisation.

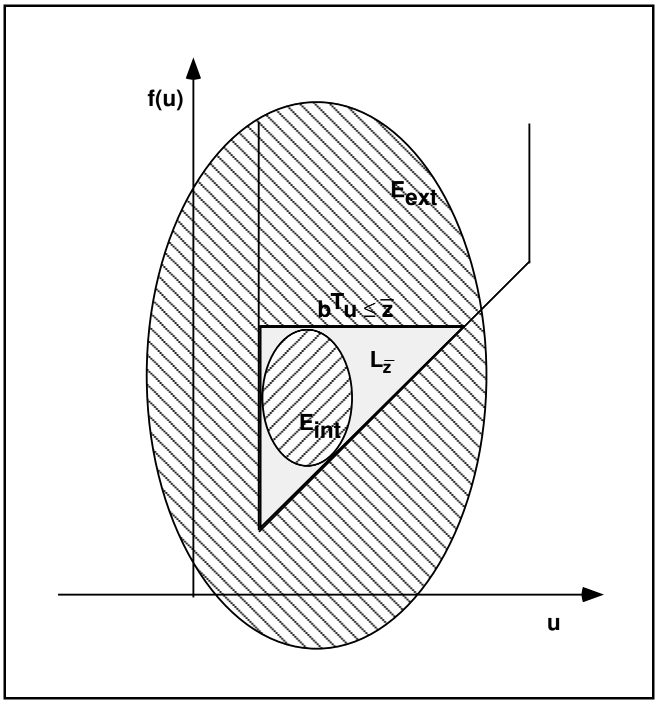

Figure 5.2: Les ellipsoïdes et l'ensemble de localisation.

5.8 Centre analytique pondéré

Goffin et Vial [3] ont proposé un algorithme permettant de calculer un centre analytique pondéré. L'introduction de poids dans la formulation est une façon de donner plus ou moins d'importance aux coupes. Bien que peu testée dans la pratique, cette formulation peut s'avérer intéressante. Par exemple augmenter l'importance de la borne supérieure par rapport aux coupes influencera l'algorithme projectif qui déterminera un centre analytique "plus bas" dans l'ensemble de localisation, c'est-à-dire plus proche du minimum de la relaxation linéaire. Dans l'implantation de ACCPM nous n'avons pas utilisé ce concept excepté pour le calcul de la borne inférieure que nous verrons plus tard.

Pour calculer un centre analytique pondéré, on ajoute au problème (4.11) les poids $\mu_i > 0$. On obtient alors la fonction potentielle suivante :

$$\begin{array}{rll} \text{Minimiser} & \psi_\mu(u,s) = -\mu_0 \ln(\overline{z} - b^T u) - \sum_{j=1}^{\kappa} \mu_j \ln s_j & \\ \text{s.c.} & G^T u - s = c, \qquad\qquad s > 0 & \\ & \overline{z} - b^T u > 0. & \end{array} \tag{5.21}$$

En menant les mêmes calculs que précédemment, on obtient aisément les conditions d'optimalité suivantes :

$$\begin{array}{rcll} Gx &=& b & \\ G^T u - s &=& c, & s \quad > 0 \\ b^T u + s_0 &=& \overline{z}, & s_0 \quad > 0 \\ Xs &=& \frac{s_0}{\mu_0} N\mathbf{1} & \quad > 0, \end{array} \tag{5.22}$$

où $x = \lambda/\lambda_0$ et N est la matrice diagonale dont les éléments sont les composantes $\mu_1, \ldots, \mu_\kappa$. En posant $\mu_{\kappa+1} = \sum_{j=0}^{\kappa} \mu_j$, on peut associer à (5.21) la fonction potentielle suivante dans l'espace primal :

$$\begin{array}{rl} \text{Minimiser} & \varphi_\mu(x) = \mu_{\kappa+1} \ln(\overline{z} - c^T x) - \sum_{j=1}^{\kappa} \mu_j \ln x_j \\ \text{s.c.} & Gx = b \\ & x > 0. \end{array} \tag{5.23}$$

On vérifie facilement que les problèmes (5.21) et (5.23) ont les mêmes conditions d'optimalité. Le problème homogène associé est :

$$\begin{array}{rl} \text{Minimiser} & \tilde{\varphi}_\mu(\tilde{x}) = \mu_{\kappa+1} \ln \tilde{c}^T \tilde{x} - \sum_{j=0}^{\kappa} \mu_j \ln \tilde{x}_j \\ \text{s.c.} & \tilde{G}\tilde{x} = 0 \\ & \tilde{x} > 0. \end{array} \tag{5.24}$$

On vérifie de même que $\tilde{\varphi}_\mu(\tilde{x})$ est homogène de degré zéro et qu'en divisant les composantes de $\tilde{x}$ par $\tilde{x}_0$, les problèmes (5.23) et (5.24) sont équivalents. On vérifiera de même que ce dernier est équivalent à :

$$\min_{\tilde{p} \in \mathbb{R}^{\kappa+1}} \{\tilde{\varphi}_\mu(\tilde{x}^k + \tilde{p}) : \quad \tilde{G}\tilde{p} = 0, \tilde{c}^T \tilde{p} = 0, \ \tilde{x}^k + \tilde{p} > 0\}. \tag{5.25}$$

En procédant comme pour le calcul du centre analytique, on est amené à résoudre le problème suivant :

$$\begin{array}{rl} \text{Minimiser} & -\tilde{d}^T \tilde{N} \tilde{X}^{-1} \mathbf{1} + \frac{1}{2} \|\tilde{N}^{1/2} \tilde{X}^{-1} \tilde{d}\|^2 \\ \text{s.c.} & \tilde{G}\tilde{d} = 0 \\ & \tilde{c}^T \tilde{d} = 0 \\ & \tilde{x} + \tilde{d} > 0. \end{array}$$

Les conditions d'optimalité du premier ordre sont :

$$\begin{aligned}
\tilde{N}\tilde{X}^{-2}\tilde{d} - \tilde{N}\tilde{X}^{-1}\mathbf{1} + \lambda_0\tilde{c} + \tilde{G}^T\lambda &= 0\\
\tilde{G}\tilde{d} &= 0\\
\tilde{c}^T\tilde{d} &= 0,
\end{aligned}$$

où (λ_0, λ) sont les multiplicateurs de Lagrange et où $\tilde{N}$ est la matrice diagonale dont les éléments sont les composantes $\mu_0 \ldots \mu_\kappa$. En prémultipliant la première équation par $\tilde{N}^{-1/2}\tilde{X}$, en posant $\tilde{\mathcal{X}} = \tilde{N}^{-1/2}\tilde{X}$ et $\tilde{q} = \tilde{\mathcal{X}}^{-1}\tilde{d}$, on obtient :

$$\begin{aligned}
\tilde{N}^{1/2}\mathbf{1} &= \tilde{q} + (\tilde{\mathcal{X}}\tilde{c}, \tilde{\mathcal{X}}\tilde{G}^T)\begin{pmatrix}\lambda_0\\ \lambda\end{pmatrix}\\
\begin{pmatrix}\tilde{c}^T\tilde{\mathcal{X}}\\ \tilde{G}\tilde{\mathcal{X}}\end{pmatrix}\tilde{q} &= 0.
\end{aligned}$$

On retrouve un système semblable à (5.7) avec $\tilde{N}^{1/2}\mathbf{1}$ et $\tilde{q}$ appartenant au noyau de $\tilde{G}\tilde{\mathcal{X}}$. Un raisonnement identique à celui fait pour le calcul du centre analytique permet d'obtenir $\tilde{q}$:

$$\tilde{q} = \tilde{N}^{1/2}\mathbf{1} - \frac{\tilde{c}^T\tilde{x}}{\|P_{\mathcal{N}(\tilde{G}\tilde{\mathcal{X}})}\tilde{\mathcal{X}}\tilde{c}\|^2}P_{\mathcal{N}(\tilde{G}\tilde{\mathcal{X}})}\tilde{\mathcal{X}}\tilde{c}. \tag{5.26}$$

Si dans l'agorithme projectif on utilise, pour le calcul de $\tilde{q}$, (5.26) et non (5.11) on obtient un centre analytique pondéré. On peut de même [3], définir un centre analytique pondéré approché, un ellipsoïde inclus dans l'ensemble de localisation et centré en ce point, un ellipsoïde englobant l'ensemble de localisation et centré en ce point.

La complexité de cet algorithme projectif a été étudiée par Goffin et Vial [3]. Ils montrent qu'elle est :

$$O\left(\frac{\mu_{\kappa+1}L}{\theta^2}\right),$$

où L est le nombre de bits nécessaires pour stocker les données du problème et θ le critère d'arrêt.

Bibliographie

[1] G. de Ghellinck et J.-P. Vial, "A polynomial Newton method for linear programming", *Algorithmica* **1** (1986), 425-453.

[2] J. L. Goffin, A. Haurie et J. P. Vial, "Decomposition and nondifferentiable optimization with the projective algorithm", *Management Science* **38** (1992) 284-302.

[3] J. L. Goffin et J. P. Vial, "On the computation of weighted analytic centers and dual ellipsoids with the projective algorithm", *Mathematical programming*, **60** (1993) 81-92. (Quelques résultats proviennent d'une version précédente "Working paper" GERAD, Montréal).

[4] J. L. Goffin, Z. Luo et Y. Ye, "Complexity analysis of an interior cutting plane method for convex feasibility problems", manuscrit, McGill University, Canada (1993), To appear in *SIAM J. Optimization.*

[5] C. C. Gonzaga, "Large steps path-following methods for linear programming, Part II : Potential reduction method", *SIAM Journal on Optimization* **1** (1991) 280-292.

[6] N. Karmarkar, "A new polynomial-time algorithm for linear programming", in *Combinatorica* **4** (1984) 373-395.

[7] C. Roos et J. P. Vial, "Analytic centers in linear programming", Report 88-68, Faculty of Technical Mathematics and Informatics, Technische Universiteit Delft, Holland (1988).

[8] J.-P. Vial, "A unified approach to projective algorithms for linear programming", in *Lecture Notes in Mathematics 1405* , S. Verlag, ed., Proceedings of Fifth French-German Conference held in (Castel Novel, 1988), 191-220.

Chapitre 6

Mise en œuvre de l'algorithme

Les chapitres précédents donnent les éléments de base de l'algorithme ACCPM. Cependant, pour une mise en œuvre efficace, il est nécessaire d'ajouter quelques raffinements. En effet, nous avons vu au cours des deux chapitres précédents ce qui forme le cœur de l'algorithme mais rien n'a été précisé quant à la manière de faire une recherche unidimensionnelle, de calculer une borne inférieure et d'ajouter une coupe. Ces précisions vont faire l'objet de ce chapitre. Nous verrons, de plus, comment initialiser l'algorithme ACCPM (Phase I) et comment opérer pour limiter les instabilités numériques dans l'algorithme projectif (raffinement de la direction de Newton, procédure zéro-machine).

6.1 Recherche unidimensionnelle

Nous avons vu comment calculer la direction de Newton $\tilde{d} = \tilde{X}\tilde{q}$. Reste maintenant à déterminer le pas que l'algorithme projectif doit faire dans cette direction. Autrement dit, il nous faut déterminer la constante α telle que $\tilde{\varphi}(\tilde{x}^{k+1}) = \tilde{\varphi}(\tilde{x}^k + \alpha\tilde{d})$ soit le plus petit possible, tout en vérifiant la condition de non-négativité de $\tilde{x}^{k+1}$, c'est-à-dire $\tilde{x}^k + \alpha\tilde{d} > 0$.

Nous avons vu que la fonction $\tilde{\varphi}(\tilde{x} + \tilde{d})$ est convexe sur son ensemble de définition. Etant donné que $\alpha\tilde{G}\tilde{d} = 0$ et $\alpha\tilde{c}^T\tilde{d} = 0$, le problème $\min_\alpha\{\tilde{\varphi}(\tilde{x} + \alpha\tilde{d}) : \tilde{x} + \alpha\tilde{d} > 0\}$ est donc convexe en α.

Si on développe la fonction $\tilde{\varphi}(\tilde{x}+\alpha\tilde{d})$, comme $\tilde{c}^T d = 0$, on obtient :

$$\begin{aligned}
\tilde{\varphi}(\tilde{x}+\alpha\tilde{d}) &= (\kappa+1)\ln(\tilde{c}^T(\tilde{x}+\alpha\tilde{d})) - \sum_{j=0}^{\kappa}\ln(\tilde{x}_j+\alpha\tilde{d}_j) \\
&= (\kappa+1)\ln\tilde{c}^T\tilde{x} - \sum_{j=0}^{\kappa}\ln(\tilde{x}_j(1+\alpha\tilde{q}_j)) \\
&= (\kappa+1)\ln\tilde{c}^T\tilde{x} - \sum_{j=0}^{\kappa}\ln\tilde{x}_j - \sum_{j=0}^{\kappa}\ln(1+\alpha\tilde{q}_j).
\end{aligned}$$

Comme nous cherchons à déterminer la valeur de α pour laquelle cette fonction est minimale, il n'est pas nécessaire de conserver les termes en $\tilde{x}$. Le problème devient :

$$\operatorname{argmin}_\alpha\{\tilde{\phi}(\alpha) = -\sum_{j=0}^{\kappa}\ln(\mathbf{1}+\alpha\tilde{q}_j) : 1+\alpha\tilde{q} > 0\}.$$

Cette fonction est convexe sur son ensemble de définition qui est :

$$\max\left\{-\infty, \frac{-1}{\max_{\tilde{q}_i>0}(\tilde{q}_i)}\right\} < \alpha < \min\left\{\frac{-1}{\min_{\tilde{q}_i<0}(\tilde{q}_i)}, \infty\right\}.$$

Il est intéressant de constater que $\tilde{\phi}'(0) = -\tilde{q}^T\mathbf{1}$. Comme $\tilde{q}$ est la projection orthogonale de $\mathbf{1}$, nous avons $\tilde{q}^T\mathbf{1} = \|\tilde{q}\|^2$ et $\tilde{\phi}'(0) = -\|\tilde{q}\|^2$. Par conséquent, le minimum de $\tilde{\phi}$ doit se trouver à droite de 0, ce qui limite le domaine de recherche du minimum à :

$$0 < \alpha < \min\left\{\frac{-1}{\min_{\tilde{q}_i<0}(\tilde{q}_i)}, \infty\right\}.$$

Le minimum est obtenu par la méthode de Newton. Une itération se calcule de la manière suivante :

$$\begin{aligned}
\alpha^{k+1} &:= \alpha^k - \frac{\tilde{\phi}'(\alpha^k)}{\tilde{\phi}''(\alpha^k)} \\
&:= \alpha^k + \frac{\sum_{j=0}^{\kappa}(q_j/(1+\alpha q_j))}{\sum_{j=0}^{\kappa}(q_j/(1+\alpha q_j))^2}.
\end{aligned}$$

Si, dans le principe, nous avons bien implanté la méthode de Newton pour déterminer ce minimum, nous verrons au chapitre 8 que nous y avons apporté quelques raffinements. Ceux-ci interviennent à deux niveaux : dans le critère d'arrêt (valeur de la dérivée, nombre maximal d'itérations), dans la manière de réagir si on sort du domaine de recherche.

Lorsque le minimum est atteint (appelons $\bar{\alpha}$ ce minimum) le pas Newton dans l'algorithme projectif est donc :

$$\tilde{x}^{k+1} := \frac{1}{1+\bar{\alpha}\tilde{q}_0}\tilde{X}^k(\mathbf{1}+\bar{\alpha}\tilde{q}).$$

6.2 Calcul d'une borne inférieure

Nous avons vu qu'une borne supérieure de la valeur de la solution optimale du problème original (4.1) est envoyée par l'oracle lorsque celui-ci renvoie un plan de support de l'objectif. Pour définir un critère d'arrêt de l'algorithme ACCPM il nous faut connaître aussi une borne inférieure. L'écart entre ces deux valeurs nous permet alors de définir le saut de dualité. De nombreuses politiques peuvent être envisagées.

1. Il est tout d'abord intéressant de noter qu'à tout moment, x est un point réalisable du problème $\mathcal{P}$. Le scalaire $c^T x$ est donc une borne inférieure du problème $\mathcal{P}$. Comme ce problème est le dual de $\mathcal{D}$, approximation linéaire extérieure du problème original (4.1), $c^T x$ est donc aussi une borne inférieure de (4.1). Il est donc aisé de déterminer une borne inférieure après, ou même durant, le calcul du centre analytique approché :

$$\theta_1 = c^T x^c = \overline{z} - \tilde{c}^T \tilde{x}^c. \tag{6.1}$$

   Cette façon de calculer une borne inférieure est très rapide mais la valeur obtenue est de qualité assez médiocre. De nombreuses autres stratégies ont donc été élaborées.

2. Par opposition, la meilleure borne est celle obtenue par la résolution du programme linéaire $\mathcal{D}$ ou $\mathcal{P}$. Son calcul est malheureusement coûteux puisqu'il nécessite la résolution d'un programme linéaire.

3. Une approche intermédiaire consiste à trouver un point x réalisable tel que $c^T x$ soit le plus grand possible, sans pour autant nécessiter de trop importants calculs. Dans l'espace homogène, on remarque que $\tilde{c}^T \tilde{x} = \overline{z} - c^T x$ est le saut de dualité défini précédemment. Le problème devient alors : trouver un point $\tilde{x}$ tel que $\tilde{G}\tilde{x} = 0$ et tel que $\tilde{c}^T \tilde{x}$ soit le plus petit possible.

   Comme au centre analytique approché que nous venons de calculer, nous avions une direction de centrage $\tilde{d}$ qui assure que tout déplacement le long de cette direction conduit à un point $\tilde{x}$ dans le noyau de $\tilde{G}$, il suffit donc de faire une recherche unidimensionnelle le long de celle-ci visant à minimiser $\tilde{c}^T\tilde{x}$. Le problème s'écrit :

$$\min_\alpha \left\{ \tilde{\phi}(\alpha) = \frac{\tilde{c}^T\tilde{X}^c(\mathbf{1}+\alpha\tilde{q})}{1+\alpha\tilde{q}_0} : \ \tilde{X}^c(\mathbf{1}+\alpha\tilde{q}) > 0 \right\}.$$

Comme $\tilde{X}^c(\mathbf{1}+\alpha\tilde{q}) > 0$ et $\tilde{x}^c > 0$, l'ensemble de définition est le suivant :

$$\max\left\{-\infty, \frac{-1}{\max_{\tilde{q}_i>0}(\tilde{q}_i)}\right\} \leq \alpha \leq \min\left\{\frac{-1}{\min_{\tilde{q}_i<0}(\tilde{q}_i)}, \infty\right\}. \tag{6.2}$$

Etant donné que $\tilde{c}^T\tilde{X}^c(\mathbf{1}+\alpha\tilde{q}) = \tilde{c}^T\tilde{x}^c + \alpha\tilde{c}^T\tilde{d} = \tilde{c}^T\tilde{x}^c$ car $\tilde{c}^T\tilde{d} = 0$, le problème se récrit :

$$\min_\alpha \left\{ \tilde{\phi}(\alpha) = \frac{\tilde{c}^T\tilde{x}^c}{1+\alpha\tilde{q}_0} : \ \max\{-\infty, \frac{-1}{\max\limits_{\tilde{q}_i>0}(\tilde{q}_i)}\} \leq \alpha \leq \min\{\frac{-1}{\min\limits_{\tilde{q}_i<0}(\tilde{q}_i)}, \infty\} \right\}.$$

Sachant que le problème est borné, que $\tilde{c}^T\tilde{x}^c > 0$ et que la dérivée suivant α de cette fonction est $\tilde{\phi}(\alpha)' = \frac{-\tilde{q}_0\tilde{c}^T\tilde{x}^c}{(1+\alpha\tilde{q}_0)^2}$, la solution du problème est donc :

Si $\tilde{q}_0 > 0$, **alors**

$\alpha = \frac{-1}{\min_{\tilde{q}_i<0}(\tilde{q}_i)}$;

Sinon

$\alpha = \frac{-1}{\max_{\tilde{q}_i>0}(\tilde{q}_i)}$;

Fsi.

La borne inférieure obtenue est $\theta_2 = \overline{z} - \frac{\tilde{c}^T\tilde{x}^c}{1+\alpha\tilde{q}_0}$. Par définition $\theta_2 \geq \theta_1$.

4. Une dernière possibilité consiste à chercher une meilleure direction $\tilde{d}$ dans le noyau de $\tilde{G}$ et de faire ensuite une recherche unidimensionnelle dans cette direction afin de minimiser $\tilde{c}^T(\tilde{x}^c + \alpha\tilde{d})$.

   Pour cela on s'appuie sur l'étude faite pour calculer un centre analytique pondéré. Soit le vecteur des pondérations μ dont toutes les composantes sont égales à l'unité excepté $\mu_0 > 0$ (cette pondération permet de privilégier plus ou moins l'information sur la borne supérieure), on déduit de la dernière condition d'optimalité (5.22) :

$$x^Ts = \frac{\kappa s_0}{\mu_0}.$$

   Le saut de dualité est donc d'autant plus petit que μ_0 est grand.

   Si l'on fait l'hypothèse que le poids μ_0 mis sur l'objectif est si grand qu'il rend négligeable les poids unitaires mis sur les contraintes, on peut approximer le vecteur à projeter :

$$\tilde{\mathcal{X}}\tilde{c} \equiv \tilde{X}\begin{pmatrix}\mu_0\overline{z}\\ \mathbf{0}\end{pmatrix} = \mu_0\overline{z}\begin{pmatrix}1\\ \mathbf{0}\end{pmatrix},$$

où $\mathbf{0}$ est le vecteur nul de dimension appropriée.

Inversement, si le poids sur l'objectif est négligeable par rapport aux poids unitaires mis sur les autres contraintes, on peut approximer le vecteur à projeter :

$$\tilde{\mathcal{X}}\tilde{c} \equiv \tilde{X}\begin{pmatrix} 0 \\ -c \end{pmatrix} = \begin{pmatrix} 0 \\ -Xc \end{pmatrix}.$$

Lors du calcul du centre analytique, on calcule la projection du vecteur $\tilde{X}\tilde{c}$ dans l'espace nul de la matrice $\tilde{G}\tilde{X}$. Le vecteur $\tilde{X}\tilde{c}$ peut être interprété comme la somme de deux vecteurs :

$$\tilde{X}\tilde{c} = \overline{z}\begin{pmatrix} 1 \\ \mathbf{0} \end{pmatrix} + \begin{pmatrix} 0 \\ -Xc \end{pmatrix}.$$

Cette remarque, ainsi que les précédentes, motive l'étude particulière de la projection de $\binom{1}{\mathbf{0}}$ dans l'espace nul de $\tilde{G}\tilde{X}$. Il est probable que, dans l'absolu, la projection de $\tilde{\mathcal{X}}\tilde{c}$ dans l'espace nul de $\tilde{G}\tilde{\mathcal{X}}$ avec un poids μ_0 important donne une meilleure direction de recherche et ainsi un saut de dualité plus petit. Cependant la factorisation de $\tilde{G}\tilde{X}$ ayant déjà été effectuée lors du calcul du centre analytique, la direction donnée par cette projection reste relativement peu coûteuse et donc plus intéressante.

La direction $\tilde{d}$ est la suivante :

$$\tilde{d} = P_{\mathcal{N}(\tilde{G}\tilde{X})}\begin{pmatrix} \overline{z} \\ \mathbf{0} \end{pmatrix}.$$

Par définition, cette direction se trouve dans le noyau de $\tilde{G}\tilde{X}$. Comme pour le calcul de la borne inférieure précédente, cette borne s'obtient en faisant la recherche unidimensionnelle suivante :

$$\min_{\alpha} \left\{ \tilde{\phi}(\alpha) = \frac{\tilde{c}^T \tilde{X}^c(\mathbf{1} + \alpha\tilde{d})}{1 + \alpha\tilde{d}_0} : \ \tilde{X}^c(\mathbf{1} + \alpha\tilde{d}) > 0 \right\}.$$

L'ensemble de définition est alors :

$$\max\left\{ -\infty, \frac{-1}{\max_{\tilde{d}_i > 0}(\tilde{d}_i)} \right\} \leq \alpha \leq \min\left\{ \frac{-1}{\min_{\tilde{d}_i < 0}(\tilde{d}_i)}, \infty \right\}.$$

La dérivée de cette fonction est :

$$\tilde{\phi}(\alpha)' = \frac{\tilde{c}\tilde{X}^c\tilde{d} - \tilde{d}_0\tilde{c}^T\tilde{x}^c}{(1 + \alpha\tilde{d}_0)^2}.$$

La solution du problème est donc :

Si $\tilde{c}\tilde{X}^c\tilde{d} - \tilde{d}_0\tilde{c}^T\tilde{x}^c < 0$, **alors**

$\alpha = \frac{-1}{\min_{\tilde{d}_i<0}(\tilde{d}_i)}$;

Sinon

$\alpha = \frac{-1}{\max_{\tilde{d}_i>0}(\tilde{d}_i)}$;

Fsi.

On obtient alors la borne inférieure : $\theta_3 = \overline{z} - \frac{\tilde{c}^T\tilde{x}^c + \alpha\tilde{c}^T\tilde{X}^c\tilde{d}}{1+\alpha\tilde{d}_0}$. Dans la pratique, on observe que θ_3 est souvent meilleure que θ_2.

6.3 Technique d'adjonction de coupes

A chaque itération de l'algorithme ACCPM, l'oracle envoie soit une coupe de réalisabilité, soit une coupe d'optimalité. Posons dans un cadre général $\tilde{\omega}^T u \geq \gamma$ l'information renvoyée. L'oracle enrichit donc l'approximation linéaire du problème original (4.1) et cela se traduit par l'ajout d'une colonne dans la matrice $\tilde{G}$. L'étape suivante de l'algorithme ACCPM consiste à calculer un nouveau centre analytique par l'algorithme projectif. Rappelons que la méthode projective se base sur un point $\tilde{x}^k$ toujours primal réalisable (c.à.d. $\tilde{G}\tilde{x}^k = 0$ et $\tilde{x}^k > 0$). Lorsque l'on ajoute une colonne à $\tilde{G}$, on doit alors trouver un nouveau point réalisable. Soit $\tilde{x}$ le point courant avant l'ajout de la colonne, le point satisfait $\tilde{G}\tilde{x} = 0$ et $\tilde{x} > 0$. Soit $\tilde{\omega}$ la nouvelle colonne, le but est donc de trouver un nouveau point $\binom{\tilde{x}}{x_\omega}$ tel que :

$$\begin{cases} \tilde{G}\tilde{x} + x_\omega\tilde{\omega} = 0 \\ \tilde{x} > 0, \ \ x_\omega > 0. \end{cases}$$

On cherche alors à résoudre un problème de réalisabilité semblable à celui étudié par de Ghellinck et Vial [3]. Définissons la matrice $\bar{G}$ et le vecteur $\bar{x}$:

$$\bar{G} = (\tilde{G} \ \ \tilde{\omega}), \quad \bar{x} = \begin{pmatrix} \tilde{x} \\ x_\omega \end{pmatrix}.$$

Soit $\bar{g} = (1, 0, \ldots, 0)^T$, le problème consiste à trouver un point $\bar{x}$ tel que :

$$\begin{cases} \bar{G}\bar{x} &= & 0 \\ \bar{g}^T\bar{x} &= & 1 \\ \bar{x} &\geq & 0. \end{cases}$$

Si l'on fait l'hypothèse de compacité suivante :

Hypothèse : 6.1 $\bar{x}$ *satisfait :*

$$\{\bar{G}\bar{x} = 0, \bar{g}^T\bar{x} = 0, \bar{x} \geq 0\} \Rightarrow \bar{x} = 0.$$

Vial [6] a montré que le problème non linéaire :

$$\text{argmin}\left\{\Psi(\bar{x}) = \frac{\|\bar{G}\bar{x}\|}{\left(\prod_{i=0}^{\kappa+1}\bar{x}\right)^{\frac{1}{\kappa+2}}} : \ \bar{x} > 0\right\},$$

converge vers la solution recherchée.

Par définition, si un tel vecteur $\bar{x}$ existe, la valeur de $\Psi(\bar{x})$ en ce point est nulle. La fonction $\Psi(.)$ est homogène de degré zéro. On peut montrer [6] que ce problème est équivalent au suivant :

$$\min\{\Psi(\bar{x}+\bar{p}) : \ \bar{G}\bar{p} = 0, \ \bar{x}+\bar{p} > 0\},$$

où $\bar{x} > 0$ et où l'optimisation se fait sur $\bar{p}$.

Etant donné que la fonction logarithmique est monotone, il est indifférent pour la recherche du vecteur $\bar{x}$ de travailler avec $\Phi(\bar{x}) = \ln\Psi(\bar{x})$. Comme pour le calcul du centre analytique, nous recherchons la solution de ce problème par la méthode de Newton. La direction $\bar{d}$ est donnée par la résolution du problème suivant :

$$\begin{aligned} \text{Minimiser} \quad & \bar{d}^T\nabla\Phi(\bar{x}) + \tfrac{1}{2}\bar{d}^T\nabla^2\Phi(\bar{x})\bar{d} \\ \text{s.c.} \quad & \bar{G}\bar{d} = 0 \\ & \bar{x}+\bar{d} > 0. \end{aligned}$$

Comme,

$$\begin{aligned} \nabla\Phi(\bar{x}) &= \frac{\bar{G}^T\bar{G}\bar{x}}{\|\bar{G}\bar{x}\|^2} - \frac{1}{\kappa+2}\bar{X}^{-1}\mathbf{1}, \\ \nabla^2\Phi(\bar{x}) &= \frac{\bar{G}^T\bar{G}}{\|\bar{G}\bar{x}\|^2} - 2\frac{\bar{G}^T\bar{G}\bar{x}\bar{x}^T\bar{G}^T\bar{G}}{\|\bar{G}\bar{x}\|^4} + \frac{1}{\kappa+2}\bar{X}^{-2}, \end{aligned}$$

et $\bar{G}\bar{d} = 0$, la direction de Newton est obtenue en résolvant le problème :

$$\begin{aligned} \text{Minimiser} \quad & \tfrac{1}{\kappa+2}(-\bar{d}^T\bar{X}^{-1}\mathbf{1} + \tfrac{1}{2}\bar{d}^T\bar{X}^{-2}\bar{d}) \\ \text{s.c.} \quad & \bar{G}\bar{d} = 0 \\ & \bar{x}+\bar{d} > 0. \end{aligned}$$

Un raisonnement semblable à celui fait pour le calcul du centre analytique donne comme direction : $\bar{d} = \bar{X}\bar{q}$ où $\bar{q}$ est la direction suivante :

$$\bar{q} = \mathbf{1} - \bar{X}\bar{G}^T(\bar{G}\bar{X}^2\bar{G}^T)^{-1}\bar{G}\bar{X}\mathbf{1}.$$

L'algorithme de réalisabilité consiste donc à itérer suivant cette direction de la façon suivante :

$$\bar{x}^{k+1} := \frac{\bar{X}^k(\mathbf{1} + \bar{\alpha}\bar{q})}{1 + \bar{\alpha}\bar{q}_0},$$

jusqu'à ce que $\|\bar{G}\bar{x}\| < \varepsilon$ avec ε petit (proche de la précision machine). Dans notre cas, nous avons déjà un point $\tilde{x}$ tel que $\tilde{G}\tilde{x} = 0$. Le nouveau problème peut alors être considéré comme une perturbation du problème précédent. Si l'on développe $\bar{q}$ en fonction de $\tilde{G}$, $\tilde{x}$, $\tilde{\omega}$ et x_ω, on obtient tout d'abord :

$$\bar{G}\bar{X}\mathbf{1} = \tilde{G}\tilde{X}\mathbf{1} + x_\omega\tilde{\omega} = x_\omega\tilde{\omega},$$

et par la formule de Sherman-Morrison-Woodbury :

$$\begin{aligned}(\bar{G}\bar{X}^2\bar{G}^T)^{-1} &= (\tilde{G}\tilde{X}^2\tilde{G}^T + x_\omega^2\tilde{\omega}\tilde{\omega}^T)^{-1}\\ &= (\tilde{G}\tilde{X}^2\tilde{G}^T)^{-1} - \frac{x_\omega^2}{1 + x_\omega^2\tau^2}(\tilde{G}\tilde{X}^2\tilde{G}^T)^{-1}\tilde{\omega}\tilde{\omega}^T(\tilde{G}\tilde{X}^2\tilde{G}^T)^{-1},\end{aligned}$$

avec $\tau^2 = \tilde{\omega}^T(\tilde{G}\tilde{X}^2\tilde{G}^T)^{-1}\tilde{\omega}$.

Il découle que :

$$\begin{aligned}(\bar{G}\bar{X}^2\bar{G}^T)^{-1}\bar{G}\bar{X}\mathbf{1} &= (\tilde{G}\tilde{X}^2\tilde{G}^T)^{-1}\tilde{\omega}(x_\omega - \frac{x_\omega^3\tau^2}{1 + x_\omega^2\tau^2})\\ &= \frac{x_\omega}{1 + x_\omega^2\tau^2}(\tilde{G}\tilde{X}^2\tilde{G}^T)^{-1}\tilde{\omega},\end{aligned}$$

dont on déduit :

$$\begin{aligned}\bar{q} &= \mathbf{1} - \frac{x_\omega}{1 + x_\omega^2\tau^2}\begin{pmatrix}\tilde{X}\tilde{G}^T\\ x_\omega\tilde{\omega}^T\end{pmatrix}(\tilde{G}\tilde{X}^2\tilde{G}^T)^{-1}\tilde{\omega}\\ &= \mathbf{1} - \frac{x_\omega}{1 + x_\omega^2\tau^2}\begin{pmatrix}\tilde{X}\tilde{G}^T(\tilde{G}\tilde{X}^2\tilde{G}^T)^{-1}\tilde{\omega}\\ x_\omega\tau^2\end{pmatrix}.\end{aligned}$$

Il est alors toujours possible de prendre x_ω suffisamment petit de manière à ce que $\bar{q}$ soit positif et génère le point réalisable suivant :

$$0 < \begin{pmatrix}\tilde{X} & 0\\ 0 & x_\omega\end{pmatrix}\bar{q} = \begin{pmatrix}\tilde{X}(\mathbf{1} + \alpha\tilde{d})\\ \alpha\end{pmatrix}, \tag{6.3}$$

où

$$\alpha = \frac{x_\omega}{1 + x_\omega^2 \tau^2} \quad \text{et} \quad \tilde{d} = -\tilde{X}\tilde{G}^T(\tilde{G}\tilde{X}^2\tilde{G}^T)^{-1}\tilde{\omega}.$$

On remarquera que la direction $\tilde{d}$ est équivalente à la direction proposée par Mitchell dans [5] p. 54 pour ajouter un colonne. Le vecteur $\bar{q}$ étant dans le noyau de $\bar{G}\bar{X}$, le point $\bar{x} = \binom{\tilde{X}(\mathbf{1}+\alpha\tilde{d})}{\alpha}$ est primal réalisable (c.à.d. $\bar{G}\bar{x} = 0$). L'objectif est donc atteint.

Comme, par la suite, l'algorithme va chercher à déterminer un nouveau centre analytique, un choix judicieux pour α est de le prendre en fonction de la prochaine fonction potentielle à minimiser dans l'algorithme projectif. On va donc déterminer α de manière à minimiser la fonction suivante :

$$\begin{aligned}
\bar{\varphi}(\bar{x}) &= (\kappa+2)\ln[\tilde{c}^T\tilde{x} + \alpha(\tilde{c}^T\tilde{X}\tilde{d} - \gamma)] - \ln\alpha - \sum_{j=0}^{\kappa}\ln\tilde{x}_j(1+\alpha\tilde{d}_j) \\
&= (\kappa+2)\ln[\tilde{c}^T\tilde{x} + \alpha(\tilde{c}^T\tilde{X}\tilde{d} - \gamma)] - \ln\alpha - \sum_{j=0}^{\kappa}\ln\tilde{x}_j - \sum_{j=0}^{\kappa}\ln(1+\alpha\tilde{d}_j).
\end{aligned}$$

Comme l'objectif est de déterminer la valeur de α pour laquelle cette fonction est minimale, il n'est pas nécessaire de conserver la somme des termes $\tilde{x}$. De plus, comme $\bar{x}$ est primal réalisable[1] $\bar{c}^T\bar{x} > 0$, la valeur dans le premier logarithme est toujours positive. Le problème devient :

$$\begin{aligned}
\operatorname{argmin}_\alpha \quad & \bar{\phi}(\alpha) = (\kappa+2)\ln[\tilde{c}^T\tilde{x} + \alpha(\tilde{c}^T\tilde{X}\tilde{d} - \gamma)] - \ln\alpha - \sum_{j=0}^{\kappa}\ln(1+\alpha\tilde{d}_j) \\
\text{s.c.} \quad & \mathbf{1} + \alpha\tilde{d} > 0 \\
& \alpha > 0.
\end{aligned}$$

L'ensemble de définition de $\bar{\phi}(\alpha)$ est le suivant :

$$0 < \alpha < \min\left\{\frac{-1}{\min_{\tilde{d}_j<0}(\tilde{d}_j)}, \infty\right\}.$$

Contrairement à la recherche unidimensionnelle faite lors du calcul du centre analytique, on ne peut garantir la convexité de cette fonction. On n'est donc pas certain qu'une méthode de Newton pure converge. Nous avons constaté cependant que la variante proposée lors de la recherche unidimensionnelle du calcul du centre analytique donne aussi de bons résultats pour déterminer l'optimum de $\bar{\phi}(\alpha)$. Des

[1] La supériorité stricte provient de ce que nous ne sommes pas à l'optimum du problème original.

détails sur l'algorithme seront donnés au chapitre 8. Une itération s'effectue de la manière suivante :

$$\begin{aligned}\alpha^{k+1} &:= \alpha^k - \frac{\bar{\phi}'(\alpha^k)}{\bar{\phi}''(\alpha^k)}\\ &:= \alpha^k - \frac{(\kappa+2)b/(a+\alpha b) - 1/\alpha - \sum_{j=0}^{\kappa}(d_j/(1+\alpha d_j))}{-(\kappa+2)(b/(a+\alpha b))^2 + 1/\alpha^2 + \sum_{j=0}^{\kappa}(d_j/(1+\alpha d_j))^2},\end{aligned}$$

où $a = \tilde{c}^T\tilde{x}$ et $b = \tilde{c}^T\tilde{X}\tilde{d} - \gamma$.

La direction $\bar{q}$ proposée précédemment ne tient pas compte de l'objectif. On pourrait cependant l'intégrer. L'objectif intervient de la manière suivante : $\bar{c}^T\bar{x} = \overline{z}x_0 - c^Tx - \gamma x_\omega = \alpha$, avec $x_0 = 1$. En posant $\theta = \overline{z} - \alpha$, on peut définir la contrainte suivante $\hat{c}^T\bar{x} = 0$. En définissant $\hat{G} = \binom{\hat{c}^T}{\bar{G}}$, le problème se ramène au problème de réalisabilité étudié précédemment :

$$\begin{cases} \hat{G}\bar{x} &= 0\\ \bar{g}^T\bar{x} &= 1\\ \bar{x} &\geq 0.\end{cases}$$

Il est possible de monter que ceci permet de garder le saut de dualité constant. En effet, la direction $\bar{q}$ que l'on obtient se trouve dans le noyau de $\hat{G}\bar{X}$. Par conséquent, $\hat{c}^T\bar{X}\bar{q} = 0$. Après normalisation et en développant, on trouve : $\hat{c}^T\bar{x}^{k+1} = \frac{\hat{c}^T\bar{X}\bar{q}}{x_0q_0} = \theta - \frac{c^TXq}{x_0q_0} - \frac{\gamma x_w q_w}{x_0q_0} = 0$. Enfin, en substituant θ par sa valeur on trouve : $\bar{c}^T\bar{x}^{k+1} = \alpha$.

Une méthode plus raffinée [1] tient compte de l'objectif en le dissociant des contraintes. Elle est cependant beaucoup plus complexe et fait intervenir une recherche bidimensionnelle. Cette méthode n'ayant jamais fait l'objet d'une implantation informatique, nous laissons le lecteur se référer à [1] pour plus d'information.

6.4 Phase I - Phase II

Nous avons vu que l'algorithme projectif cherche le centre analytique de l'ensemble de localisation $\mathcal{L}_{\overline{z}}$. Pour que ce point puisse être défini, cet ensemble doit être compact. Initialement ce dernier n'est formé que par les boîtes. Si le premier appel à l'oracle[2] ne renvoie pas de coupes de d'optimalité, l'ensemble de localisation

[2] Le premier centre analytique est défini comme le point au milieu des boîtes. De par sa simplicité, il ne nécessite pas d'être calculé par l'algorithme projectif et peut être directement envoyé à l'oracle.

n'est pas compact et il n'est pas possible de définir un centre analytique[3]. Plus précisément, tant que l'oracle ne renvoie pas de coupes d'optimalité, cet ensemble n'est pas compact. L'algorithme ACCPM tel que nous l'avons décrit ne peut donc pas être utilisé. Il nous faut introduit une première phase (appelée Phase I) dont l'objectif est de pouvoir définir un ensemble de localisation compact.

Nous venons de voir que l'ensemble de localisation n'est compact que si une coupe d'optimalité a été introduite. Pour cela, le point y envoyé à l'oracle doit être réalisable. La phase I cherche donc à résoudre le problème de réalisabilité suivant : trouver un point y tel que $y \in Y$.

Durant la phase I la matrice G n'est formée que des boîtes et des coupes de réalisabilité :

$$G = \begin{pmatrix} I_d & -I_d & \Omega_J \end{pmatrix}.$$

Il est alors possible de définir l'ensemble de localisation compact suivant :

$$\mathcal{L} = \{y : G^T y \geq c\}.$$

Par définition, si Y a un intérieur non vide, $\mathcal{L}$ a un intérieur non vide. Il est donc possible de définir un centre analytique comme au chapitre 4 (4.11) :

$$\begin{array}{rl} \text{Minimiser} & \psi(y,s) = -\sum_{j=1}^{\kappa} \ln s_j \\ \text{s.c.} & G^T y - s = c, \qquad\qquad s > 0. \end{array}$$

Le système dual associé est le suivant :

$$\begin{array}{rcl} c^T x & = & 0 \\ Gx & = & 0 \\ x & > & 0. \end{array}$$

Théorème : 6.1 *Le système dual associé n'a pas de solution.*

Preuve. Montrons que pour tout point $\bar{y}$ appartenant à l'intérieur de l'ensemble de localisation $\mathcal{L}$, il n'est pas possible de trouver un point x satisfaisant le problème dual associé.

[3]Pour certains problèmes on connaît une borne supérieure $\overline{z}$ et une borne inférieure $\underline{z}$. L'ensemble de localisation est alors toujours compact et la phase I que nous allons décrir n'est pas nécessaire.

Soit $\bar{y}$ tel que $G^T\bar{y} > c$, pour tout point $x > 0$, nous avons $x^T(G^T\bar{y} - c) > 0$ ce qui nous interdit d'avoir $c^Tx = 0$ et $G^Tx = 0$ simultanément. ∎

Théorème : 6.2 *Pour tout point $x > 0$ tel que $Gx = 0$, nous avons $c^Tx < 0$.*

Preuve. De $x^T(G^T\bar{y} - c) > 0$ on déduit $c^Tx < 0$. ∎

Il est cependant possible d'appliquer, au problème dual, l'algorithme de réalisabilité développé à la section précédente. Nous avons vu que cet algorithme itère suivant la direction $d = Xq$ où $q = P_{\mathcal{N}\left(\binom{c^T}{G}X\right)}\mathbf{1}$.

On constate donc que très peu de modifications sont nécessaires, pour pouvoir transformer l'algorithme ACCPM que nous avons décrit, afin qu'il puisse être utilisé pour résoudre le problème de réalisabilité de la phase I.

Dans l'implantation que nous avons faite nous avons procédé de manière différente. Durant la Phase I nous avons cherché à résoudre le problème suivant :

$$\begin{aligned} \text{Minimiser} \quad & h(y) \\ \text{s.c.} \quad & l \leq y \leq h, \end{aligned}$$

où $h(y)$ est une fonction indicatrice convexe telle que : $h(y) > 0$ si $y \notin Y$ et $h(y) = 0$ sinon. Pour chaque problème traité, il faut donc trouver cette fonction indicatrice $h(y)$ qui donne une "mesure" de la distance séparant le point y de Y.

Par exemple si $Y = \{y : g(y) \leq 0\}$ avec $g(y)$ convexe, on peut prendre $h(y) = g(y)$.

En résolvant ce problème, on cherche à minimiser "l'infaisabilité" de la solution y par rapport aux contraintes. Plus prosaïquement, il s'agit de minimiser une fonction convexe non contrainte. Pour cela il est possible d'utiliser l'algorithme ACCPM tel que nous l'avons décrit. L'avantage de cette méthode vient du fait qu'elle ne nécessite aucune modification de l'algorithme ; par contre il nous faut, pour chaque problème traité, définir cette fonction $h(y)$.

On remarquera enfin que, bien qu'il soit possible de résoudre ce problème jusqu'à l'optimalité, cela ne présente pas d'intérêt. L'objectif est d'avoir un point y qui appartienne à l'ensemble Y. A partir du moment où l'on obtient un tel point, il est possible de passer à la phase II (algorithme ACCPM tel que nous l'avons décrit jusqu'ici) et d'optimiser le problème initial.

En résumé et compte tenu de ce qui a été dit précédemment, on peut faire les remarques suivantes :

- En utilisant une fonction indicatrice convexe h, la phase I et la phase II ne diffèrent que par le problème à résoudre et donc par l'oracle. L'algorithme tel que nous l'avons décrit peut être utilisé dans les deux phases.

- En résolvant le problème de réalisabilité, la phase I et la phase II travaillent sur le même problème et ont donc le même oracle. L'algorithme de la phase I n'est pas identique à celui de la phase II (algorithme que nous avons détaillé aux chapitres précedents). Cependant ils sont très voisins. La principale différence provient de la structure de la matrice G.

- En utilisant une fonction indicatrice convexe h, le problème de la phase I admet 0 comme borne inférieure.

- En utilisant une fonction indicatrice convexe h, si la valeur de la borne inférieure de la solution durant la phase I est strictement supérieure à 0, il n'est pas possible de trouver un point réalisable. On en déduit que le problème initial est infaisable.

- Si Y est composé d'un ensemble de contraintes, le problème à résoudre durant phase I peut être considéré comme la somme de $h_i(y)$ fonctions indicatrices convexes. Il est alors possible d'utiliser les techniques avancées développées au chapitre suivant.

6.5 Raffinement de la direction de Newton

Lorsque l'on calcule la direction de Newton, des imprécisions numériques peuvent apparaître. La direction peut ne plus être dans le noyau de $\tilde{G}\tilde{X}$ et le nouveau point $\tilde{x}^{k+1}$ ne satisfait plus alors la condition primordiale d'être dans le noyau de $\tilde{G}$. Si ces imprécisions numériques deviennent trop importantes ($\max\{\tilde{G}\tilde{x}\} > 10^{-8}$, 10^{-6} voir même beaucoup plus lorsque l'on est près de la solution) on constate que l'algorithme projectif piétine et ne permet plus de déterminer le centre analytique approché.

Nous avons vu que le calcul de la direction $\tilde{d}$ de Newton peut s'écrire $\tilde{d} = \tilde{X}\tilde{q}$ où $\tilde{q}$ est donnée par :

$$\tilde{q} = \mathbf{1} - \frac{\tilde{c}^T\tilde{x}}{\|P_{\mathcal{N}(\tilde{G}\tilde{X})}\tilde{X}\tilde{c}\|^2} P_{\mathcal{N}(\tilde{G}\tilde{X})}\tilde{X}\tilde{c}.$$

Si on fait abstraction de la projection, les calculs engendrés font partie de l'algèbre linéaire simple, c'est-à-dire qu'ils ne font intervenir que des produits vecteur-vecteur et matrice-vecteur. On peut faire l'hypothèse que l'erreur qu'ils engendrent est négligeable. Par contre, lors du calcul de la projection, un système linéaire est résolu :

$$u = -(\tilde{G}\tilde{X}^2\tilde{G}^T)^{-1}\tilde{G}\tilde{X}^2\tilde{c}.$$

Son calcul est sensible au conditionnement de matrice $\tilde{G}\tilde{X}^2\tilde{G}^T$ et peut générer d'importantes erreurs. Dans l'implantation que nous avons faite de l'agorithme nous avons utilisé la factorisation de Cholesky pour résoudre ce système. Il est clair qu'une factorisation plus stable (par exemple la factorisation QR) diminuerait l'erreur faite lors de la résolution du système. Cependant nous verrons plus tard que, pour des raisons de performance, nous lui avons préféré la factorisation de Cholesky. De plus, quelle que soit la factorisation, pour beaucoup de fonctions à optimiser, il aurait fallu intégrer une procédure de raffinement itératif (voir Golub et van Loan [4]).

Nous allons voir deux approches équivalentes pour estimer la correction à faire.

1. Dans une première approche, nous définissons le système suivant :

$$\begin{pmatrix} I_d & \tilde{X}\tilde{G}^T \\ \tilde{G}\tilde{X} & 0 \end{pmatrix} \begin{pmatrix} \tilde{p} \\ -u \end{pmatrix} = \begin{pmatrix} \tilde{X}\tilde{c} \\ 0 \end{pmatrix}. \tag{6.4}$$

Ce système est appellé : **système augmenté**. On vérifie facilement qu'il permet d'obtenir la projection recherchée. En effet, il s'écrit :

$$\begin{aligned} \tilde{p} - \tilde{X}\tilde{G}^T u &= \tilde{X}\tilde{c} \\ \tilde{G}\tilde{X}\tilde{p} &= 0. \end{aligned} \tag{6.5}$$

En prémultipliant la première équation par $\tilde{G}\tilde{X}$, le premier terme s'annule et on obtient $u = -(\tilde{G}\tilde{X}^2\tilde{G}^T)^{-1}\tilde{G}\tilde{X}^2\tilde{c}$. En substituant u dans (6.5), on obtient :

$$\tilde{p} = \tilde{X}\tilde{c} - \tilde{X}\tilde{G}^T(\tilde{G}\tilde{X}^2\tilde{G}^T)^{-1}\tilde{G}\tilde{X}^2\tilde{c} = P_{\mathcal{N}(\tilde{G}\tilde{X})}\tilde{X}\tilde{c}.$$

Le système augmenté permet donc bien de déterminer la projection recherchée. l'objectif est donc de résoudre ce système avec le maximum de précision.

Soient e_1 et e_2 les erreurs faites sur les composantes du système augmenté :

$$\begin{pmatrix} e_1 \\ e_2 \end{pmatrix} = \begin{pmatrix} \tilde{X}\tilde{c} \\ 0 \end{pmatrix} - \begin{pmatrix} I_d & \tilde{X}\tilde{G}^T \\ \tilde{G}\tilde{X} & 0 \end{pmatrix} \begin{pmatrix} \tilde{p}_e \\ -u_e \end{pmatrix}. \tag{6.6}$$

où $\tilde{p}_e$ et u_e sont les vecteurs obtenus lorsque l'on résoud le système (6.4). Si aucune erreur n'était faite nous aurions $\tilde{p}_e = \tilde{p}$ et $u_e = u$.

Dans la pratique, l'erreur e_1 est négligeable (à la précision machine près). Ceci ne doit guère nous étonner. En effet, la première équation de (6.6) s'écrit :

$$e_1 = \tilde{X}\tilde{c} - \tilde{p}_e + \tilde{X}\tilde{G}^T u_e.$$

Comme $\tilde{p}_e = \tilde{X}\tilde{G}^T u_e + \tilde{X}\tilde{c}$, on en déduit que son calcul à partir de u_e est relativement stable. L'erreur sur $\tilde{p}_e$ est donc décrite par e_2.

En définissant $\tilde{p}_c$ et u_c, la correction sur les vecteurs $\tilde{p}_e$ et u_e respectivement, on obtient le système suivant :

$$\begin{pmatrix} I_d & \tilde{X}\tilde{G}^T \\ \tilde{G}\tilde{X} & 0 \end{pmatrix} \begin{pmatrix} \tilde{p}_e + \tilde{p}_c \\ -u_e - u_c \end{pmatrix} \equiv \begin{pmatrix} \tilde{X}\tilde{c} \\ 0 \end{pmatrix}. \tag{6.7}$$

On en déduit :

$$\begin{pmatrix} I_d & \tilde{X}\tilde{G}^T \\ \tilde{G}\tilde{X} & 0 \end{pmatrix} \begin{pmatrix} \tilde{p}_c \\ -u_c \end{pmatrix} \equiv \begin{pmatrix} e_1 \\ e_2 \end{pmatrix}. \tag{6.8}$$

La première équation de (6.8) s'écrit : $\tilde{p}_c - \tilde{X}\tilde{G}^T u_c = e_1$. En prémultipliant par $\tilde{G}\tilde{X}$ on obtient :

$$u_c = (\tilde{G}\tilde{X}^2\tilde{G}^T)^{-1}(\tilde{G}\tilde{X}\tilde{p}_c - \tilde{G}\tilde{X}e_1) = (\tilde{G}\tilde{X}^2\tilde{G}^T)^{-1}(e_2 - \tilde{G}\tilde{X}e_1), \tag{6.9}$$

et on déduit l'erreur sur la projection :

$$\tilde{p}_c = e_1 + \tilde{X}\tilde{G}^T u_c.$$

De (6.6) il est possible de calculer e_1 et e_2 et de (6.9) de calculer u_c. Le calcul de la correction $\tilde{p}_c$ et u_c est donc réalisable. Si on pouvait avoir une solution exacte du système (6.8), on pourrait calculer $\tilde{p} = \tilde{p}_e + \tilde{p}_c$ et $u = u_e + u_c$. En réalité, on ne va pas pouvoir corriger complètement $\tilde{p}_e$ et u_e. Cependant, Bjorck [2] montre[4] que le nombre de chiffres significatifs de $\tilde{p}$ est multiplié par deux, c'est-à-dire que l'erreur faite sur la projection décroît avec le carré.

[4]Pour autant que l'on calcule e_1 et e_2 avec une précision double de celle utilisée pour calculer q. Dans la pratique on se passe de cette exigence tout en constatant de bons résultats.

2. Dans une deuxième approche, on peut écrire que la projection est le résultat d'une mauvaise factorisation. Pour résoudre le système nous avons utilisé la factorisation de Cholesky[5]. Nous devrions donc avoir $\tilde{G}\tilde{X}^2\tilde{G}^T = LL^T$. Cependant, comme la factorisation est mauvaise nous avons :

$$\tilde{G}\tilde{X}^2\tilde{G}^T = LL^T + E, \tag{6.10}$$

où E est la matrice des erreurs.

Posons $\bar{E} = L^{-1}EL^{-T}$. On a donc $L\bar{E}L^T = E$ et on peut écrire (6.10) de la manière suivante :

$$\begin{aligned} \tilde{G}\tilde{X}^2\tilde{G}^T &= LL^T + L\bar{E}L^T \\ &= L(I_d + \bar{E})L^T, \end{aligned}$$

où I_d est la matrice identité.

Par définition, E est symétrique et par conséquent $\bar{E}$ l'est aussi. Dans le calcul de la projection, c'est l'inverse de cette matrice que nous utilisons, c'est-à-dire $(\tilde{G}\tilde{X}^2\tilde{G}^T)^{-1} = L^{-T}(I_d + \bar{E})^{-1}L^{-1}$. Si l'on fait l'hypothèse raisonnable que $\bar{E} \ll I_d$, alors $(I_d + \bar{E})^{-1} = I_d - \bar{E} + O(\bar{E}^2)$. On peut donc approximer l'inverse de $\tilde{G}\tilde{X}^2\tilde{G}^T$ par :

$$\begin{aligned} (\tilde{G}\tilde{X}^2\tilde{G}^T)^{-1} &= L^{-T}(I - \bar{E} + O(\bar{E}^2))L^{-1} \\ &= L^{-T}L^{-1} - L^{-T}\bar{E}L^{-1} + O(\bar{E}^2) \\ &= L^{-T}L^{-1} - D + O(D^2), \end{aligned}$$

où $D = L^{-T}\bar{E}L^{-1}$ est aussi une matrice symétrique.

Lors du calcul de la projection, au lieu d'obtenir :

$$\begin{aligned} P_{\mathcal{N}(\tilde{G}\tilde{X})}\tilde{X}\tilde{c} &= \tilde{X}\tilde{c} - \tilde{X}\tilde{G}^T(\tilde{G}\tilde{X}^2\tilde{G}^T)^{-1}\tilde{G}\tilde{X}^2\tilde{c} \\ &= \tilde{X}\tilde{c} - \tilde{X}\tilde{G}^T(L^{-T}L^{-1} - D + O(D^2))\tilde{G}\tilde{X}^2\tilde{c}, \end{aligned} \tag{6.11}$$

nous calculons le vecteur :

$$\tilde{p}_e = \tilde{X}\tilde{c} - \tilde{X}\tilde{G}^T L^{-T}L^{-1}\tilde{G}\tilde{X}^2\tilde{c}. \tag{6.12}$$

On peut récrire (6.11) ainsi :

$$P_{\mathcal{N}(\tilde{G}\tilde{X})}\tilde{X}\tilde{c} = \tilde{p}_e + \tilde{X}\tilde{G}^T D\tilde{G}\tilde{X}^2\tilde{c} + O(D^2). \tag{6.13}$$

[5]Nous discuterons du choix de la factorisation de Cholesky dans le dernier chapitre.

En multipliant par $\tilde{G}\tilde{X}$, comme $P_{\mathcal{N}(\tilde{G}\tilde{X})}\tilde{X}\tilde{c}$ est la projection de $\tilde{X}\tilde{c}$ dans l'espace nul de $\tilde{G}\tilde{X}$, on obtient :

$$\tilde{G}\tilde{X}\tilde{p}_e = -\tilde{G}\tilde{X}^2\tilde{G}^T D\tilde{G}\tilde{X}^2 c - O(D^2), \tag{6.14}$$

où $\tilde{G}\tilde{X}\tilde{p}_e$ correspond au résidu de la projection calculé $\tilde{p}_e$.

Le vecteur $\tilde{p}_e$ n'est donc pas la projection recherchée puisqu'il ne se trouve pas réellement dans le noyau de $\tilde{G}\tilde{X}$. Afin d'obtenir un meilleur vecteur $\tilde{p}_e$ on le projette de nouveau. On obtient alors :

$$\bar{P}_{\mathcal{N}(\tilde{G}\tilde{X})}\tilde{p}_e = \tilde{p}_e - \tilde{X}\tilde{G}^T(\tilde{G}\tilde{X}^2\tilde{G}^T)^{-1}\tilde{G}\tilde{X}\tilde{p}_e, \tag{6.15}$$

et en substituant la dernière occurence de $\tilde{p}_e$ par sa valeur (6.12) :

$$\begin{aligned} \bar{P}_{\mathcal{N}(\tilde{G}\tilde{X})}\tilde{p}_e &= \tilde{p}_e + \tilde{X}\tilde{G}^T(\tilde{G}\tilde{X}^2\tilde{G}^T)^{-1}\tilde{G}\tilde{X}^2\tilde{G}^T D\tilde{G}\tilde{X}^2 c + O(D^2) \\ &= \tilde{p}_e + \tilde{X}\tilde{G}^T D\tilde{G}\tilde{X}^2 c + O(D^2). \end{aligned} \tag{6.16}$$

En comparant cette dernière écriture de $\bar{P}_{\mathcal{N}(\tilde{G}\tilde{X})}\tilde{p}_e$ avec (6.13), on constate que $\bar{P}_{\mathcal{N}(\tilde{G}\tilde{X})}\tilde{p}_e$ est la projection recherchée $P_{\mathcal{N}(\tilde{G}\tilde{X})}\tilde{X}\tilde{c}$. Cependant, (6.16) ne permet pas de calculer $\bar{P}_{\mathcal{N}(\tilde{G}\tilde{X})}\tilde{p}_e$ directement puisque D et D^2 sont inconnus. De manière générale on ne peut calculer cette projection, que ce soit par (6.15) ou par (6.16). On l'approxime alors en remplaçant $\tilde{G}\tilde{X}^2\tilde{G}^T$ par LL^T. Appelons $\tilde{p}_c$ le vecteur calculé :

$$\tilde{p}_c = \tilde{p}_e - \tilde{X}\tilde{G}^T L^{-T} L^{-1}\tilde{G}\tilde{X}\tilde{p}_e.$$

En appliquant le même raisonnement que pour le calcul théorique de la projection de $\tilde{X}\tilde{c}$, de (6.13) on obtient la projection théorique de $P_{\mathcal{N}(\tilde{G}\tilde{X})}\tilde{p}_e$:

$$P_{\mathcal{N}(\tilde{G}\tilde{X})}\tilde{p}_e = \tilde{p}_c + \tilde{X}\tilde{G}^T D\tilde{G}\tilde{X}\tilde{p}_e + O(D^2).$$

En s'appuyant sur (6.14), il est possible de substituer $\tilde{G}\tilde{X}\tilde{p}_e$ dans l'équation ci-dessus et d'obtenir :

$$P_{\mathcal{N}(\tilde{G}\tilde{X})}\tilde{p}_e = \tilde{p}_c - \tilde{X}\tilde{G}^T D\tilde{G}\tilde{X}^2\tilde{G}^T D\tilde{G}\tilde{X}^2 c + O(D^2).$$

Le résidu de cette nouvelle projection est alors :

$$\begin{aligned} \tilde{G}\tilde{X}\tilde{p}_c &= \tilde{G}\tilde{X}^2\tilde{G}^T D\tilde{G}\tilde{X}^2\tilde{G}^T D\tilde{G}\tilde{X}^2 c + O(D^2) \\ &= (\tilde{G}\tilde{X}^2\tilde{G}^T D)^2\tilde{G}\tilde{X}^2 c + O(D^2) \\ &= O(D^2). \end{aligned}$$

En résumé, en projetant deux fois le vecteur $\tilde{X}\tilde{c}$ on n'obtient pas la projection recherchée mais l'erreur faite sur le résidu a décru avec le carré D.

Il est intéressant de constater que ces approches sont équivalentes. En effet, si dans la première on néglige l'erreur e_1, la projection du vecteur $\tilde{X}\tilde{c}$ s'écrit :

$$\begin{aligned}
\bar{P}_{\mathcal{N}(\tilde{G}\tilde{X})}\tilde{X}\tilde{c} &= \tilde{p}_e + \tilde{p}_c \\
&= \tilde{p}_e + \tilde{X}\tilde{G}^T u_{\varepsilon^2} \\
&= \tilde{p}_e + \tilde{X}\tilde{G}^T(\tilde{G}\tilde{X}^2\tilde{G}^T)^{-1}e_2 \\
&= \tilde{p}_e - \tilde{X}\tilde{G}^T(\tilde{G}\tilde{X}^2\tilde{G}^T)^{-1}\tilde{G}\tilde{X}\tilde{p}_e,
\end{aligned}$$

qui correspond à la projection du vecteur $\tilde{p}_e$ dans l'espace nul de $\tilde{G}\tilde{X}$, c'est-à-dire à (6.15) dans l'approche précédente.

6.6 Procédure zéro-machine

Nous avons vu dans la section précédente comment corriger la direction de Newton. L'objectif est de s'assurer que le nouveau point $\tilde{x}$ satifait $\tilde{G}\tilde{x} = 0$, condition de première importance pour l'algorithme projectif. Nous avons vu cependant que si cette correction améliore les calculs elle n'est pas parfaite. Il se peut qu'après avoir exécuté le pas de Newton (c.à.d. $\tilde{x}^{k+1} = (\tilde{x}^k + \bar{\alpha}\tilde{d})/(1 + \tilde{d}_0)$) le nouveau point $\tilde{x}^{k+1}$ ne soit pas dans le noyau de $\tilde{G}$.

La procédure zéro-machine agit sur le vecteur $\tilde{x}^{k+1}$ et garantit que $\tilde{G}\tilde{x}^{k+1}$ soit nul (à la précision machine près).

L'objectif est donc de mettre $\tilde{x}$ dans le noyau de $\tilde{G}$. Soit $\tilde{G}\tilde{x} = e$ l'erreur faite. L'écriture complète est :

$$\begin{pmatrix} 0 & I_d & -I_d & \Omega_J & \Omega_I \\ -1 & 0 & 0 & 0 & \mathbf{1}^T \end{pmatrix} \begin{pmatrix} x_0 \\ \zeta \\ \eta \\ \varrho \\ \varsigma \end{pmatrix} = \begin{pmatrix} e_1 \\ \vdots \\ e_{n+1} \end{pmatrix}. \tag{6.17}$$

On effectue alors les corrections suivantes :

1. On calcule tout d'abord le scalaire $\delta = \mathbf{1}^T\varsigma$ pour en diviser toutes les variables

de ς. On assure ainsi que $e_{n+1} = 0$. L'opération à effectuer est donc :

$$\varsigma := \frac{\varsigma}{\mathbf{1}^T \varsigma}.$$

2. Après avoir terminé cette première étape, on calcule $\tilde{G}\tilde{x} = e$. Ensuite, pour toutes les composantes de e, si $e_i > 0$ alors on ajoute $|e_i|$ à η_i sinon, on ajoute $|e_i|$ à ζ_i. On doit donc effectuer, pour toutes les composantes de e, la correction :

   Si $e_i > 0$, **alors**
   $\eta_i := \eta_i + |e_i|$;
   Sinon
   $\zeta_i := \zeta_i + |e_i|$;
   Fsi.

Il est intéressant de constater l'action des contraintes de boîte (et de leurs variables η et ζ associées) dans cette procédure. Si les boîtes l et h permettent de borner la localisation de x^* et donc de définir un ensemble de localisation initial, elles sont aussi utiles pour forcer $\tilde{x}$ à être dans le noyau de $\tilde{G}$. Elles présentent donc un double intérêt.

La procédure zéro-machine est très efficace. Malheureusement, de par sa manière brutale d'opérer, elle ne garantit pas une décroissance de $\tilde{\varphi}(\tilde{x})$. Si les instabilités numériques sont trop importantes son action perturbe trop le module projectif et celui-ci ne parvient plus à déterminer un centre analytique. Dans la pratique nous n'avons que rarement été confrontés à ce problème.

Bibliographie

[1] O. Bahn, O. du Merle, J.-L. Goffin et J.-P. Vial, “A cutting plane method from analytic centers for stochastic programming”, à paraître dans *Mathematical programming.*

[2] A. Bjorck, “Iterative Refinement of linear least squares solution II”, *BIT* **8** (1968), 8-30.

[3] G. de Ghellinck et J.-P. Vial, “A polynomial Newton method for linear programming”, *Algorithmica* **1** (1986), 425-453.

[4] G. H. Golub and C. van Loan, *Matrix Computations*, (2nd ed.) The Johns Hopkins University Press, Baltimore and London, 1989.

[5] J. E. Mitchell, *Karmarkar's algorithm and combinatorial optimization problems*, (Faculty of the Graduate School of Cornell University, 1988).

[6] J. P. Vial, “A unified approach to projective algorithms for linear programming”, in *Lecture Notes in Mathematics 1405* , S. Verlag, ed., Proceedings of Fifth French-German Conference held in (Castel Novel, 1988), 191-220.

Partie III

IMPLANTATION DE L'ALGORITHME ACCPM

Chapitre 7

Techniques avancées pour la mise en œuvre de ACCPM

Dans les chapitres précédents nous avons décrit les procédures indispensables à mettre en œuvre pour pouvoir implanter l'algorithme ACCPM. Cependant, pour améliorer les performances de l'algorithme, quelques raffinements peuvent être apportés. Ces différentes techniques vont faire l'objet de ce chapitre.

Nous allons tout d'abord aborder le cas où l'oracle renvoie plusieurs coupes, puis nous verrons comment limiter au maximum la taille de la matrice $\tilde{G}$. Nous verrons aussi comment gérer dynamiquement les boîtes, comment initialiser l'algorithme et enfin nous donnerons l'algorithme ACCPM complet.

7.1 Multicoupes

Jusqu'ici, nous avons considéré que l'oracle ne renvoyait qu'une seule coupe à chaque appel. Nous allons maintenant analyser le cas où plusieurs plans sont récupérés par le programme-maître.

Dans de nombreux cas, la fonction f est constituée de la somme de fonctions convexes dont l'oracle est séparable[1]. Ce dernier envoie alors autant coupes que de fonctions composant f. De même, l'ensemble Y peut être défini comme l'intersection de T'

[1] Le calcul des valeurs et des sous-gradiants des différentes composantes de f peut être mené séparément.

ensembles convexes $Y_{t'}$. L'oracle ne renvoie alors pas 1 plan séparant y^c de Y mais T' plans (ou moins si $y^c \in Y_{t'}$ pour certains t') séparant y^c des ensembles $Y_{t'}$.

Le problème initial (4.1) peut être reformulé de la manière suivante :

$$\begin{aligned} \text{Minimiser} \quad & f(y) = \sum_{t=1}^{m} f_t(y) \\ \text{s.c.} \quad & y \in \cap_1^{m'} Y_{t'} \\ & l \leq y \leq h, \end{aligned} \tag{7.1}$$

où les fonctions $f_t : \mathbb{R}^n \to \mathbb{R}$ et les ensembles $Y_{t'}$ sont convexes et où l et h sont les bornes (c.à.d. contraintes de boîtes) sur y.

Cette classe de problèmes est une sous-classe des problèmes traités jusqu'à présent. Nous verrons qu'elle contient de nombreux problèmes classiques (décomposition, programmation stochastique, multiflot). Il apparaît clairement que la méthode ACCPM telle que nous l'avons décrite permet de les résoudre, cependant en différenciant l'information provenant des fonctions f_t ou des ensembles $Y_{t'}$, on améliore la qualité de l'approximation linéaire du problème, ce qui induit une convergence plus rapide de l'algorithme.

Le problème (7.1) est équivalent au problème suivant :

$$\begin{aligned} \text{Minimiser} \quad & f(y) = \sum_{t=1}^{m} z_t \\ \text{s.c.} \quad & z_t \geq \sum_{t=1}^{m} f_t(y), && t \in T = \{1, \cdots, m\} \\ & y \in \cap Y_{t'}, && t' \in T' = \{1, \cdots, m'\} \\ & l \leq y \leq h. \end{aligned} \tag{7.2}$$

En définissant les sous-ensembles I_t de I pour chaque $t \in T$, l'approximation linéaire (4.6) se récrit :

$$\begin{aligned} \text{Minimiser} \quad & z_1 + \cdots + z_m \\ \text{s.c.} \quad & z_1 \geq f_1(y; y^i), && i \in I_1 \\ & \quad \vdots && \quad \vdots \\ & z_m \geq f_m(y; y^i), && i \in I_m \\ & g_{t'}(y; y^j) \leq 0, && j \in J,\ t' \in T' \\ & l \leq y \leq h. \end{aligned} \tag{7.3}$$

A chaque appel à l'oracle, le programme-maître peut recevoir autant de plans de support qu'il y a de fonctions f_t, et autant de plans séparateurs qu'il y a d'ensembles $Y_{t'}$. Dans les implantations que nous avons faites de cet algorithme, nous n'avons jamais récupéré l'information sur l'objectif si le point y^c n'appartient pas à tous les ensembles $Y_{t'}$, considérant qu'elle est inexploitable puisqu'en dehors de l'ensemble de définition. Rien cependant ne l'interdit.

Le fait de prendre en compte de façon différenciée l'information provenant des fonctions f_t et des ensembles $Y_{t'}$ ne change pratiquement rien à ce qui a été vu jusqu'a présent. Pour l'algorithme ACCPM la différence consiste à pouvoir gérer l'introduction de plusieurs formes linéaires à chaque itération. Plus exactement nous allons voir que les changements interviennent à deux niveaux : sur la stucture de $\tilde{G}$ et au moment de l'intégration dans celle-ci des coupes renvoyées par l'oracle.

7.1.1 Structure de la matrice $\tilde{G}$

Le problème $\mathcal{D}$ s'écrit maintenant :

$$\begin{array}{rl} \text{Minimiser} & z_1 + \cdots + z_{m'} \\ \text{s.c.} & \Omega_J^T y \geq \gamma_J \\ & \Omega_{I_1}^T y + z_1 \mathbf{1} \geq \gamma_{I_1} \\ & \quad\vdots \\ & \Omega_{I_m}^T y + z_m \mathbf{1} \geq \gamma_{I_m} \\ & l \leq y \leq h, \end{array} \tag{7.4}$$

où, comme dans (4.7), la matrice Ω_J^T et le vecteur γ_J contiennent les coupes de réalisabilité et les matrices Ω_I^T et les vecteurs γ_I les coupes d'optimalité.

En posant les notations suivantes :

$$u = \begin{pmatrix} y \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \quad c = \begin{pmatrix} l \\ -h \\ \gamma_J \\ \gamma_I \end{pmatrix}, \quad b = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ \vdots \\ 1 \end{pmatrix}, \quad x = \begin{pmatrix} \zeta \\ \eta \\ \varrho \\ \varsigma \end{pmatrix},$$

et

$$G = \begin{pmatrix} I_d & -I_d & \Omega_J & \Omega_{I_1} & \Omega_{I_2} & \cdots & \Omega_{I_m} \\ 0 & 0 & 0 & \mathbf{1}_1^T & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1}_2^T & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \cdots & \mathbf{1}_m^T \end{pmatrix},$$

on retrouve l'écriture du problème $\mathcal{P}$ (4.9) :

$$\begin{aligned} \text{Maximiser} \quad & c^T x \\ \text{s.c.} \quad & Gx = b \\ & x \geq 0, \end{aligned}$$

et de son dual :

$$\begin{aligned} \text{Minimiser} \quad & b^T u \\ \text{s.c.} \quad & G^T u - s = c \\ & s \geq 0. \end{aligned}$$

Il apparaît clairement que seule la structure de la matrice $\tilde{G}$ change. Il y a dans cette nouvelle matrice autant de lignes de convexité que de fonctions convexes dans l'objectif. Tous les calculs qui ont été développés peuvent donc être réappliqués directement.

Les tests que nous avons faits ont montré que la différenciation de l'information provenant de l'objectif et des contraintes permettait effectivement de converger en moins d'itérations vers l'optimum. Cependant la matrice $(\tilde{G}\tilde{X}^2\tilde{G}^T)$ est maintenant de dimension $(n + m) \times (n + m)$. Il en résulte que le calcul de sa factorisation est plus long. Or, comme elle était déjà l'opération dans le programme-maître qui consommait le plus de temps, il faut, pour que cette variante de l'algorithme soit rentable, que le gain fait sur le nombre d'itérations ne soit pas consommé par la perte occasionnée par la factorisation. Heureusement, nous allons voir qu'il est possible d'exploiter la structure de cette matrice pour minimiser l'effort supplémentaire du à la factorisation et donc d'assurer que la différentiation de l'information soit une variante qui améliore pratiquement toujours les performances de l'algorithme ACCPM.

7.1.2 Factorisation de la matrice $\tilde{G}\tilde{X}^2\tilde{G}^T$

La matrice $\tilde{G}\tilde{X}$ a la structure :

$$\tilde{G}\tilde{X} = \begin{pmatrix} \vdots & \hat{I}_d & -\hat{I}_d & \hat{\Omega}_J & \hat{\Omega}_{I_1} & \hat{\Omega}_{I_2} & \cdots & \hat{\Omega}_{I_m} \\ \vdots & 0 & 0 & 0 & \hat{\mathbf{1}}_1^T & 0 & \cdots & 0 \\ -\hat{b} & 0 & 0 & 0 & 0 & \hat{\mathbf{1}}_2^T & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \vdots & 0 & 0 & 0 & 0 & 0 & \cdots & \hat{\mathbf{1}}_m^T \end{pmatrix},$$

où le chapeau indique que la composante a été multipliée par la sous-matrice de $\tilde{X}$ correspondante. Cette matrice a plein rang.

La matrice $\tilde{G}\tilde{X}^2\tilde{G}^T$ s'écrit :

$$\tilde{G}\tilde{X}^2\tilde{G}^T = \begin{pmatrix} 2\hat{I}_d^2 + \hat{\Omega}_J\hat{\Omega}_J^T + \sum_{t=1}^m \hat{\Omega}_{I_t}\hat{\Omega}_{I_t}^T & \hat{\Omega}_{I_1}\hat{\mathbf{1}}_1 & \hat{\Omega}_{I_2}\hat{\mathbf{1}}_2 & \cdots & \hat{\Omega}_{I_m}\hat{\mathbf{1}}_m \\ \hat{\mathbf{1}}_1^T\hat{\Omega}_{I_1}^T & \hat{\mathbf{1}}_1^T\hat{\mathbf{1}}_1 & 0 & \cdots & 0 \\ \hat{\mathbf{1}}_2^T\hat{\Omega}_{I_2}^T & 0 & \hat{\mathbf{1}}_2^T\hat{\mathbf{1}}_2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \hat{\mathbf{1}}_m^T\hat{\Omega}_{I_m}^T & 0 & 0 & \cdots & \hat{\mathbf{1}}_m^T\hat{\mathbf{1}}_m \end{pmatrix} + \hat{b}\hat{b}^T.$$

En posant :

$$\begin{aligned} A &= 2\hat{I}_d^2 + \hat{\Omega}_J\hat{\Omega}_J^T + \sum_{t=1}^m \hat{\Omega}_{I_t}\hat{\Omega}_{I_t}^T, \\ B &= (\,\hat{\Omega}_{I_1}\hat{\mathbf{1}}_1 \quad \hat{\Omega}_{I_2}\hat{\mathbf{1}}_2 \quad \cdots \quad \hat{\Omega}_{I_m}\hat{\mathbf{1}}_m\,), \end{aligned}$$

et

$$C = \begin{pmatrix} \hat{\mathbf{1}}_1^T\hat{\mathbf{1}}_1 & 0 & \cdots & 0 \\ \vdots & \hat{\mathbf{1}}_2^T\hat{\mathbf{1}}_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \hat{\mathbf{1}}_m^T\hat{\mathbf{1}}_m \end{pmatrix},$$

la matrice $\tilde{G}\tilde{X}^2\tilde{G}^T$ se récrit :

$$\tilde{G}\tilde{X}^2\tilde{G}^T = \begin{pmatrix} A & B \\ B^T & C \end{pmatrix} + \hat{b}\hat{b}^T.$$

Posons :

$$\bar{G} = \begin{pmatrix} A & B \\ B^T & C \end{pmatrix}.$$

En appliquant le complément de Schur, on obtient :

$$(\tilde{G}\tilde{X}^2\tilde{G}^T)^{-1} = \bar{G}^{-1} - \frac{\bar{G}^{-1}\hat{b}\hat{b}^T\bar{G}^{-1}}{1+\hat{b}^T\bar{G}^{-1}\hat{b}}.$$

L'objectif est de pouvoir calculer $x = \bar{G}^{-1}y$. Comme la matrice $\bar{G}$ est symétrique définie positive et a plein rang, il est possible d'en calculer directement la factorisation de Cholesky. Cependant, nous avons vu que sa dimension est alors $(n+m) \times (n+m)$. Nous allons voir ci-dessous comment organiser les calculs pour minimiser l'effort dû à la différenciation de l'information provenant de l'objectif.

L'astuce consiste à calculer de façon adéquate la factorisation de Cholesky, c'est-à-dire à déterminer une matrice triangulaire M telle que $\bar{G} = MM^T$.

En posant tout d'abord $P = \begin{pmatrix} 0 & I_d \\ I_d & 0 \end{pmatrix}$, la matrice $\bar{G}$ s'écrit $\bar{G} = PLL^TP^T$ avec LL^T :

$$LL^T = \begin{pmatrix} C & B^T \\ B & A \end{pmatrix}.$$

La matrice C étant diagonale et strictement positive, sa factorisation sous la forme $\Lambda\Lambda^T$, avec Λ diagonale, est évidente. De même Λ^{-1} et Λ^{-T} sont définis et triviaux. LL^T s'écrit alors :

$$LL^T = \begin{pmatrix} \Lambda & 0 \\ 0 & I_d \end{pmatrix} \begin{pmatrix} I_d & \Lambda^{-1}B^T \\ B\Lambda^{-T} & A \end{pmatrix} \begin{pmatrix} \Lambda^T & 0 \\ 0 & I_d \end{pmatrix}.$$

De plus,

$$\begin{pmatrix} I_d & \Lambda^{-1}B^T \\ B\Lambda^{-T} & A \end{pmatrix} = \begin{pmatrix} I_d & 0 \\ B\Lambda^{-T} & I_d \end{pmatrix} \begin{pmatrix} I & 0 \\ 0 & A - B\Lambda^{-T}\Lambda^{-1}B^T \end{pmatrix} \begin{pmatrix} I_d & 0 \\ B\Lambda^{-T} & I_d \end{pmatrix}.$$

Prouvons alors le théorème suivant :

Théorème : 7.1 [1]. *$A - BC^{-1}B^T$ est symétrique définie positive.*

Preuve. Comme $\bar{G}$ est définie positive, pour x et y non nuls de dimensions appropriées :

$$f(x,y) = x^TAx + 2x^TBy + y^TCy > 0.$$

En prenant $y = -C^{-1}B^Tx$ on obtient pour tout x non nul :

$$\begin{aligned} f(x, -C^{-1}B^Tx) &= x^TAx - 2x^TBC^{-1}B^Tx + x^TBC^{-T}CC^{-1}B^Tx \\ &= x^TAx - x^TBC^{-1}B^Tx \\ &= x^T(A - BC^{-1}B^T)x. \end{aligned}$$

Comme $f(x,y)$ est strictement positive pour tout x et y non nuls, $x^T(A-BC^{-1}B^T)x$ l'est aussi.

Comme de plus $A - BC^{-1}B^T$ à la forme suivante :

$$A - BC^{-1}B^T = 2\hat{I}_d^2 + \hat{\Omega}_J\hat{\Omega}_J^T + \sum_{t'=1}^{m} \hat{\Omega}_{I_{t'}}\hat{\Omega}_{I_t}^T - \sum_{t'=1}^{m} \frac{\hat{\Omega}_{I_t}\hat{\mathbf{1}}\hat{\mathbf{1}}^T\hat{\Omega}_{I_t}^T}{\hat{\mathbf{1}}^T\hat{\mathbf{1}}},$$

elle est symétrique. Cette matrice est donc symétrique définie positive. ∎

Comme $A-B\Lambda^{-T}\Lambda^{-1}B^T = A-BC^{-1}B^T$ et comme nous avons vu que $A-BC^{-1}B^T$ est symétrique définie positive, il est possible de la factoriser par Cholesky. Soit $A - B\Lambda^{-T}\Lambda^{-1}B^T = \Delta\Delta^T$. On obtient alors L :

$$L = \begin{pmatrix} \Lambda & 0 \\ 0 & I_d \end{pmatrix} \begin{pmatrix} I_d & 0 \\ B\Lambda^{-T} & I_d \end{pmatrix} \begin{pmatrix} I_d & 0 \\ 0 & \Delta \end{pmatrix} = \begin{pmatrix} \Lambda & 0 \\ B\Lambda^{-T} & \Delta \end{pmatrix}.$$

La factorisation de $\bar{G}$ est donc la suivante :

$$M = \begin{pmatrix} B\Lambda^{-T} & \Delta \\ \Lambda & 0 \end{pmatrix}.$$

Enfin, on remarquera qu'il est possible de réorganiser la formule de Shermann-Morisson-Woodbury afin de minimiser les calculs :

$$(\tilde{G}\tilde{X}^2\tilde{G}^T)^{-1} = M^{-T}M^{-1} - \frac{M^{-T}M^{-1}\hat{b}\hat{b}^T M^{-T}M^{-1}}{1+\hat{b}^T M^{-T}M^{-1}\hat{b}} \quad (7.5)$$

$$= M^{-T}\left(I_d - \frac{\tau\tau^T}{1+\tau^T\tau}\right)M^{-1}, \quad (7.6)$$

avec $\tau = M^{-1}\hat{b}$.

En conclusion, dans la méthode proposée ci-dessus, l'effort de calcul n'intervient que lors de la factorisation de $A - BC^{-1}B^T$. Cet effort est réduit par rapport à une factorisation[2] directe de $\tilde{G}\tilde{X}^2\tilde{G}^T$ puisque sa dimension est de $n \times n$ et non de $(n+m)\times(n+m)$. Nous verrons que pour certains problèmes (problèmes stochastiques, problèmes de multiflots...) m est souvent beaucoup plus important que n. De plus, si l'on conduit les calculs de cette manière on réduit considérablement la place mémoire nécessaire pour mémoriser la factorisation. Seule la matrice Δ de dimension $n \times n$ doit être mémorisée, le calcul de Λ et $B\Lambda^{-T}$ étant immédiat.

[2]La complexité de la factorisation de Cholesky d'une matrice $n \times n$ est : $O(1/3n^3)$.

7.1.3 Adjonction de plusieurs colonnes

Nous avons vu que si on tient compte de façon différenciée de l'information provenant des diverses fonctions f_t et ensembles $Y_{t'}$, cela ne faisait que changer la structure de la matrice $\tilde{G}$; les calculs dans le programme-maître pouvant être menés de manière identique au cas général étudié précédemment. De plus, nous venons de voir comment exploiter de façon adéquate la nouvelle structure de $\tilde{G}$ lors de la factorisation de $\tilde{G}\tilde{X}\tilde{G}^T$. Il reste cependant un point à éclaircir. Comment intégrer un ensemble de colonnes dans la matrice $\tilde{G}$?

La section § 6.3 décrit comment intégrer une colonne. Dans le cas où nous avons plusieurs colonnes à ajouter, il suffit de les agréger et d'appliquer la même méthode. Après avoir déterminé le niveau auquel cette colonne agrégée doit être introduite il ne reste plus qu'à la désagréger. Soit Ω l'ensemble des colonnes et ϖ un vecteur positif non nul, la colonne ω agrégée s'écrit : $\omega = \Omega\varpi$. De même, si Γ est le vecteur des composantes de l'objectif associées à chaque colonne, la composante de l'objectif agrégée s'écrit : $\gamma = \varpi^T\Gamma$. Bien que de nombreuses politiques quand à la manière d'agréger les colonnes peuvent être utilisées, la seule que nous ayons réellement testée consiste à prendre le vecteur ϖ égal au vecteur unité. Cela revient à donner une importance égale à chaque colonne. Une alternative raisonnable consisterait à prendre ϖ comme une pondération normée donnée en fonction des distances, en valeur absolue dans la métrique de l'ellipsoïde, du centre analytique à chaque coupe. Une autre alternative encore consisterait à prendre ϖ comme une pondération normée donnée en fonction de la valeur absolue de la variable d'écart associée à chacune des coupes au centre analytique. Ces alternatives n'ont malheureusement pas été testées.

Une fois les colonnes agrégées, il est possible, par les mêmes calculs que ceux développés au paragraphe § 6.3, de déterminer le niveau de cette colonne. Un point cependant diffère lors de la recherche unidimensionnelle. Comme par la suite, l'algorithme va chercher à déterminer un nouveau centre analytique avec les colonnes désagrégées, un choix judicieux pour α est de le prendre en fonction de la fonction potentielle à minimiser dans l'algorithme projectif. On va donc déterminer α de manière à minimiser la fonction suivante :

$$\bar{\varphi}(\bar{x}) = (\kappa + 1 + m)\ln[\tilde{c}^T\tilde{x} + \alpha(\tilde{c}^T\tilde{X}\tilde{d} - \gamma)] - m\ln\alpha - \sum_{j=0}^{\kappa}\ln\tilde{x}_j(1 + \alpha\tilde{d}_j),$$

avec m le nombre de colonnes à ajouter. Au paragraphe § 6.3 m était égal à 1.

Le paramètre α qui va minimiser cette fonction va être déterminé en appliquant la variante de l'algorithme de Newton que nous avons vu au paragraphe § 6.3. Une itération se calcule de la manière suivante :

$$\begin{aligned}
\alpha^{k+1} &:= \alpha^k - \frac{\bar{\phi}'(\alpha^k)}{\bar{\phi}''(\alpha^k)} \\
&:= \alpha^k - \frac{(\kappa+1+m)b/(a+\alpha b) - m/\alpha - \sum_{j=0}^{\kappa}(d_j/(1+\alpha d_j))}{-(\kappa+1+m)(b/(a+\alpha b))^2 + m/\alpha^2 + \sum_{j=0}^{\kappa}(d_j/(1+\alpha d_j))^2},
\end{aligned}$$

où $a = \tilde{c}^T \tilde{x}$ et $b = \tilde{c}^T \tilde{X} \tilde{d} - \gamma$.

7.2 Réduction du nombre de coupes

7.2.1 Elimination des colonnes

Description du test

La partie la plus coûteuse en temps, dans le calcul du centre analytique provient de la résolution du système linéaire : $(\tilde{G}\tilde{X}^2\tilde{G}^T)u = \tilde{G}\tilde{X}^2\tilde{c}$. Durant ce calcul, un temps important est consommé[3] pour former la matrice $(\tilde{G}\tilde{X}^2\tilde{G}^T)$. Il paraît donc légitime de chercher à limiter au maximum le nombre de colonnes de la matrice $\tilde{G}\tilde{X}$.

A chaque itération une ou plusieurs coupes sont ajoutées à la relaxation linéaire du problème original (4.1). Il est possible qu'après quelques itérations les informations apportées par ces coupes ne soient plus intéressantes. Comme une colonne de $\tilde{G}$ contient l'information apportée par une coupe, en supprimant une coupe on supprime une colonne.

Le test que nous allons développer dans cette section permet de détecter les coupes "inutiles" afin de les éliminer. Ce test est basé sur les ellipsoïdes inscrits et exinscrits (voir paragraphe § 5.7). Il consiste à estimer la position des coupes par rapport à l'ellipsoïde exinscrit. Rappelons que cet ellipsoïde contient l'ensemble de localisation. Donc, si un plan coupant contient strictement cet ellipsoïde dans son demi-espace, il ne peut pas être tangent à l'ensemble de localisation. Il ne participe alors pas à la définition de la solution optimale et peu de ce fait être éliminé.

L'objectif est donc de déterminer les colonnes de $\tilde{G}$ qui peuvent être éliminées ou bien, dans l'espace dual, les contraintes de $\tilde{G}^T$ qui peuvent être supprimées. Nous

[3] La complexité du produit d'une matrice $n \times \kappa$ par sa transposée est : $O(\kappa n^2)$.

avons vu que l'ellipsoïde dans l'espace dual centré au centre analytique approché est défini de la manière suivante :

$$\mathcal{E} = \{u : \|\tilde{X}\tilde{G}^T(u - u^c)\| \le \varepsilon\},$$

où ε fixe sa "taille" .

Dans un premier temps, déterminons la taille de l'ellipsoïde tangent à la contrainte $G_i^T u \ge c_i$. Pour cela, résolvons le problème :

$$\begin{array}{rl} \text{Maximiser} & \frac{1}{2}\|\tilde{X}\tilde{G}^T(u - u^c)\|^2 \\ \text{s.c.} & G_i^T u = c_i. \end{array}$$

Les conditions d'optimalité de ce problème sont :

$$\tilde{G}\tilde{X}^2\tilde{G}^T(u - u^c) - G_i\lambda = 0 \tag{7.7}$$

$$G_i^T u = c_i, \tag{7.8}$$

où λ est le multiplicateur de Lagrange. De (7.7) on obtient :

$$u = u^c + \lambda(\tilde{G}\tilde{X}^2\tilde{G}^T)^{-1}G_i.$$

En utilisant (7.8), on déduit le multiplicateur de Lagrange :

$$\lambda = \frac{c_i - G_i^T u^c}{G_i^T(\tilde{G}\tilde{X}^2\tilde{G}^T)^{-1}G_i}.$$

Le maximum de la fonction est donc :

$$\frac{1}{2}\|\tilde{X}\tilde{G}^T(u - u^c)\|^2 = \frac{1}{2}\lambda^2\|\tilde{X}\tilde{G}^T(\tilde{G}\tilde{X}^2\tilde{G}^T)^{-1}G_i\|^2.$$

Comme :

$$\begin{aligned} \|\tilde{X}\tilde{G}^T(\tilde{G}\tilde{X}^2\tilde{G}^T)^{-1}G_i\|^2 &= G_i^T(\tilde{G}\tilde{X}^2\tilde{G}^T)^{-T}\tilde{G}\tilde{X}^2\tilde{G}^T(\tilde{G}\tilde{X}^2\tilde{G}^T)^{-1}G_i \\ &= G_i^T(\tilde{G}\tilde{X}^2\tilde{G}^T)^{-1}G_i, \end{aligned}$$

la valeur du maximum est :

$$\frac{1}{2}\frac{(G_i^T u^c - c_i)^2}{G_i^T(\tilde{G}\tilde{X}^2\tilde{G}^T)^{-1}G_i}.$$

La contrainte $G_i^T u \geq c_i$ intersecte donc l'ellipsoïde extérieur $\mathcal{E}_{ext}$, défini par le théorème (5.9), si

$$\frac{(G_i^T u^c - c_i)^2}{G_i^T(\tilde{G}\tilde{X}^2\tilde{G}^T)^{-1}G_i} \leq \varepsilon_e^2,$$

avec $\varepsilon_e^2 = \frac{\kappa}{\kappa+1}(\tilde{c}^T\tilde{x})^2$.

On en déduit :

Théorème : 7.2 *Si*

$$\tilde{s}_i = \tilde{G}_i^T u^c - \tilde{c}_i > \varepsilon_e \sigma_i,$$

où $\sigma_i^2 = \tilde{G}_i^T(\tilde{G}\tilde{X}^2\tilde{G}^T)^{-1}\tilde{G}_i$, *alors il n'y a pas d'intersection entre l'ellipsoïde externe et la contrainte. La colonne associée à cette dernière peut donc être éliminée de* $\tilde{G}$.

Preuve. La preuve de l'intersection entre l'ellipsoïde et la contrainte découle de ce qui vient d'être dit. De plus, comme il n'y a pas d'intersection et comme cet ellipsoïde englobe l'ensemble de localisation $\mathcal{L}_{\overline{z}}$, la contrainte n'intervient donc en rien dans la définition de ce dernier. Elle peut donc être éliminée. ∎

Dans la pratique, ce test se révèle souvent insuffisant. Beaucoup d'autres colonnes de $\tilde{G}$ peuvent être éliminées sans pour autant supprimer de l'information significative. On a alors imaginé de réduire l'ellipsoïde de référence d'un facteur homothétique ρ. Le test devient :
Si $\tilde{s}_i > \varepsilon_e/\rho\sigma_i$ où $\sigma_i^2 = \tilde{G}_i^T(\tilde{G}\tilde{X}^2\tilde{G}^T)^{-1}\tilde{G}_i$ et $\rho > 1$ est une constante fixée, la colonne de $\tilde{G}$ associée à cette contrainte est éliminée.

Il n'existe pas de fondement théorique pour ce test. Il y a cependant une valeur $\rho_{max} = \frac{\sqrt{\kappa(1+\kappa)}}{1-\theta}$ au delà de laquelle cet ellipsoïde réduit est plus petit que l'ellipsoïde intérieur $\mathcal{E}_{int}$ et où toutes les colonnes sont alors éliminées. Comme il n'existe pas d'argument solide pour fixer ρ entre ces deux valeurs extrêmes, on utilise un critère accessoire basé sur le nombre de plans coupants restants. Pour déterminer une solution au problème, au moins $n+1$ plans de support seront nécessaires. Un test judicieux consiste donc à garantir la conservation d'au moins $n+1$ contraintes, c'est-à-dire $n+1$ colonnes de la matrice $\tilde{G}$. De plus, pour satisfaire la contrainte de convexité, il faut garantir l'existence d'au moins un plan de support de la fonction objectif.

Compte tenu de ces remarques, et après avoir testé de nombreuses stratégies, nous avons établi le test suivant : pour chaque contrainte, on calcule :

$$\lambda_i = \frac{\tilde{s}_i}{\varepsilon_e\sqrt{\tilde{G}_i^T(\tilde{G}\tilde{X}^2\tilde{G}^T)^{-1}\tilde{G}_i}}.$$

Nous avons alors :

$$\begin{cases} \lambda_i > 1 : & G_i^T u \geq c_i \quad \text{est extérieure à} \quad \mathcal{E}_{ext} \quad \text{et peut donc être éliminée,} \\ \lambda_i = 1 : & G_i^T u \geq c_i \quad \text{est tangente à} \quad \mathcal{E}_{ext} \quad \text{et peut donc être éliminée,} \\ \lambda_i < 1 : & G_i^T u \geq c_i \quad \text{coupe} \quad \mathcal{E}_{ext}. \end{cases}$$

Le scalaire λ_i représente la distance de la contrainte par rapport au centre analytique approché dans la métrique de l'ellipsoïde. Les contraintes qui coupent $\mathcal{E}_{ext}$ sont triées par ordre des λ_i décroissants de manière à éliminer en priorité les contraintes les plus éloignées.

De nombreuses heuristiques ont été envisagées pour ajuster les différents paramètres de ce test. La valeur de ρ, tout d'abord, a fait l'objet de beaucoup d'expérimentations. Si elle est trop petite, le test est inefficace. A contrario, si elle est trop élevée un nombre démesuré de plans sont éliminés et la méthode ACCPM peut ne pas converger. De même, le nombre de colonnes à conserver reste un problème ouvert. Fixer à n ce nombre est souvent trop petit. De plus, le fait que ce paramètre et ρ soient correlés rend difficile leur réglage. Enfin, ce test est relativement coûteux puisqu'il nécessite la résolution d'un systéme linéaire pour chaque contrainte. Il ne doit donc pas être effectué à chaque itération. La fréquence à laquelle ce test est appliqué est aussi un paramètre particulièrement sensible et nécessite souvent un affinage particulier pour chaque type de problème.

Pour finir, nous avons retenu les paramètres suivants :

- la valeur de $\lambda_i = \frac{1-\theta}{\sqrt{\kappa(1+\kappa)}}$ est tolérée pour envisager d'éliminer la contrainte, ce qui signifie que toute colonne est éliminable. La valeur de λ_i ne sert donc que pour le tri.
- le nombre minimal de colonnes conservées est $2n$.
- Soit $\hbar$ la fréquence à laquelle est effectuée le test. Initialement $\hbar$ est fixée à $3n$. Par la suite sa valeur est égale à la précédente additionnée de la différence entre n et le nombre de colonnes éliminées lors du précédent test. Le test s'effectuera donc d'autant plus tard que le précédent aura été infructueux.

De manière générale, ce test est dangereux et coûteux. Même s'il est possible d'utiliser pour le calcul des λ_i la factorisation de $\tilde{G}\tilde{X}^2\tilde{G}^T$ calculée lors de la dernière itération du module projectif, pour chaque contrainte le calcul de σ_i implique la résolution d'un système linéaire. Dans la pratique, nous avons observé que si ce test est particulièment intéressant pour certains types de problèmes, il n'est pas rare que pour d'autres il soit plus judicieux de le désactiver.

Restauration de la réalisabilité

Lorsque l'on supprime des colonnes de $\tilde{G}$, on se trouve face à un problème de réalisabilité semblable à celui rencontré lors de l'adjonction d'une colonne. Le test basé sur les ellipsoïdes doit donc être suivi d'une procédure permettant de retrouver un point $\bar{x}$ tel que $\bar{G}\bar{x} = 0$ où $\bar{G}$ est la matrice $\tilde{G}$ réduite.

Soit S l'ensemble des indices des colonnes à supprimer. Posons $x_\omega = \sum_{i \in S} \tilde{x}_i$ et $\bar{x} = \{\tilde{x}_i : i \notin S\}$. De même posons $\tilde{\omega}$ la somme des colonnes de $\tilde{G}$ à supprimer et $\bar{G}$ les colonnes restantes. L'objectif est de résoudre le problème suivant :

$$\left\{\begin{array}{rcl} \bar{G}\bar{x} & = & 0 \\ \bar{g}^T\bar{x} & = & 1 \\ \bar{x} & \geq & 0, \end{array}\right.$$

avec $\bar{g} = (1, 0, \ldots, 0)^T$. Nous sommes exactement dans le même cadre d'étude que lors de l'ajout de colonnes. Nous avons vu au paragraphe § 6.3 que ce problème se résoud par une méthode de Newton dont la direction est $\bar{d} = \bar{X}\bar{q}$ où :

$$\bar{q} = \mathbf{1} - \bar{X}\bar{G}^T(\bar{G}\bar{X}^2\bar{G}^T)^{-1}\bar{G}\bar{X}\mathbf{1}.$$

L'algorithme de réalisabilité consiste donc à itérer suivant cette direction de la façon suivante :

$$\bar{x}^{k+1} = \frac{\bar{X}^k(\mathbf{1} + \bar{\alpha}\bar{q})}{1 + \bar{\alpha}\bar{q}_0},$$

jusqu'à ce que $\|\bar{G}\bar{x}\| < \varepsilon$. Comme $\bar{x}^k > 0$ à chaque itération et comme $\bar{q}$ est dans le noyau de $\bar{G}\bar{X}$ on choisit plutôt le critère d'arrêt $\bar{q} > 0$ qui ne nécessite aucun calcul. Dans la pratique, une seule itération est souvent suffisante.

Tout comme dans le cas de l'adjonction d'une colonne, on peut utiliser la factorisation de $\tilde{G}\tilde{X}^2\tilde{G}^T$ pour calculer $\bar{q}$. $\tilde{G}$ et $\tilde{x}$ s'écrivent :

$$\tilde{G} = (\bar{G} \quad \tilde{\omega}), \quad \tilde{x} = \begin{pmatrix} \bar{x} \\ x_\omega \end{pmatrix}.$$

Si on développe $\bar{q}$ en fonction de $\tilde{G}$, $\tilde{x}$, $\tilde{\omega}$ et x_ω on obtient tout d'abord :

$$\bar{G}\bar{X}\mathbf{1} = \tilde{G}\tilde{X}\mathbf{1} - x_\omega\tilde{\omega} = -x_\omega\tilde{\omega},$$

et par la formule de Sherman-Morrison-Woodbury :

$$\begin{aligned}(\bar{G}\bar{X}^2\bar{G}^T)^{-1} &= (\tilde{G}\tilde{X}^2\tilde{G}^T - x_\omega^2\tilde{\omega}\tilde{\omega}^T)^{-1}\\ &= (\tilde{G}\tilde{X}^2\tilde{G}^T)^{-1} + \frac{x_\omega^2}{1-x_\omega^2\tau^2}(\tilde{G}\tilde{X}^2\tilde{G}^T)^{-1}\tilde{\omega}\tilde{\omega}^T(\tilde{G}\tilde{X}^2\tilde{G}^T)^{-1},\end{aligned}$$

avec $\tau^2 = \tilde{\omega}^T(\tilde{G}\tilde{X}^2\tilde{G}^T)^{-1}\tilde{\omega}$.

Il découle :

$$\begin{aligned}(\bar{G}\bar{X}^2\bar{G}^T)^{-1}\bar{G}\bar{X}\mathbf{1} &= (\tilde{G}\tilde{X}^2\tilde{G}^T)^{-1}\tilde{\omega}(-x_\omega - \frac{x_\omega^3\tau^2}{1-x_\omega^2\tau^2})\\ &= \frac{-x_\omega}{1-x_\omega^2\tau^2}(\tilde{G}\tilde{X}^2\tilde{G}^T)^{-1}\tilde{\omega},\end{aligned}$$

dont on déduit :

$$\bar{q} = \mathbf{1} + \frac{x_\omega}{1-x_\omega^2\tau^2}\bar{X}\bar{G}^T(\tilde{G}\tilde{X}^2\tilde{G}^T)^{-1}\tilde{\omega}.$$

Dans l'implantation d'ACCPM, la factorisation de $\tilde{G}\tilde{X}^2\tilde{G}^T$ n'a pas été utilisée pour calculer $\bar{q}$. En effet, comme nous avons choisi de placer la procédure de restauration de la réalisabilité après la procédure d'adjonction de la nouvelle colonne, la factorisation de $\tilde{G}\tilde{X}^2\tilde{G}^T$ n'est donc plus disponible. Cependant, si nous avions opté pour l'ordre inverse, la factorisation aurait été disponible et cela aurait rendu plus attrayant la suppression de colonnes par la méthode des ellipsoïdes.

Enfin, si l'on fait l'hypothèse que les colonnes à éliminer ont été correctement sélectionnées, leur niveau devrait être proche de zéro (c.à.d. $x_w \simeq 0$) puisqu'il correspond à la valeur marginale de la contrainte. La dernière écriture de $\bar{q}$ montre alors qu'il n'est guère surprenant que dès la première itération, $\bar{q}$ soit presque toujours positif.

Remarques et conclusions

L'élimination de contraintes s'est souvent avérée inefficace, le surcoût engendré étant souvent plus important que le gain[4]. De plus, ce test élimine parfois trop de colonnes

[4] Nous verrons au dernier chapitre une estimation de la complexité de ce test

et induit alors un plus grand nombre d'itérations dans ACCPM pour déterminer l'optimum.

D'autres tests que celui basé sur les ellipsoïdes ont été essayés, certains basés sur la valeur des composantes de $\tilde{x}$, d'autres sur $\tilde{s}$. Aucun n'a donné de résultats concluants.

Le plus précis des tests que nous ayons imaginé semble être celui basé sur les ellipsoïdes. C'est aussi le plus coûteux en temps de calcul. Il est donc indispensable de ne pas l'exécuter à chaque itération. Malheureusement, plus le nombre de colonnes de $\tilde{G}$ est important, moins ce test devient précis (le rapport entre l'ellipsoïde interne et externe augmente) et plus les calculs prennent du temps. Il nous faut alors trouver un compromis. Si celui-ci est relativement facile à établir lorsque seule une colonne est incorporée à chaque itération, nous n'y sommes pas parvenus lorsqu'un ensemble de colonnes est introduit.

Une dernière alternative consiste à trier les variables d'écart $\tilde{s}$ par ordre décroissant. On divise alors cet ensemble trié en quatre parties égales. On élimine ensuite directement les colonnes de $\tilde{G}$ correspondant au premier quart et on effectue le test basé sur les ellipsoïdes sur les deux quarts suivants. Cette alternative permet de concentrer l'effort de calcul sur seulement la moitiée des coupes (celles qui posent un litige) et donc de diviser par deux le coût en temps de calcul. Il n'a malheureusement pas été implanté dans notre version.

Nous venons de voir comment restaurer la réalisabilité après la suppression des colonnes. Une autre solution consiste à agréger ces colonnes plutôt que de les supprimer. Conceptuellement le problème reste identique :

$$\left.\begin{array}{rcl} \sum_{i=1}^{\kappa} G_{ij_1}^T u_i & \geq & c_{j_1} \\ \sum_{i=1}^{\kappa} G_{ij_2}^T u_i & \geq & c_{j_2} \end{array}\right\} \Longleftrightarrow \sum_{i=1}^{\kappa}(G_{ij_1}^T + G_{ij_2}^T)u_i \quad \geq \quad c_{j_1} + c_{j_2}.$$

Il reste alors une colonne de plus que si nous les avions réellement supprimées mais il n'est pas nécessaire de restaurer la réalisabilité puisque le nouveau vecteur $\tilde{x}$ est toujours primal réalisable. Cette idée n'a pas été testée.

7.2.2 Filtre sur les coupes à introduire

Un autre test, beaucoup plus simple et relativement efficace, consiste à ne pas introduire une colonne qui est déjà présente dans la matrice $\tilde{G}$. Il est possible en effet qu'en un autre point, le même plan de support ait été introduit.

Tout au long de l'algorithme des coupes sont introduites afin d'enrichir l'approximation linéaire du problème original. Etudions les coupes d'optimalité. Rappelons tout d'abord que ce sont des plans de support de la fonction objectif f et qu'elles ont été mémorisées dans les colonnes de la matrice G. Pour connaître la valeur d'un plan $i \in I$ au point courant u^c, il suffit de calculer $\gamma_i - \Omega_i^T y^c$. En effet :

$$\begin{aligned} \gamma_i - \Omega_i^T y^c &= f(y^i) - \xi^T f(y^i) y^i + \xi^T f(y^i) y^c \\ &= f(y^i) + \xi^T f(y^i)(y^c - y^i). \end{aligned}$$

On remarque que pour tout $i \in I$, $\gamma_i - \Omega_i^T y^c = c_i - \tilde{G}_i^T u^c + z = z - \tilde{s}_i$. Ces valeurs sont donc directement disponibles puisque $\tilde{s}$ a été calculé lors d'une des étapes du calcul de $\tilde{q}$ dans le module projectif (voir (5.12)).

On peut donc effectuer le test suivant :

Si $z - \tilde{s}_i = f(y^c)$ avec $i \in I$, alors il s'agit du même plan de support[5] et il est inutile de l'incorporer dans la matrice.

Ce test peut être relâché en posant :

Si $|f(y^c) + \tilde{s}_i - z| < \varepsilon$ avec $i \in I$ et où ε est une petite valeur quelconque ($\varepsilon < 10^{-6}$), alors il s'agit du même plan de support et il est inutile de l'incorporer dans la matrice.

Ce type de test permet de filtrer les coupes d'optimalité envoyées par l'oracle avant que toute opération soit faite par le programme-maître, c'est-à-dire avant que les coupes soit intégrées à la matrice $\tilde{G}$. Il peut se généraliser aux coupes de réalisabilité si celles-ci sont des plans de support de l'ensemble Y.

Non seulement ce test est très peu coûteux (il ne nécessite que des comparaisons) mais il est efficace au moment adéquat, c'est-à-dire en fin d'exécution lorsque beaucoup d'informations sont présentes (et donc lorsque la nombre de colonnes dans la matrice $\tilde{G}$ est grand) et que seule la valeur de la borne supérieure $\overline{z}$ évolue[6]. De plus, ce test est particulièrement efficace lorsque plusieurs plans de support sont intégrés à chaque itération et que la fonction est linéaire par morceaux (problèmes stochastiques, problèmes de multiflots).

[5] La non différentiabilité ne permet pas de garantir qu'il s'agisse du même plan. Cependant, dans la pratique ce test semble fonctionner convenablement même pour les problèmes non différentiables.

[6] Par opposition avec le test basé sur les ellipsoïdes qui devient imprécis lorsque le nombre de colonnes croît.

7.3 Gestion dynamique des boîtes

La gestion dynamique des boîtes a deux intérêts. Tout d'abord elle permet de les prendre initialement très grandes ou très petites, sans affecter le comportement de l'algorithme. On s'assure ainsi de trouver l'optimum sans trop augmenter le nombre d'itérations. Le second intérêt, moins immédiat mais tout aussi important, assure une certaine stabilité numérique en ne gardant dans les calculs que des nombres d'un ordre de grandeur comparable.

7.3.1 Gestion par les ellipsoïdes

Cette technique consiste à ajuster les boîtes de manière à ce qu'elles soient tangentes à l'ellipsoïde extérieur. Pour cela il suffit que :

$$\lambda_i = \frac{\tilde{s}_i}{\varepsilon_e \sqrt{(\tilde{G}\tilde{X}^2\tilde{G}^T)^{-1}_{ii}}} = 1, \qquad \forall i = 1, 2, \ldots, 2n,$$

où le scalaire ε_e a été défini par le théorème 5.9.

Si $i \leq n$, comme $\tilde{s}_i = \tilde{G}_i^T u + \tilde{c}_i = u_i + \tilde{c}_i$, la réduction est la suivante :

$$\tilde{c}_i = \varepsilon_e \sqrt{(\tilde{G}\tilde{X}^2\tilde{G}^T)^{-1}_{ii}} - u_i, \qquad \forall i = 1, \ldots, n.$$

De même, si $n < i \leq 2n$, comme $\tilde{s}_i = \tilde{G}_i^T u + \tilde{c}_i = -u_{i-n} + \tilde{c}_i$ la réduction est la suivante :

$$\tilde{c}_i = \varepsilon_e \sqrt{(\tilde{G}\tilde{X}^2\tilde{G}^T)^{-1}_{(i-n)(i-n)}} + u_{i-n}, \qquad \forall i = n+1, \ldots, 2n.$$

Comme le test d'élimination de colonnes, cette réduction nécessite la résolution d'un système linéaire pour chaque composante des boîtes. Elle est donc coûteuse.

7.3.2 Gestion heuristique

Une autre solution consiste à comparer le vecteur courant u^c à ces boîtes. Dès que celui-ci s'éloigne ou se rapproche “trop” d'une boîte on agit sur cette dernière. On procède ainsi :

pour $i = 1, \ldots, n$, **faire**
  Si $|(u_i - \tilde{c}_i)/u_i| > \rho_1$, **alors** $\tilde{c}_i := \tilde{c}_i/\rho_2$ **fsi** ; % Rapproche la boîte inf. %
  Si $|(u_i - \tilde{c}_{i+n})/u_i| > \rho_1$, **alors** $\tilde{c}_{i+n} := \tilde{c}_{i+n}/\rho_2$ **fsi** ; % Rapproche la boîte sup. %
  Si $|(u_i - \tilde{c}_i)/u_i| > \rho_3$, **alors** $\tilde{c}_i := \rho_4 \tilde{c}_i$ **fsi** ; % Eloigne la boîte inf. %
  Si $|(u_i - \tilde{c}_{i+n})/u_i| < \rho_3$, **alors** $\tilde{c}_{i+n} := \rho_4 \tilde{c}_{i+n}$ **fsi** ; % Eloigne la boîte sup. %
Fait.

Ce test est heuristique. Il permet cependant d'être complètement insensible aux valeurs l et h de départ. Ainsi, même si on choisit de trop petites boîtes qui n'englobent pas l'optimum, celles-ci seront automatiquement augmentées. Inversement, si les boîtes sont trop grandes, elles seront réajustées.

Nous avons expérimenté les valeurs suivantes : $\rho_1 = 10^4$, $\rho_2 = 10^3$, $\rho_3 = 0.9$, $\rho_4 = 10^2$.

7.4 Phase initiale de l'algorithme

L'algorithme ACCPM débute par un appel à l'oracle. Pour cela, il nous faut un point y appartenant à l'ensemble de localisation. Initialement, cet ensemble n'est défini que par les boîtes. Si on ne connaît pas d'approximation de la solution, on prend comme point y de départ le centre de ces boîtes. Soit $y^c = \frac{h+l}{2}$.

En retour l'oracle renvoie une ou plusieurs formes linéaires. En fonction du type de la réponse, la phase initiale va être différente.

7.4.1 Sans phase I

Si dans l'ensemble de formes linéaires envoyées par l'oracle, il y a au moins un plan de support de $f(y)$, il est possible de passer directement à la phase II. La valeur de $\overline{z}$ est mise à jour avec $\overline{z} = f(y^c)$. La ou les colonnes (plans de support de l'objectif renvoyés par l'oracle) sont ajoutées à $\tilde{G}$.

Pour pouvoir déterminer le nouveau centre analytique approché il nous faut connaître un point $\tilde{x}$ réalisable, c'est-à-dire un point $\tilde{x}$ tel que $\tilde{G}\tilde{x} = 0$ et $\tilde{x} > 0$. Malheureusement, il n'est pas possible d'utiliser la méthode décrite en fin de section § 6.3 pour déterminer ce point. En effet, avant l'ajout des colonnes, la matrice $\tilde{G}$ à la forme suivante :

$$\begin{pmatrix} 0 & I_d & -I_d \\ -\mathbf{1} & 0 & 0 \end{pmatrix},$$

où le vecteur **1** est réduit au scalaire 1 si la fonction objectif est unique. Par contre, si l'objectif est constitué de la somme de fonctions convexes, la dimension de **1** est égale au nombre de fonctions dont l'objectif est composé.

De par la présence de ce vecteur unité, il n'est pas possible de déterminer un vecteur $\tilde{x}$ précédemment admissible[7], c'est-à-dire un vecteur $\tilde{x} > 0$ tel que $\tilde{G}\tilde{x} = 0$ avant l'ajout des colonnes. Une solution consiste à introduire la, ou les colonnes dans la matrice $\tilde{G}$ et à appliquer l'algorithme de réalisabilité décrit à la section § 6.3.

Dans l'implantation de l'algorithme, nous avons procédé différemment. Après avoir introduit la, ou les colonnes dans $\tilde{G}$ nous avons mis les composantes correspondantes du vecteur $\tilde{x}$ à 1 et avons cherché le vecteur $\tilde{x}$ dans le noyau de $\tilde{G}$ en utilisant la procédure zéro-machine décrite au paragraphe § 6.6. Si cette manière de procéder est en général peu recommandable[8], elle ne pénalise pas trop l'algorithme ACCPM. En effet, si nous avions utilisé l'algorithme de réalisabilité, le gain (en nombre d'itérations) fait dans l'algorithme projectif aurait été consommé par les itérations faites dans l'algorithme de réalisabilité. Comme, de plus, la procédure zéro-machine est beaucoup moins coûteuse en temps que l'algorithme de réalisabilité ce choix semble raisonnable.

7.4.2 Avec phase I

Si, dans l'ensemble de colonnes envoyées par l'oracle, il n'y a pas de plan de support de $f(y)$, il n'est pas possible de passer à la phase II. Un passage par la phase I s'impose.

Nous allons, dans ce qui va suivre, nous placer dans le contexte où l'on cherche, durant la phase I, à minimiser une fonction indicatrice convexe. Cette manière de procéder étant celle que nous avons choisi dans l'implantation.

Dans ce cas, nous connaissons une borne inférieure : 0. Il est donc possible d'intégrer un plan de support horizontal au niveau 0. La matrice initial $\tilde{G}$ a alors la forme

[7]Nous verrons en quoi cela aurait pu nous être utile dans la section 7.4.2 "Avec phase I".

[8]Nous avons vu l'importance d'introduire correctement les colonnes dans $\tilde{G}$ pour diminuer le nombre d'itérations dans l'algorithme projectif.

suivante :

$$\begin{pmatrix} 0 & I_d & -I_d & 0 \\ -\mathbf{1} & 0 & 0 & I_d \end{pmatrix},$$

où la troisième matrice identité I_d est réduite au scalaire 1 si la fonction indicatrice $h(y)$ est unique. Ceci étant un cas particulier, nous allons considérer par la suite que $h(y)$ est composé de la somme de m fonctions convexes[9]. Les calculs qui vont maintenant être développés ont pour objectif de montrer comment il est possible d'utiliser cette borne inférieure pour commencer la phase I de façon efficace.

De (7.4) on déduit l'approximation linéaire de ce problème durant la phase I :

$$\begin{aligned} \text{Minimiser} \quad & z_1 + z_2 + \cdots + z_m \\ \text{s.c.} \quad & \Omega_{J_1}^T y + z_1 \geq \gamma_{J_1} \\ & \vdots \\ & \Omega_{J_m}^T y + z_m \geq \gamma_{J_m} \\ & l \leq y \leq h. \end{aligned}$$

Lorsque seuls les plans horizontaux sont présents, ce problème s'écrit :

$$\begin{aligned} \text{Minimiser} \quad & z_1 + z_2 + \cdots + z_m \\ \text{s.c.} \quad & z_1 \mathbf{1} \geq \mathbf{0} \\ & \vdots \\ & z_m \mathbf{1} \geq \mathbf{0} \\ & l \leq y \leq h. \end{aligned}$$

On peut de même définir le dual de ces problèmes. Le dual du premier est :

$$\begin{aligned} \text{Maximiser} \quad & l^T \zeta - h^T \eta + \gamma_{J_1}^T \varrho_1 + \cdots + + \gamma_{J_m}^T \varrho_m \\ \text{s.c.} \quad & \zeta - \eta + \Omega_{J_1} \varrho_1 + \cdots + \Omega_{J_m} \varrho_m = 0 \\ & \mathbf{1}^T \varrho_1 = 1 \\ & \vdots \\ & \mathbf{1}^T \varrho_m = 1 \\ & \zeta, \eta, \varrho \geq 0. \end{aligned}$$

[9]Ce cas peut survenir par exemple si l'ensemble Y est composé de l'intersection de m ensembles convexes et que l'on prend une fonction indicatrice par ensemble.

et du second :

$$\begin{aligned} \text{Maximiser} \quad & l^T\zeta - h^T\eta \\ \text{s.c.} \quad & \zeta - \eta = 0 \\ & \mathbf{1}^T\varrho_1 = 1 \\ & \vdots \\ & \mathbf{1}^T\varrho_m = 1 \\ & \zeta, \eta, \varrho \geq 0. \end{aligned}$$

Nous avons envoyé à l'oracle le point $y^c = \frac{h+l}{2}$. En retour, nous avons reçu une borne supérieure $\overline{z} = h(y^c)$ et un ensemble de plans de support. Il est possible de faire l'hypothèse que nous connaissions déjà cette borne supérieure et que l'ensemble de localisation s'écrivait alors :

$$\mathcal{L}_{\overline{z}} = \{(y, z) \in \mathbb{R}^{n+m} : \sum_{t=1}^{m} z_t \leq \overline{z},\ z_t \geq 0,\ l \leq y \leq h\}.$$

Comme nous n'avons pas d'informations sur le problème, faisons l'hypothèse que toutes les composantes de z étaient égales. Le vecteur (y, z) qui appartenait à cet ensemble de localisation devait avoir la forme :

$$\begin{pmatrix} y \\ z_1 \\ \vdots \\ z_m \end{pmatrix} = \begin{pmatrix} \frac{h+l}{2} \\ \alpha\overline{z} \\ \vdots \\ \alpha\overline{z} \end{pmatrix}, \qquad \alpha \in]0, \frac{1}{m}].$$

Si on veut pouvoir utiliser la méthode développée au paragraphe § 6.3 pour ajouter les colonnes, il faut connaître le vecteur $\tilde{x}$ associé à ce vecteur (y, z). Plus exactement, il faut connaître le vecteur $\tilde{x}$ qui aurait été obtenu par le calcul du centre analytique tel que le vecteur u associé soit ce vecteur (y, z). Nous avons vu que les conditions d'optimalité pour le calcul du centre analytique sont les suivantes :

$$\begin{aligned} Gx &= b, & x &> 0 \\ G^T u - s &= c, & s &> 0 \\ b^T u + s_0 &= \overline{z}, & s_0 &> 0 \\ Xs &= s_0\mathbf{1}. \end{aligned}$$

De par la manière dont le vecteur u (c.à.d. (y, z)) a été choisi, la deuxième et la troisième condition d'optimalité sont satisfaites. De plus, la première condition d'optimalité et la stucture de la matrice $\tilde{G}$ impliquent que $\zeta = \eta$ et $\varrho_t = 1\ \forall t = 1, \cdots, m$. Enfin, les variables d'écart sont les suivantes :

$$\begin{array}{rcll} s_0 & = & \overline{z} - \sum_{t=1}^{m} z_t, & t = 1, \ldots, m \\ s_t & = & z_t, & t = 1, \ldots, m \\ s_{m+1} & = & y - l & \\ s_{m+n+1} & = & h - y. & \end{array}$$

De la quatrième condition d'optimalité, on déduit le système suivant :

$$\begin{array}{rcll} \varrho_t z_t & = & \overline{z} - \sum_{t=1}^{m} z_t, & t = 1, \ldots, m \\ \zeta(y - l) & = & \overline{z} - \sum_{t=1}^{m} z_t & \\ \eta(h - y) & = & \overline{z} - \sum_{t=1}^{m} z_t. & \end{array}$$

Comme $\eta = \zeta$ et $\sum_{t=1}^{m} z_t = m\alpha\overline{z}$ on obtient :

$$\begin{pmatrix} \zeta \\ \eta \\ \varrho_1 \\ \vdots \\ \varrho_m \end{pmatrix} = \begin{pmatrix} \frac{2\overline{z}(1-m\alpha)}{h-l} \\ \frac{2\overline{z}(1-m\alpha)}{h-l} \\ \frac{1-m\alpha}{\alpha} \\ \vdots \\ \frac{1-m\alpha}{\alpha} \end{pmatrix}, \qquad \alpha \in]0, \frac{1}{m}].$$

La condition $\varrho_t = 1$ implique que $\alpha = 1/(1 + m)$. On vérifie que α satisfait bien : $0 < \alpha \leq m$. On trouve alors le vecteur $\tilde{x}$:

$$\tilde{x} = \begin{pmatrix} x_0 \\ \zeta \\ \eta \\ \varrho_1 \\ \vdots \\ \varrho_m \end{pmatrix} = \begin{pmatrix} 1 \\ \frac{2\overline{z}}{(1+m)(h-l)} \\ \frac{2\overline{z}}{(1+m)(h-l)} \\ 1 \\ \vdots \\ 1 \end{pmatrix}.$$

Ayant trouvé un vecteur $\tilde{x}$ réalisable, il est possible de calculer la factorisation de $\tilde{G}\tilde{X}^2\tilde{G}^T$ et d'utiliser la méthode développée en fin de section § 6.3 pour ajouter les colonnes. Il est intéressant de remarquer que la factorisation est immédiate. En posant $\beta = \frac{2\overline{z}}{(1+m)(h-l)}$, la matrice à factoriser est la suivante :

$$\tilde{G}\tilde{X}^2\tilde{G}^T = \begin{pmatrix} 2\beta^2 I_d & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{pmatrix} + bb^T.$$

La factorisation de Cholesky de cette matrice avant la correction de rang 1 est la matrice symétrique suivante :

$$M = \begin{pmatrix} \sqrt{2}\beta I_d & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{pmatrix}.$$

En appliquant la formule de Shermann-Morisson-Woodbury, comme nous avons vu précédemment, on obtient l'inverse recherché :

$$(\tilde{G}\tilde{X}^2\tilde{G}^T)^{-1} = M^{-T}M^{-1} - \frac{M^{-T}M^{-1}bb^TM^{-T}M^{-1}}{1 + b^TM^{-T}M^{-1}b} \tag{7.9}$$

$$= M^{-T}\left(I_d - \frac{\tau\tau^T}{1+\tau^T\tau}\right)M^{-1}, \tag{7.10}$$

avec $\tau = M^{-1}b$.

Les tests ont montré que procéder de cette manière lors de la phase d'initialisation permettait de limiter de façon importante le nombre d'itérations nécessaires pour calculer le premier centre analytique. Cependant, le calcul de celui-ci nécessite presque toujours beaucoup plus d'itérations que les suivants.

Enfin, il est important de se souvenir que l'on a introduit ces plans horizontaux lors de l'initialisation de la phase I car ils devront être supprimés avant de commencer la phase II.

7.5 L'algorithme ACCPM complet

Nous avons vu maintenant, tous les ingrédients de l'algorithme ACCPM. Il est donc possible de le décrire plus précisément que nous l'avons fait précédemment. Dans ce qui va suivre, nous considérerons que l'oracle renvoie des coupes sans préciser si elles sont uniques ou non et s'il s'agit de coupes d'optimalité ou de réalisabilité. De plus, l'oracle renvoie la valeur de la fonction au point où il est appelé. Si les contraintes ne sont pas satisfaites, nous ferons l'hypothèse que cette valeur est infinie.

Algorithme ACCPM complet

Paramètres :
  Définir un niveau de tolérance pour l'objectif : $\varepsilon \geq 0$;
  Définir des bornes sur y : l et $h \in \mathbb{R}^n$;
Initialisation :
  $k := 0$;
  Borne inférieure de la valeur de la solution : $\underline{z} := -\infty$;
  Centre analytique initial :$y = ((h+l)/2, \cdots, (h+l)/2)$;
  Appel à l'oracle : Input : y ; Output : coupe(s), $\theta_1 := f(y)$;
  Borne supérieure de la valeur de la solution : $\overline{z} := \theta_1$;

Phase I

Si Pas de plan de support de l'objectif, **alors**
  Initialisation de la Phase I : introduire des plans horizontaux et ajouter
    des colonnes renvoyées par l'oracle. $\underline{z} := 0$;
  Déterminer le nouveau centre analytique approché : u ;
  Calculer une borne inférieure θ_2 ; $\underline{z} := \max\{\underline{z}, \theta_2\}$;
  Si $\underline{z} > 0$, **alors** STOP : problème sans solution **fsi** ;
  Appel à l'oracle : Input : y ; Output : coupe(s), $\theta_1 := f(y)$;
  Tant que Pas de plan de support de l'objectif, **faire**
    Mettre à jour l'ensemble de localisation : appliquer le tests sur les
      boîtes, ajouter les colonnes ;
    Déterminer le nouveau centre analytique approché : u ;
    Calculer une borne inférieure θ_2 ; $\underline{z} := \max\{\underline{z}, \theta_2\}$;
    Si $\underline{z} > 0$, **alors** STOP : problème sans solution **fsi** ;
    Appel à l'oracle : Input : y. Output : coupe(s), $\theta_1 := f(y)$;
    $k := k + 1$;
  Fait ;
  Supprimer les plans horizontaux ; $\underline{z} := -\infty$; $\overline{z} := \theta_1$;
Fsi ;

Phase II

Initialisation de la Phase II : ajouter les colonnes et appeler la procédure zero-machine ;

Déterminer le nouveau centre analytique approché : u ;

Calculer une borne inférieure θ ; $\underline{z} := \max\{\underline{z}, \theta\}$;

Tant que $\overline{z} - \underline{z} > \varepsilon$, **faire**

Appel à l'oracle : Input : y ; Output : coupe(s), $\theta_1 := f(y)$;

$\overline{z} := \min\{\overline{z}, \theta_1\}$;

Mettre à jour l'ensemble de localisation : appliquer le tests pour supprimer les colonnes, appliquer le tests sur les boîtes, et ajouter les colonnes ;

Déterminer le nouveau centre analytique : u ;

Calculer une borne inférieure θ_2 ; $\underline{z} := \max\{\underline{z}, \theta_2\}$;

$k := k + 1$;

Fait ;

Bibliographie

[1] O. Bahn, O. du Merle, J.-L. Goffin et J.-P. Vial, “A cutting plane method from analytic centers for stochastic programming”, à paraître dans *Mathematical programming.*

Chapitre 8

Algorithmes

Dans les chapitres précédents nous avons vu en détail comment mettre en œuvre la méthode ACCPM. Afin de synthéthiser les différentes informations nous allons maintenant donner les principaux algorithmes nécessaires à la programmation de ACCPM.

Nous verrons tout d'abord dans une partie introductive le contexte dans lequel nous nous plaçons ainsi que le style des algorithmes que nous allons adopter. Nous ferons ensuite une rapide discussion sur la structure de données que nous avons utilisée et nous terminerons par la description des différentes procédures qui forment l'algorithme ACCPM.

8.1 Introduction

Dans les sections qui vont suivre, nous allons nous placer dans le cadre général où la fonction $f(x)$ peut être considérée comme la somme de m fonctions convexes. De même, nous considérerons que, durant la phase I, on minimise une fontion convexe indicatrice qui peut, elle aussi, être considérée comme la somme de m fonctions convexes. Nous ferons, de plus, l'hypothèse que les coupes envoyées par l'oracle durant la phase I peuvent être considérées comme des coupes de réalisabilité par la phase II. Cette hypothèse se justifie par le fait que, dans la pratique, nous avons toujours trouvé une fonction indicatrice satisfaisant ces exigences. Par exemple, si $Y = \{y : g(y) \leq 0\}$ avec $g(y)$ convexe, on peut prendre $g(y)$ comme fonction indicatrice et les coupes envoyées durant la phase I pourrons être considérées comme des coupes de réalisabilité par la phase II. L'information obtenue lors de la phase I n'est pas perdue puisqu'elle peut être utilisé durant la phase II (ce qui ne serait pas le cas si nous ne faisions pas cette hypothèse).

Les procédures vont être décrites suivant le “style” de programmation en Matlab [1]. Les mots clefs tel que **Tantque-faire**, **Si-alors-sinon**, ... seront notés en gras ainsi que le nom des procédures et des fonctions. Après le nom des procédures ou des fonctions, nous trouverons entre parentèses les différents paramètres en entrée. Les paramètres en sortie seront donnés avant, entre crochets. Prenons par exemple une version simplifiée de la procédure **Oracle**. Pour un centre analytique u^c donné, elle renvoie les différents constituants d'une coupe (ω et γ) et une possible borne supérieure ($\overline{z}$). Suivant le style que nous avons adopté un appel à cette procédure s'écrit :

$$[\omega, \gamma, \overline{z}] = \textbf{ Oracle } (u^c).$$

8.2 La structure de données

Toute structure peut être envisagée depuis une version élémentaire où tout est dense (matrice et vecteur) jusqu'a une version sophistiquée qui utilise les moindres creux des matrices et des vecteurs. Il est clair que plus cette structure est sophistiquée plus elle sera délicate à développer mais plus le programme à des chances d'être performant.

Dans la version avec laquelle nous avons élaboré les expérimentations numériques nous n'avons pas exploité la structure des colonnes envoyées par l'oracle. Seule la matrice $\tilde{G}$ à fait l'objet d'une attention particulière. Nous avons vu que, d'un point de vue théorique, cette matrice a la forme suivante :

$$\tilde{G} = \begin{pmatrix} 0 & I_d & -I_d & \Omega_J & \Omega_{I_1} & \Omega_{I_2} & \cdots & \Omega_{I_m} \\ -1 & 0 & 0 & 0 & \mathbf{1}_1^T & 0 & \cdots & 0 \\ -1 & 0 & 0 & 0 & 0 & \mathbf{1}_2^T & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ -1 & 0 & 0 & 0 & 0 & 0 & \cdots & \mathbf{1}_m^T \end{pmatrix}.$$

On remarquera le creux présent dans cette matrice. Creux d'autant plus important que m est grand. Nous avons donc décidé de ne stocker en mémoire que les matrices Ω et de traiter de manière implicite les autres éléments de $\tilde{G}$. Cette structure permet de diminuer non seulement l'espace mémoire utile mais aussi le temps nécessaire pour tout produit faisant intervenir $\tilde{G}$. Comme, avec la factorisation, c'est l'opération la plus coûteuse (en espace mémoire et en temps de calcul), ceci est d'un intérêt non négligeable. En contrepartie, cette structure va nécessiter l'élaboration de procédures spéciales gérant les produits : $\tilde{G}a$, $\tilde{G}^T b$ et $\tilde{G}\tilde{X}^2\tilde{G}^T$. Nous ne rentrerons pas dans le détail de ces procédures.

Dans l'implantation de l'algorithme nous avons choisi, de plus, de mémoriser les colonnes envoyées par l'oracle les unes à la suite des autres dans une seule matrice (appelée Gr). Bien que cette politique soit économique en place mémoire[1], elle va nécessiter un réordonnancement de ces colonnes à chaque fois que l'on désirera en ajouter ou en supprimer. En effet, la structure de $\tilde{G}$[2] doit rester identique tout au long de la résolution du problème.

On aurait pu aussi envisager de stocker les colonnes dans différentes matrices Ω en fonction de leur type. Il aurait alors été nécessaire de réserver une plus grande place mémoire, ce qui nous a semblé prohibitif. Cependant, dans le cas où aucun test d'élimination n'est activé, toutes les matrices Ω_I ont le même nombre de colonnes et l'argument sur l'espace mémoire ne tient plus. Cette structure serait alors plus appropriée puisqu'elle ne nécessite pas le réordonnacement de $\tilde{G}$.

Désirant rester dans un cadre général nous avons donc choisi de mémoriser à la suite les colonnes envoyées par l'oracle dans la matrice Gr. Pour toutes les entités, autre que $\tilde{G}$, aucune structure particulière n'a été définie.

8.3 Programme principal

Le programme principal constitue le point d'entrée dans ACCPM. Dans cette procédure les différents paramètres sont initialisés. Nous ne rentrerons pas dans le détail de l'initialisation (procédure **Initialisation**). Indiquons simplement qu'il s'agit tout d'abord de lire dans un ficher input.dat les valeurs intrinsèques à la marche de l'algorithme (critère d'arrêt de l'algorithme ACCPM, critère d'arrêt du module projectif ...) et de lire les données décrivant le problème à résoudre (initialisation de l'oracle par l'intermédiaire d'un fichier : fichier.dat). C'est aussi dans cette procédure qu'est alloué l'espace mémoire pour les vecteurs et les matrices.

[1]Comparée à une autre politique qui consisterait à mémoriser les colonnes dans différentes matrices Ω

[2]Les colonnes de réalisabilité ou les colonnes provenant d'une même fonction f_i doivent être côte à côte dans $\tilde{G}$

Ensuite, la procédure **Master** est appellée. Elle va résoudre le problème chargé par la procédure **Initialisation** suivant la démarche définie lors de la lecture du fichier input.dat. Avant et après la procédure **Master** le temps est mesuré afin de pouvoir indiquer le temps qu'a pris cette résolution.

Enfin, la procédure **Résultat** affiche la solution, le temps et les différents paramètres désirés. Elle ne sera pas non plus détaillée.

Programme principal

Main ()
  $[\tilde{x}, u^c, \tilde{G}, \overline{z}, \underline{z}]$= **Initialisation** (input.dat, fichier.dat) ;
  tps := clock() ;
  $[u_{best}, \overline{z}]$= **Master** $(\tilde{x}, u^c, \tilde{G}, \overline{z}, \underline{z}, \varepsilon)$;
  $tps := clock() - tps$;
  Résultat $(u_{best}, \overline{z}, tps)$;
Fin.

8.4 Procédure Master

La procédure "**Master**" remplit les fonctions du programme-maître. Pour comprendre les paragraphes qui vont suivre on se référera à l'algorithme donné en fin de section (et inversement) et pour plus de détail aux explications données dans les chapitres précédents.

- Initialement le centre analytique u^c est choisi au milieu des boîtes. Ce point est envoyé à la procédure **Oracle**. Cette procédure étant dépendante du problème que l'on cherche à résoudre elle ne sera pas détaillée.

  Indiquons simplement que la procédure **Oracle** renvoie "nb_1" coupes. Dans notre implantation de la méthode nous avons considéré que toutes les coupes étaient du même type, c'est-à-dire soit des coupes de réalisabilité soit des coupes d'optimalité. Le paramètre "type" indique le type des coupes. Une coupe est décrite par $\omega^T u \geq \gamma_j$. La variable Ω est donc une matrice de "nb_1" colonnes et γ un vecteur de "nb_1" éléments. S'il s'agit de coupes d'optimalité ou si on n'est pas en phase II, l'oracle renvoie aussi la valeur de la fonction au point u^c : $\overline{z}'$. De plus, s'il s'agit d'une coupe d'optimalité, l'oracle renvoie dans le vecteur f les m valeurs des différentes fonctions de l'objectif (On rappelle que l'objectif est composé de m fonctions convexes).

- Ensuite, si l'oracle a renvoyé des coupes d'optimalité il est possible de passer directement en phase II. Toutes les composantes du vecteur $\tilde{x}$ sont alors initialisées à 1 et les coupes sont ajoutées à la matrice $\tilde{G}$ et au vecteur $\tilde{c}$ grâce à la procédure **Ajout**. Cette dernière présente la particularité de garder la structure de la matrice $\tilde{G}$ telle que nous l'avons toujours décrite et ceci pour rendre possible l'utilisation de cette structure lors de la factorisation. Elle doit donc insérer les coupes aux emplacements adéquats dans $\tilde{G}$ (et respectivement dans $\tilde{c}$). Devant la simplicité de la procédure **Ajout** il ne nous parait pas nécessaire d'approfondir sa description.

- Si l'oracle n'a pas renvoyé de coupes d'optimalité un passage par la phase I s'impose. Au préalable, des plans de supports horizontaux sont ajoutés à $\tilde{G}$ par l'intermédiaire de la procédure **Ajout**. Ils rendent possible le calcul de la factorisation de Cholesky de $\tilde{G}\tilde{X}^2\tilde{G}^T$ et permettent d'utiliser ainsi la procédure **Addcol** pour ajouter à $\tilde{G}$ les colonnes envoyées par l'oracle. Nous reviendrons plus en détail sur cette dernière procédure. Indiquons simplement que, non seulement, elle fait appel à la procédure **Ajout** pour ajouter dans $\tilde{G}$ et $\tilde{c}$ les coupes envoyées par l'oracle, mais que, de plus, elle détermine un nouveau vecteur $\tilde{x}$ tel que $\tilde{G}\tilde{x} = 0$.

- La phase I itère sur quatre procédures : **Modproj**, **Borninf**, **Oracle** et **Addcol** ; tant que l'oracle n'a pas renvoyé de coupes d'optimalité. D'une manière générale ces procédures forment le cœur de ACCPM. Les deux dernières ont déjà été commentées. La procédure **Modproj** contient l'algorithme projectif qui détermine le nouveau centre analytique. Il est donc logique d'obtenir les vecteurs u^c et $\tilde{s}$ en sortie de celle-ci. Durant l'algorithme projectif le vecteur $\tilde{x}$ va évoluer et comme celui-ci sert aussi dans d'autres procédures, il se trouve aussi en sortie de cette procédure. On trouve aussi en sortie de cette procédure la matrice R, factorisation de Cholesky de la matrice $\tilde{G}\tilde{X}^2\tilde{G}^T$ et la direction $\tilde{q}$. La procédure **Borninf** calcul une borne inférieure.

- Si durant la phase I l'oracle renvoie des coupes d'optimalité, on passe alors en phase II. Au préalable cependant, il faut supprimer de $\tilde{G}$ et $\tilde{c}$ les plans horizontaux. Ceci se fait par l'intermédiaire de la procédure **Suppression** qui, comme la procédure **Ajout**, garde la structure de la matrice $\tilde{G}$. Comme pour la procédure **Ajout** et pour les mêmes raisons on ne décrira pas la procédure **Suppression**. Ensuite, il faut transformer les coupes existantes dans $\tilde{G}$ en coupes de réalisabilité (c.à.d. mettre 0 dans chaque colonne à la place du 1 dans les lignes de convexités). Enfin, on appel la procédure **Zeromach** (zéro-machine) qui détermine un point quelconque $\tilde{x}$ tel que $\tilde{G}\tilde{x} = 0$.

- La phase II itère sur les mêmes procédures que la phase I plus quelques autres qui n'interviennent que pour améliorer les performances de l'algorithme. Elles ne sont donc pas nécessaire au fonctionnement du programme-maître. La première d'entres elles est la procédure **Ajustboite**. Elle permet d'ajuster les boîtes. Ceci se fait en fonction du centre analytique u^c. Si celui-ci se rapproche trop de l'une d'elles cette dernière est automatiquement éloignée[3] (changement de la composante correspondante dans $\tilde{c}$). Si les boites sont éloignées, la borne inférieure $\underline{z}$ n'est plus valable. Elle est alors mise à $-\infty$. De par sa simplicité, nous ne détaillerons pas plus cette procédure.

- On remarquera que l'oracle renvoie la valeur de la fonction en u^c dans la variable $\overline{z}'$. Cette valeur n'est pas nécessairement la meilleure borne supérieure que l'on ait obtenue (variable $\overline{z}$). S'il s'avère que c'est la cas $\overline{z}$ ainsi que $\tilde{c}_0$ est mis à jour et le vecteur u^c est mémorisé (solution optimale courante).

- La procédure **Testcol** supprime, des coupes envoyées par l'oracle, celles qui sont déjà présentes dans la matrice $\tilde{G}$. On a vu en effet que l'oracle pouvait générer plusieurs fois la même coupe. En sortie de cette procédure le nombre de colonnes dans Ω et le nombre de composantes dans γ peut avoir diminué. Il arrive même que toutes soit éliminées. Dans ce cas, il n'est pas nécessaire de faire appel à la procédure **Addcol**.

- Le test basé sur les ellipsoïdes est programmé dans la procédure **Testellipse**. Cette procédure n'est pas appelée toutes les itérations. Pour simplifier nous avons écrit dans l'algorithme que, si le protocole sur la fréquence d'appel indique “Ok”, alors on applique le test. On obtient alors en sortie, la liste des indices des colonnes de $\tilde{G}$ qui doivent être supprimées. Si cette liste est non vide ($nb_2 \neq 0$), on fait appel à la procédure **Suppression** qui retire de $\tilde{G}$ (respectivement $\tilde{c}$) les colonnes correspondantes. Il faut ensuite appellé la procédure **Realisabilite** qui détermine un nouveau point $\tilde{x}$ tel que $\tilde{G}\tilde{x} = 0$.

- La phase II se termine lorsque le saut de dualité relatif est plus petit que ε, paramètre lu lors de l'initialisation.

[3]On peut aussi envisager une version qui rapporche les boîtes.

Procédure Master

$[u_{best}, \overline{z}]$= **Master** $(\tilde{x}, u^c, \tilde{G}, \overline{z}, \underline{z}, \varepsilon)$
  $u^c := \frac{(h+l)}{2}\mathbf{1}$;
  $[\Omega, \gamma, \overline{z}', f, type, nb_1]$= **Oracle** (u^c) ;
  Si $type$ = coupe d'optimalité, **alors** % Préparation phase II %
    $\tilde{x} := \mathbf{1}$;
    $[\tilde{G}, \tilde{c}]$= **Ajout** $(\tilde{G}, \tilde{c}, \Omega, \gamma, type)$;
    $\tilde{c}_0 := \overline{z}'$; $\overline{z} := \overline{z}'$;
  Sinon % Préparation phase I %
    $[\tilde{G}, \tilde{c}]$= **Ajout** $(\tilde{G}, \tilde{c}, \mathbf{0}, 0,$ coupe d'optimalité$)$; % Plans horizontaux %
    $\tilde{x}_0 := 1$; $\tilde{x}_{1..2n} := \frac{\overline{z}}{h-l}\mathbf{1}$; $\tilde{x}_{2n+1...} := \mathbf{1}$;
    Factorisation élémentaire de $\tilde{G}\tilde{X}^2\tilde{G}^T = RR^T$;
    $\tilde{c}_0 := \overline{z}'$; $\underline{z} := 0$;
    $[\tilde{G}, \tilde{c}, \tilde{x}]$= **Addcol** $(\tilde{G}, \tilde{c}, \Omega, \gamma,$coupe d'optimalité$, nb_1, \tilde{x}, R)$;
  Fsi ;

  Tant que $type$ = coupe de réalisabilité, **faire** % Phase I %
    $[u^c, \tilde{x}, \tilde{s}, \tilde{q}, R]$= **Modproj** $(\tilde{x}, \tilde{c}, \tilde{G}, \varepsilon)$;
    $[\ \underline{z}]$= **Borninf** $(\tilde{x}, \tilde{c}, \tilde{G}, R, \tilde{q}, \underline{z})$;
    Si $\underline{z} > 0$, **alors** problème sans solution : **STOP fsi** ;
    $[\Omega, \gamma, \overline{z}', f, type, nb_1]$= **Oracle** (u^c) ;
    Si $type$ = coupe d'optimalité, **alors** % Préparation phase II %
      $[\tilde{G}, \tilde{c}]$= **Suppression** $(\tilde{G}, \tilde{c}, liste)$; % Sup. plans horizontaux %
      Transformation des contraintes existantes en contraintes de type J ;
      $[\tilde{G}, \tilde{c}]$= **Ajout** $(\tilde{G}, \tilde{c}, \Omega, \gamma, type)$;
      $[\tilde{x}]$= **Zeromach** $(\tilde{G}, \tilde{x})$;
      $\tilde{c}_0 := \overline{z}'$; $\overline{z} := \overline{z}'$; $\underline{z} := -\infty$;
    Sinon % Toujours phase I %
      $\tilde{c}_0 := \min\{\overline{z}', \tilde{c}_0\}$;
      $[\tilde{G}, \tilde{c}, \tilde{x}]$= **Addcol** $(\tilde{G}, \tilde{c}, \Omega, \gamma,$coupe d'optimalité$, nb_1, \tilde{x}, R)$;
    Fsi ;
  Fait ;

Tant que $\frac{\overline{z}-\underline{z}}{\max\{1,|\overline{z}|\}} > \varepsilon$, **faire** % Phase II %

$[\tilde{c}, \underline{z}]$= **Ajusteboite** $(\tilde{c}, u^c)$;

$[u^c, \tilde{x}, \tilde{s}, \tilde{q}, R]$= **Modproj** $(\tilde{x}, \tilde{c}, \tilde{G}, \varepsilon)$;

$[\underline{z}]$= **Borninf** $(\tilde{x}, \tilde{c}, \tilde{G}, R, \tilde{q}, \underline{z})$;

Si $\frac{\overline{z}-\underline{z}}{\max\{1,|\overline{z}|\}} > \varepsilon$, **alors**

$[\Omega, \gamma, \overline{z}', f, type, nb_1]$= **Oracle** (u^c) ;

Si $\overline{z}' < \overline{z}$, **alors**

$u_{best} := u^c$;

$\overline{z} := \overline{z}'$; $\tilde{c}_0 := \overline{z}'$;

Fsi ;

Si $\frac{\overline{z}-\underline{z}}{\max\{1,|\overline{z}|\}} > \varepsilon$, **alors**

Si $type$ = coupe d'optimalité, **alors**

$[\Omega, \gamma, nb_1]$= **Testcol** $(f, \tilde{s}, u^c, \Omega, \gamma, nb_1)$;

Fsi ;

Si protocole sur la fréquence = "Ok", **alors**

$[liste, nb_2]$= **Testellispe** $(\tilde{G}, \tilde{c}, \tilde{x}, \tilde{s}, R, u^c)$;

Fsi ;

Si $nb_1 \neq 0$, **alors**

$[\tilde{G}, \tilde{c}, \tilde{x}]$= **Addcol** $(\tilde{G}, \tilde{c}, \Omega, \gamma, type, nb_1\tilde{x}, R)$;

Fsi ;

Si $nb_2 \neq 0$, **alors**

$[\tilde{G}, \tilde{c}]$= **Suppression** $(\tilde{G}, \tilde{c}, liste)$;

$[\tilde{x}]$= **Realisabilite** $(\tilde{G}, \tilde{c}, \tilde{x})$;

Fsi ;

Fsi ;

Fsi ;

Fait ;

Fin.

8.5 Procédure Modproj

La procédure **Modproj** contient l'algorithme projectif qui va permettre de déterminer le centre analytique approché. Elle nécessite en entrée un vecteur $\tilde{x}$ tel que $\tilde{G}\tilde{x} = 0$.

La procédure **Modproj** commence par faire appel à la procédure **Zeromachine** afin de s'assurer que $\tilde{G}\tilde{x} = 0$. Elle détermine ensuite la factorisation de Cholesky de $\tilde{G}\tilde{X}^2\tilde{G}^T$. Nous ne détaillerons pas ces procédures élémentaires.

Enfin, la procédure **Modproj** fait appel à la procédure **Calq** qui va calculer la direction $\tilde{q}$. Nous avons vu que l'algorithme projectif effectuait des itérations tant que $\|\tilde{q}\| \geq \theta$, avec $\theta \leq 1$. Dans la pratique nous verrons (chapitre suivant) qu'il est judicieux de prendre $\theta = 1$.

En réalité, le test d'arrêt est plus sophistiqué. Tout d'abord, le centre analytique approché peut être difficile à déterminer (instabilités numériques). On limite alors la recherche à 40 itérations. Si après celles-ci on n'est pas parvenu à déterminer un centre analytique approché, on regarde si le point courant u^c appartient à l'ensemble de localisation $\mathcal{L}_{\overline{z}}$, c'est-à-dire si $\tilde{s} > 0$. Si c'est le cas on considère que u^c est acceptable et on sort de la procédure **Modproj**. Si ce n'est pas le cas on laisse alors une seconde chance à l'algorithme de trouver au moins un point tel que $\tilde{s} > 0$ en moins de quarante itérations. Si malgrés toute cette recherche on n'y parvient pas, on stoppe alors ACCPM. Ce cas extrême est très rare et provient toujours d'instabilités numériques dues à la nature du problème.

Ensuite, durant les itérations il peut arriver que $\tilde{c}^T\tilde{x} \leq \varepsilon$. Comme $\tilde{c}^T\tilde{x}$ mesure le saut de dualité et comme la borne supérieure $\overline{z}$ n'a pas changée, la borne inférieure s'est donc améliorée. On peut donc stopper l'algorithme ACCPM et afficher la solution trouvée.

Pour finir, indiquons que la procédure **recuniproj** effectue une recherche unidimensionnelle le long de la direction $\tilde{q}$ afin de déterminer le pas qui doit être effectué. Cette procédure sera décrite plus tard.

Procédure Modproj

$[u^c, \tilde{x}, \tilde{s}, \tilde{q}, R]$= **Modproj** $(\tilde{x}, \tilde{c}, \tilde{G}, \varepsilon)$
  $[\tilde{x}]$= **Zeromach** $(\tilde{G}, \tilde{x})$;
  $[R]$= **Factor** $(\tilde{G}, \tilde{x})$;
  $[\tilde{q}, u^c, \tilde{s}]$= **Calq** $(\tilde{G}, \tilde{x}, R)$;
  $k_2 := 0$;

  Répéter
    $k_2 := k_2 + 1$; $k_1 := 0$;
    Tant que ($\|q\| \geq 1$ **et** $\frac{\tilde{c}^T\tilde{x}}{\max\{1,|\bar{z}|\}} > \varepsilon$ **et** $k_1 < 40$), **faire**
      $k_1 := k_1 + 1$;
      $[\alpha]$= **Recuniproj** $(\tilde{q})$;
      $\tilde{x} := \tilde{x} + \frac{\alpha\tilde{X}\tilde{q}}{1+\alpha\tilde{x}_0\tilde{q}_0}$;
      Si $\frac{\tilde{c}^T\tilde{x}}{\max\{1,|\bar{z}|\}} > \varepsilon$, **alors**
        $[\tilde{x}]$= **Zeromach** $(\tilde{G}, \tilde{x})$;
        $[R]$= **Factor** $(\tilde{G}, \tilde{x})$;
        $[\tilde{q}, u^c, \tilde{s}]$= **Calq** $(\tilde{G}, \tilde{x}, R)$;
      Fsi ;
    Fait ;
  Tant que ($\min\{\tilde{s}\} < 0$ **et** $\frac{\tilde{c}^T\tilde{x}}{\max\{1,|\bar{z}|\}} > \varepsilon$ **et** $k_2 < 2$) ;
  Si $\min\{\tilde{s}\} < 0$, **alors STOP fsi** ;
Fin.

8.5.1 Procédure Calq

La procédure **Calq** détermine la direction $\tilde{q}$. Elle suit exactement les opérations qui ont été détaillées dans les chapitres précédents. Tout d'abord elle calcule les vecteurs u^c, $\tilde{s}$ et $\tilde{p} = P_{\mathcal{N}(\tilde{G}\tilde{X})}\tilde{X}\tilde{c}$. Ensuite, si des erreurs numériques trop grandes se sont introduites, elle améliore cette projection comme vu précédemment. Nous discuterons dans le dernier chapitre, ce que nous considérons comme "trop grandes". Remarquons simplement qu'il s'agit d'une fraction du saut de dualité relatif actuellement obtenu.

Enfin, cette procédure s'assure que la direction $\tilde{q}$ possède au moins une composante négative. Si ce n'est pas le cas, de trop grandes erreurs numériques se sont introduites et on stoppe l'algorithme ACCPM. En effet, on peut alors prouver que cela signifie que l'on a trouvé non seulement un vecteur tel que $\tilde{G}\tilde{x} = 0$ mais aussi tel que $\tilde{c}^T\tilde{x} = 0$ ce qui contredit l'hypothèse que l'ensemble de localisation $\mathcal{L}_{\overline{z}}$ à un itérieur non vide.

Procédure Calq

$[\tilde{q}, u^c, \tilde{s}]$= **Calq** $(\tilde{G}, \tilde{x}, R)$
  $u^c := -(RR^T)^{-1}\tilde{G}\tilde{X}^2\tilde{c}$;
  $\tilde{s} := \tilde{c} + \tilde{G}u^c$;
  $\tilde{p} := \tilde{X}\tilde{s}$;
  Si $\frac{\tilde{G}\tilde{X}\tilde{p}}{||\tilde{X}\tilde{p}||} > \frac{(\overline{z}-z)10^{-4}}{\max\{1,\overline{z}\}}$, **alors**
    $u^c := u^c - (RR^T)^{-1}\tilde{G}\tilde{X}\tilde{p}$;
    $\tilde{s} := \tilde{c} + \tilde{G}u^c$;
    $\tilde{p} := \tilde{X}\tilde{s}$;
  Fsi ;
  $\tilde{q} := 1 - \frac{\tilde{c}^T\tilde{x}}{\tilde{p}^T\tilde{p}}\tilde{p}$;
  Si $\min\{\tilde{q}\} \geq 0$, **alors STOP fsi** ;
Fin.

8.5.2 Procédure Recuniproj

La procédure **Recuniproj** effectue une recherche unidimensionnelle le long de la direction $\tilde{q}$ afin de déterminer le pas qui doit être effectué. Elle exécute exactement ce qui a été discuté au chapitre 6 en y ajoutant cependant quelques raffinements.

Le test d'arrêt tout d'abord indique que l'on va faire au maximum une centaine d'itérations. Si durant celles-ci la valeur de la dérivée au point courant α est 10^{-6} fois plus petite que la valeur de la dérivée à la première itération ou si la valeur de la dérivée au point courant α est plus petite que 10^{-6}, alors on considère que α est le paramètre recherché.

Durant les itérations il peut arriver que α sorte des limites fixées ($0 < \alpha < \alpha_{max}$). On le recentre alors en prenant pour α le milieu entre la borne violée et la valeur α précédente.

Si à une itération, la dérivée seconde s'annule (instabilités numériques), on arrête alors l'algorithme ACCPM. En réalité il faudrait envisager une méthode plus robuste comme par exemple la méthode de "dichotomie avec dérivée". Ce pas n'étant jamais survenu nous nous sommes contentés du test.

Procédure Recuniproj

$[\alpha]=$ **Recuniproj** $(\tilde{q})$

$\alpha_{max} := \frac{-1}{\min\{\tilde{q}\}}$;

$\alpha := \frac{\alpha_{max}}{2}$;

$k := 1$;

$\tilde{\psi}'(\alpha) := -\sum_{i=0}^{\kappa} \dfrac{\tilde{q}_i}{1+\alpha\tilde{q}_i}$;

$\tilde{\psi}''(\alpha) := \sum_{i=0}^{\kappa} \left(\dfrac{\tilde{q}_i}{1+\alpha\tilde{q}_i}\right)^2$;

$\varepsilon' := \max\{|\tilde{\psi}'(\alpha)|10^{-6}, 10^{-6}\}$;

Tant que ($|\tilde{\psi}'(\alpha)| > \varepsilon'$ **et** $k < 100$), **faire**

$k := k+1$;

Si $\tilde{\psi}''(\alpha) = 0$, **alors STOP fsi** ;

$\alpha_{old} := \alpha$;

$\alpha := \alpha - \frac{\tilde{\psi}'(\alpha)}{\tilde{\psi}''(\alpha)}$;

Si $\alpha \geq \alpha_{max}$, **alors** $\alpha := \frac{\alpha_{max}+\alpha_{old}}{2}$ **fsi** ;

Si $\alpha \leq 0$, **alors** $\alpha := \frac{\alpha_{old}}{2}$ **fsi** ;

$\tilde{\psi}'(\alpha) := -\sum_{i=0}^{\kappa} \dfrac{\tilde{q}_i}{1+\alpha\tilde{q}_i}$;

$\tilde{\psi}''(\alpha) := \sum_{i=0}^{\kappa} \left(\dfrac{\tilde{q}_i}{1+\alpha\tilde{q}_i}\right)^2$;

Fait ;

Fin.

8.6 Procédure Addcol

Cette procédure permet d'ajouter à $\tilde{G}$ et $\tilde{c}$ les informations envoyées par l'oracle. Elle permet de plus, de trouver un point $\tilde{x}$ tel que $\tilde{G}\tilde{x} = 0$ de manière à faire ensuite le moins d'itérations possible dans **modproj** pour trouver un nouveau centre analytique.

La première opération que l'on effectue consiste à aggéger les colonnes de Ω en une seule en complétant par des 1 s'il s'agit de coupes d'optimalité. On applique alors la procédure vue au chapitre 6 afin de déterminer un nouveau point $\tilde{x}$ tel que $\tilde{G}\tilde{x} = 0$. En résumé, on détermine une direction $\tilde{d}$ et le pas α que l'on doit faire le long de celle-ci. La procédure **Recuniadd** effectue la recherche unidimensionnelle qui détermine α.

Une fois $\tilde{d}$ et α connus on calcule les composantes de $\tilde{x}$ et on ajoute à $\tilde{G}$ et $\tilde{c}$ les différentes coupes par la procédure **Ajout**.

Procédure Addcol

$[\tilde{G}, \tilde{c}, \tilde{x}]$= **Addcol** $(\tilde{G}, \tilde{c}, \Omega, \gamma, type, nb_1, \tilde{x}, R)$
  $\omega := \Omega\mathbf{1}$;
  $c := \gamma^T\mathbf{1}$;
  Si $type$ = coupe d'optimalité, **alors**
    Ajouter pour chacunes d'elles un 1 à l'emplacement adéquat (ligne de convexité) dans ω ;
  Fsi ;
  $\tilde{d} := -\tilde{X}\tilde{G}^T(RR^T)^{-1}\omega$;
  $[\alpha]$= **Recuniadd** $(\tilde{d}, nb_1, \tilde{x}, c)$;
  $\tilde{x} := \tilde{x} + \frac{\alpha\tilde{X}\tilde{d}}{1+\alpha\tilde{x}_0\tilde{d}_0}$;
  $\tilde{x}_{\kappa+1\ldots\kappa+nb_1} := \frac{\alpha}{1+\alpha\tilde{x}_0\tilde{d}_0}$;
  $[\tilde{G}, \tilde{c}]$= **Ajout** $(\tilde{G}, \tilde{c}, \Omega, \gamma, type)$;
Fin.

8.6.1 Procédure Recuniadd

Cette procédure effectue une recherche unidimensionnelle pour déterminer le pas que l'on doit faire dans la direction $\tilde{d}$ déterminée par la procédure **Addcol**.

Les commentaires pour cette procédure sont les mêmes que ceux faits pour la procédure **Recuniproj**.

Procédure Recuniadd

$[\alpha]$= **Recuniadd** $(\tilde{d}, nb_1, \tilde{x}, c)$

$\alpha_{max} := \frac{-1}{\min\{\tilde{d}\}}$;

$\alpha := \frac{\alpha_{max}}{2}$;

$k := 1$;

$\bar{\psi}'(\alpha) := (\kappa + 1 + nb_1)\frac{\tilde{c}^T\tilde{X}\tilde{d}-c}{\tilde{c}^T\tilde{x}+\alpha(\tilde{c}^T\tilde{X}\tilde{d}-c)} - \frac{1}{\alpha} - \sum_{i=0}^{\kappa}\frac{\tilde{d}_i}{1+\alpha\tilde{d}_i}$;

$\bar{\psi}''(\alpha) := -(\kappa + 1 + nb_1)\left(\frac{\tilde{c}^T\tilde{X}\tilde{d}-c}{\tilde{c}^T\tilde{x}+\alpha(\tilde{c}^T\tilde{X}\tilde{d}-c)}\right)^2 + \frac{1}{\alpha^2} + \sum_{i=0}^{\kappa}\left(\frac{\tilde{d}_i}{1+\alpha\tilde{d}_i}\right)^2$;

$\varepsilon' := \max\{|\bar{\psi}'(\alpha)|10^{-6}, 10^{-6}\}$;

Tant que $(|\bar{\psi}'(\alpha)| > \varepsilon'$ **et** $k < 100)$, **faire**

$k := k + 1$;

Si $\bar{\psi}''(\alpha) = 0$, **alors STOP fsi** ;

$\alpha_{old} := \alpha$;

$\alpha := \alpha - \frac{\bar{\psi}'(\alpha)}{\bar{\psi}''(\alpha)}$;

Si $\alpha \geq \alpha_{max}$, **alors** $\alpha := \frac{\alpha_{max}+\alpha_{old}}{2}$ **fsi** ;

Si $\alpha \leq 0$, **alors** $\alpha := \frac{\alpha_{old}}{2}$ **fsi** ;

$\bar{\psi}'(\alpha) := (\kappa + 1 + nb_1)\frac{\tilde{c}^T\tilde{X}\tilde{d}-c}{\tilde{c}^T\tilde{x}+\alpha(\tilde{c}^T\tilde{X}\tilde{d}-c)} - \frac{1}{\alpha} - \sum_{i=0}^{\kappa}\frac{\tilde{d}_i}{1+\alpha\tilde{d}_i}$;

$\bar{\psi}''(\alpha) := -(\kappa + 1 + nb_1)\left(\frac{\tilde{c}^T\tilde{X}\tilde{d}-c}{\tilde{c}^T\tilde{x}+\alpha(\tilde{c}^T\tilde{X}\tilde{d}-c)}\right)^2 + \frac{1}{\alpha^2} + \sum_{i=0}^{\kappa}\left(\frac{\tilde{d}_i}{1+\alpha\tilde{d}_i}\right)^2$;

Fait ;

Fin.

8.7 Procédure Borninf

La procédure **Borninf** calcule une borne inférieure $\underline{z}$ suivant la démarche décrite au chapitre 6. En résumé, si le saut de dualité relatif est encore loin de celui recherché, on calcule une borne inférieure uniquement à l'aide de $\tilde{q}$; sinon, on cherche une borne inférieure (probablement) meilleure en utilisant d'autres vecteurs dans le noyau de $\tilde{G}\tilde{X}$ (par exemple : $\tilde{q}$, $\tilde{p} = P_{\mathcal{N}(\tilde{G}\tilde{X})}\binom{\overline{z}}{\mathbf{0}}$ et $\tilde{q} - \tilde{p}$).

Procédure Borninf

$[\underline{z}]=$ **Borninf** $(\tilde{x}, \tilde{c}, \tilde{G}, R, \tilde{q}, \underline{z})$
  Si $\frac{\overline{z}-\underline{z}}{\max\{1,\overline{z}\}} < 50\varepsilon$, **alors**
    $u' := -\overline{z}\tilde{x}_0(RR^T)^{-1}b$;
    $\tilde{s}' := \binom{\overline{z}}{\mathbf{0}} + \tilde{G}^T u'$;
    $\tilde{p} := \tilde{X}\tilde{s}'$;
    Si $\frac{\tilde{G}\tilde{X}\tilde{p}}{||\tilde{X}\tilde{p}||} > \frac{(\overline{z}-\underline{z})10^{-4}}{\max\{1,\overline{z}\}}$, **alors**
      $u' := u' - (RR^T)^{-1}\tilde{G}\tilde{X}\tilde{p}$;
      $\tilde{s} := \binom{\overline{z}}{\mathbf{0}} + \tilde{G}u'$;
      $\tilde{p} := \tilde{X}\tilde{s}$;
    Fsi ;
    Si $(\tilde{c}^T\tilde{X}\tilde{p} - \tilde{p}_0\tilde{c}^T\tilde{x} > 0)$, **alors** $\alpha := \min_{\tilde{p}_i<0}\{\tilde{p}_i\}$;
    Sinon $\alpha := \max_{\tilde{p}_i>0}\{\tilde{p}_i)\}$ **fsi** ;
    $\underline{z} := \max\{\underline{z}, \overline{z} - \frac{\tilde{c}^T\tilde{x}+\alpha\tilde{c}\tilde{X}\tilde{d}}{1-\tilde{q}_0/\alpha}\}$;   % Première borne inférieure %
    $\tilde{p} = \tilde{q} - \tilde{p}$;
    Si $(\tilde{c}^T\tilde{X}\tilde{p} - \tilde{p}_0\tilde{c}^T\tilde{x}) > 0$, **alors** $\alpha := \min_{\tilde{p}_i<0}\{\tilde{p}_i\}$;
    Sinon $\alpha := \max_{\tilde{p}_i>0}\{\tilde{p}_i)\}$ **fsi** ;
    $\underline{z} := \max\{\underline{z}, \overline{z} - \frac{\tilde{c}^T\tilde{x}+\alpha\tilde{c}\tilde{X}\tilde{d}}{1-\tilde{q}_0/\alpha}\}$;   % Deuxième borne inférieure %
  Fsi ;
  Si $\tilde{q}_0 > 0$, **alors** $\alpha := \min_{\tilde{q}_i<0}\{\tilde{q}_i\}$;
  Sinon $\alpha := \max_{\tilde{q}_i>0}\{\tilde{q}_i)\}$ **fsi** ;
  $\underline{z} := \max\{\underline{z}, \overline{z} - \frac{\tilde{c}^T\tilde{x}}{1-\tilde{q}_0/\alpha}\}$;   % Troisième borne inférieure %
Fin.

8.8 Procédure Testcol

La procédure **Testcol** n'est pas indispensable à ACCPM mais elle permet d'en améliorer les performances. Cette procédure filtre les coupes envoyées par l'oracle avant de les intégrer à $\tilde{G}$ et $\tilde{c}$. Pour réaliser ceci, elle prend pour chaque colonne de Ω toutes les colonnes de $\tilde{G}$ du même type[4] et regarde si l'une d'elles ne donne pas la même valeur que f_i pour le point courant u^c. Si c'est le cas la colonne Ω_i et la composante γ_i sont éliminées.

Procédure Testcol

$[\Omega, \gamma, nb_1]$= **Testcol** $(f, \tilde{s}, u^c, \Omega, \gamma, nb_1)$
  Pour $i = 1$ **à** nb_1, **faire**
    Pour toute les colonnes j de $\tilde{G}$ du même type i, **faire**
      Si $|f_i + \tilde{s}_j - u^c_{n+i}| < 10^{-6}$, **alors**
        Supprimer la colonne Ω_i ;
        Supprimer la composante γ_i ;
        $nb_1 := nb_1 - 1$;
        Sortir de la boucle sur j ;
      Fsi ;
    Fait ;
  Fait ;
Fin.

8.9 Procédure Testellipse

Cette procédure détermine la liste des colonnes à éliminer. Ce test est basé sur les ellipsoïdes inscrits et exinscrits comme vu au chapitre 7. En résumé, il calcule les distances des différentes coupes au centre analytique dans la métrique de l'ellipsoïde. Il effectue ensuite un tri par ordre décroissant par l'intermédiaire de la procédure

[4] On entend ici par "même type" le fait qu'elle soit une coupe d'optimalité de la même fonction f_i.

Tri. Cette dernière effectue un tri sur les distantes et renvoie la liste des indices des coupes correspondantes. Pour finir, en fonction des différents protocoles désirés on ne garde que les nb_2 premiers éléments de cette liste. Ces éléments indiquent alors les indices des coupes (des colonnes de $\tilde{G}$) qui doivent être supprimées.

La première partie de cette procédure est facultative. Elle a pour but d'ajuster les boîtes en fonction de l'ellipsoïde extérieur (voir chapitre 7).

Procédure Testellipse

$[liste, nb_2]$= **Testellipse** $(\tilde{G}, \tilde{c}, \tilde{x}, \tilde{s}, R, u^c)$
  $nb_2 = 0$;
  $\varepsilon_e := \left(\frac{\kappa}{\kappa+1}\right)^{\frac{1}{2}} \tilde{c}^T \tilde{x}$;
  Pour $j = 1$ **à** κ, **faire**
    $g := R^{-1}\tilde{G}_j$;
    Si $j \leq 2n$, **alors** % Ajustement des boîtes %
      Si $j \leq n$, **alors** % Boîtes Inférieures%
        $\tilde{c}_j = \varepsilon_e\sqrt{g^T g} - u^c_{j-1}$;
      Sinon % Boîtes Supérieures%
        $\tilde{c}_j = \varepsilon_e\sqrt{g^T g} + u^c_{j-1}$;
      Fsi ;
    Sinon % Calcul de la distance u^c-coupe j %
      $liste.dist_j := \frac{\tilde{s}_j}{\varepsilon_e\sqrt{g^T g}}$;
      $liste.ind_j := j$;
    Fsi ;
  Fait ;
  $[liste]$= **Tri** $(liste)$;
  Tant que il est possible d'éliminer, **faire** $nb_2 = nb_2 + 1$ **fait** ;
Fin.

8.10 Procédure Réalisabilité

La procédure **Realisabilite** détermine un vecteur $\tilde{x}$ tel que $\tilde{G}\tilde{x} = 0$ lorsque l'on a supprimé des colonnes dans $\tilde{G}$. Elle suit la démarche décrite au chapitre 7.

Procédure Realisabilite

$[\tilde{x}]=$ **Realisabilite** $(\tilde{G}, \tilde{c}, \tilde{x})$
  $[\tilde{x}]=$ **Zeromach** $(\tilde{G}, \tilde{x})$;
  $[R]=$ **Factor** $(\tilde{G}, \tilde{x})$;
  $u' := -\tilde{X}\tilde{G}(RR^T)^{-1}\tilde{G}\tilde{x}$;
  $\tilde{q} := \mathbf{1} + \tilde{X}\tilde{G}u'$;
  Si $\frac{\tilde{G}\tilde{X}\tilde{q}}{||\tilde{X}\tilde{q}||} > \frac{(\overline{z}-z)10^{-4}}{\max\{1,\overline{z}\}}$, **alors**
    $u' := u' - (RR^T)^{-1}\tilde{G}\tilde{X}\tilde{q}$;
    $\tilde{q} := \mathbf{1} + \tilde{X}\tilde{G}u'$;
  Fsi ;

  Tant que $\min\{tq\} < 0$, **faire**
    $[\alpha]=$ **Recuniproj** $(\tilde{q})$;
    $\tilde{x} := \tilde{x} + \frac{\alpha\tilde{X}\tilde{q}}{1+\alpha\tilde{x}_0\tilde{q}_0}$;
    $[\tilde{x}]=$ **Zeromach** $(\tilde{G}, \tilde{x})$;
    $[R]=$ **Factor** $(\tilde{G}, \tilde{x})$;
    $u' := -\tilde{X}\tilde{G}(RR^T)^{-1}\tilde{G}\tilde{x}$;
    $\tilde{q} := \mathbf{1} + \tilde{X}\tilde{G}u'$;
    Si $\frac{\tilde{G}\tilde{X}\tilde{q}}{||\tilde{X}\tilde{q}||} > \frac{(\overline{z}-z)10^{-4}}{\max\{1,\overline{z}\}}$, **alors**
      $u' := u' - (RR^T)^{-1}\tilde{G}\tilde{X}\tilde{q}$;
      $\tilde{q} := \mathbf{1} + \tilde{X}\tilde{G}u'$;
    Fsi ;
  Fait ;
  $\tilde{x} := \frac{\tilde{X}\tilde{q}}{\tilde{x}_0\tilde{q}_0}$;
Fin.

Bibliographie

[1] C. B. Moler, J. Little, S. Bangert et S. Kleiman, *Pro-Matlab User's Guide*, (MathWorks, Sherborn, M.A., 1987).

Chapitre 9

Etude expérimentale

L'algorithme ACCPM que nous avons décrit dans les chapitres précédents a été testé dans de nombreuses versions et sur de nombreux problèmes. Bien que les chapitres précédents soient relativement détaillés, beaucoup de questions restent en suspens. Notamment :

- Quelle factorisation doit être choisie (QR, Cholesky, ou autre)?
- Comment la taille des boîtes influence-t-elle l'algorithme?
- Quelle doit être la valeur de θ pour le test d'arrêt dans le calcul du centre analytique approché?
- Quand et comment doit être appliqué le test d'élimination des colonnes de $\tilde{G}$?
- Quelle précision demander dans les recherches unidimensionnelles?

L'espace qui existe entre la théorie et la pratique ne peut être comblé que par l'expérimentation numérique. C'est pourquoi, dans le présent chapitre nous allons tenter, en nous appuyant sur des problèmes classiques de l'optimisation, de répondre le plus précisément possible à ces questions.

Nous commencerons tout d'abord par décrire les différents problèmes sur lesquels nous avons testé l'algorithme. Puis, nous discuterons le choix des paramètres de la méthode. Ensuite nous donnerons les résultats numériques obtenus pour chacun des problèmes et nous terminerons par une rapide comparaison entre ACCPM et l'algorithme de Dantzig-Wolfe.

9.1 Les problèmes tests

Commençons tout d'abord par décrire succintement les problèmes sur lesquels nous avons testé l'algorithme ACCPM. Dans la mesure du possible, il sera donné pour chacun d'eux la description du problème et de l'oracle.

9.1.1 Problèmes de programmation géométrique

Description du problème

La forme standard de ces problèmes est :

$$\begin{array}{llll} \text{minimiser} & \sum_{i=1}^{n_0} d_i \prod_{j=1}^{n} r_j^{a_{ij}} & & \\ \text{s.c.} & & & \\ & \sum_{i=n_{t-1}+1}^{n_t} d_i \prod_{j=1}^{n} r_j^{a_{ij}} & \leq 1, & t = 1, \ldots, p \\ & r_j & > 0, & j = 1, \ldots, n. \end{array}$$

Pour les résoudre nous avons appliqué la transformation $r_j = \exp(v_j)$. Nous avons donc travaillé sur des problèmes du type :

$$\begin{array}{rl} \text{Minimiser} & f(y) \\ \text{s.c.} & g(y) \leq 0 \\ & l \leq y \leq h, \end{array}$$

où $f, g : \mathbb{R}_+^n \to \mathbb{R}$ sont les suivantes :

$$\begin{aligned} f(y) &= \sum_{i=1}^{n_0} d_i \exp\Big(\sum_{j=1}^{n} a_{ij} y_j\Big) \\ g(y) &= \max\{g_t(y) : t = 1, \ldots, p\}, \end{aligned}$$

avec

$$g_t(y) = \sum_{i=n_{t-1}+1}^{n_t} d_i \exp\Big(\sum_{j=1}^{n} a_{ij} y_j\Big) - 1,$$

et où d_i est positif ; l et h sont des vecteurs de $\mathbb{R}^n$ satisfaisant $l < h$; et où $n_0, \ldots, n_p$ sont des entiers positifs.

Ces problèmes sont au nombre de 16. Les références sur les 14 premiers peuvent se trouver dans [15]. Les derniers (problèmes 12 et 12b) nous ont été directement fournis par Kortanek. Leurs caractéristiques se trouvent dans le tableau 9.1.

Problème	Nombre de variables	Nombre de contraintes	Valeur du minimum
1	2	1	2.00000
3	3	1	$6.29984\ 10^3$
2	4	2	0.01210
4	4	1	$1.26303\ 10^5$
6	4	3	$6.23250\ 10^5$
8	7	7	$1.78477\ 10^2$
9a	7	4	$1.80976\ 10^3$
9b	7	4	$9.11880\ 10^2$
9c	7	4	$5.43668\ 10^2$
7a	8	7	29.2295
7b	8	7	29.2264
10	10	7	$3.23732\ 10^7$
5	11	3	3.18325
11	22	36	$1.80322\ 10^3$
12	30	10	$3.34556\ 10^4$
12b	30	39	$3.25145\ 10^4$

Tableau 9.1: Problèmes de programmation géométrique : caractéristiques.

Description de l'oracle

Pour un point y^c donné, si $g(y^c) \leq 0$ l'oracle renvoie la coupe d'optimalité suivante :

$$z \geq \omega^T y + \gamma,$$

où $\omega = \nabla f(y^c)$ et $\gamma = f(y^c) - \nabla f(y^c)^T y^c$.

Si, par contre, $g(y^c) > 0$ alors l'oracle sélectionne $\bar{t} = \mathrm{argmax}_t\{g_t(y)\}$ et renvoie la coupe de réalisabilité suivante :

$$\omega^T y \leq \gamma,$$

où $\omega = \nabla g_{\bar{t}}(y^c)$ et $\gamma = \nabla g_{\bar{t}}(y^c)^T y^c - g_{\bar{t}}(y^c)$.

Bien que nous ayons utilisé cet oracle, bien d'autres versions auraient pu être envisagées. La plus élementaire consisterait à envoyer toujours la coupe d'optimalité[1] et toutes les coupes de réalisabilité (un plan de support par contrainte violée). Une autre possibilité, serait de décomposer l'objectif en une somme de n_0 fonctions convexes. Cette dernière version conduirait très probablement à atteindre l'optimum en faisant moins d'appels à l'oracle (moins d'itérations de l'algorithme ACCPM).

9.1.2 Problèmes de multiflots

Description du problème

Nous utilisons ici la formulation de Minoux [17]. Le problème se définit par un graphe orienté[2] $\mathcal{G} = (\mathcal{V}, \mathcal{A})$, d'un ensemble de biens $\mathcal{I}$ décrits par un ensemble de vecteurs exogènes $d^i = (d^i_s)_{s\in\mathcal{V}}$ (indiquant les ressouces et les demandes) tel que $\mathbf{1}^T d^i = 0$. Des flots doivent être transportés dans le graphe $\mathcal{G}$. Ainsi, si $x^i = (x^i_a)_{a\in\mathcal{A}}$ dénote le flot du bien i et $\mathcal{N}$ est la matrice d'incidence sommet-arc du graphe $\mathcal{G}$, les flots réalisables de i appartiennent à :

$$\mathcal{F}^i = \{x^i \geq 0 : \mathcal{N}x^i = d^i\}.$$

Dans notre étude nous nous sommes restreints aux problèmes ayant une unique origine et une unique destination pour chaque bien, c'est-à-dire aux problèmes pour lesquels les vecteurs d^i ont exactement une composante positive et une composante négative.

Un coût linéaire c^i est associé au flot de chaque bien. Un coût non linéaire (ou linéaire) est associé au flot total $\sum_{i\in\mathcal{I}} x^i_a$ passant sur un arc.

[1] La phase I ne serait alors plus nécessaire.

[2] Ce généralise aisément à un graphe non orienté.

Définissons $\mathcal{A}' \subset \mathcal{A}$ le sous ensemble des arcs ayant soit un coût non linéaire soit une capacité limitée du flot total pouvant le traverser.

Le problème de multiflots peut être formuler de la manière suivante :

$$
\begin{aligned}
\text{Minimiser} \quad & \sum_{i \in \mathcal{I}} {c^i}^T x^i + \sum_{a \in \mathcal{A}'} f_a(y_a) & \\
\text{s.c.} \quad & \sum_{i \in \mathcal{I}} x_a^i \le y_a, & \forall a \in \mathcal{A}' \subset \mathcal{A} \\
& x^i \in \mathcal{F}^i, & \forall i \in \mathcal{I} \\
& 0 \le y_a \le \gamma_a, & \forall a \in \mathcal{A}' \subset \mathcal{A}.
\end{aligned}
$$

Dans cette formulation les composantes du vecteur $y = (y_a)_{a \in \mathcal{A}'}$ sont censées représenter le flot total circulant sur l'arc a. En réalité, ce sont des bornes supérieures. Cependant, en faisant l'hypothèse que les fonctions $f_a(y_a)$ sont strictement croissantes, à l'optimum, nous aurons bien $\sum_{i \in \mathcal{I}} x_a^i = y_a$. On fait, de plus, l'hypothèse que les fonctions $f_a(y_a)$ sont convexes et que les coûts c_i sont positifs. Ceci assure que les multiplicateurs de Lagrange (prix des contraintes couplantes) sont non-négatifs dans l'algorithme ACCPM. De cette dernière remarque découle que l'oracle n'a pas à se soucier des cycles négatifs.

Les deux derniers types de contraintes sont des contraintes de réseaux et de capacités respectivement. Si nous n'avions pas le premier type de contraintes, ce problème serait séparable. Les premières contraintes sont donc appelées contraintes couplantes puisqu'elles coordonnent les différents réseaux.

Dans les problèmes que nous avons résolus la fonction $f_a(y_a)$ a eu deux formulations :

- Version non linéaire du problème : $f_a(y_a) = \frac{y_a}{\gamma - y_a}$.
- Version linéaire du problème : $f_a(y_a) = 0$.

On remarquera que dans la version linéaire du problème, il est possible de simplifier la formulation en agrégeant le premier et le troisième type de contraintes. On obtient :

$$
\begin{aligned}
\text{Minimiser} \quad & \sum_{i \in \mathcal{I}} {c^i}^T x^i & \\
\text{s.c.} \quad & \sum_{i \in \mathcal{I}} x_a^i \le \gamma_a, & \forall a \in \mathcal{A}' \subset \mathcal{A} \\
& x^i \in \mathcal{F}^i, & \forall i \in \mathcal{I}.
\end{aligned}
$$

Les problèmes auxquels nous sommes confrontés dans la réalité peuvent être très grands. Par exemple, un problème avec 500 nœuds, 1000 arcs et 5000 biens conduit à un programme linéaire de 2.5 10^6 contraintes et 5 10^6 variables. Pour pouvoir les résoudre nous avons dualisé les contraintes couplantes pour définir le Lagrangien partiel suivant :

$$L(x;y;u) = \sum_{i\in\mathcal{I}} {c^i}^T x^i + \sum_{a\in\mathcal{A}'} f_a(y_a) + \sum_{a\in\mathcal{A}'} u_a(\sum_{i\in\mathcal{I}} x^i_a - y_a),$$

où u_a est le multiplicateur de Lagrange.

Il est alors possible de définir le problème :

$$L_P(x;y) = \max_{u\geq 0} L(x;y;u),$$

et le problème :

$$L_D(u) = \min_{\substack{x^i\in\mathcal{F}^i\\ 0\leq y\leq\gamma_a}} L(x;y;u).$$

Nous avons vu, section § 1.2.2, à propos de la “décomposition par les contraintes”, que $L_P(x;y)$ est convexe, $L_D(u)$ est concave et :

$$\min_{\substack{x^i\in\mathcal{F}^i\\ 0\leq y\leq\gamma_a}} L_P(x;y) = \max_{u\geq 0} L_D(u).$$

Etant donné que la dimension du vecteur u est beaucoup plus petite que la dimension du vecteur $(x^T, y^T)^T$ nous avons résolu le problème :

$$\max_{u\geq 0} L_D(u).$$

Les tests ont été effectués sur les problèmes standards non différentiables NDO22 et NDO148 [10]. Leurs caractéristiques se trouvent dans le tableau 9.2.

Description de l’oracle

La fonction $-L_D(u)$ est convexe. L’objectif est donc de déterminer le minimum d’une fonction convexe non contrainte avec zéro comme borne inférieure pour la boîte. la fonction $L_D(u)$ s’écrit :

$$L_D(u) \;=\; \min_{\substack{x^i\in\mathcal{F}^i\\ 0\leq y\leq\gamma_a}} \left\{ \sum_{i\in\mathcal{I}} {c^i}^T x^i + \sum_{a\in\mathcal{A}'} f_a(y_a) + \sum_{a\in\mathcal{A}'} u_a(\sum_{i\in\mathcal{I}} x^i_a - y_a) \right\}$$

Problème	Nœuds	Arcs	Biens	Optimum
NDO22 linéaire	14	22	23	-1328.99
NDO22 non linéaire	14	22	23	-103.412
NDO148 linéaire	61	148	122	-178551
NDO148 non linéaire	61	148	122	-151.926

Tableau 9.2: Problèmes de multiflots : caractéristiques.

$$\begin{aligned} &= \min_{x^i \in \mathcal{F}^i} \sum_{i \in \mathcal{I}} (c^i - u)^T x^i + \min_{0 \le y \le \gamma_a} \sum_{a \in \mathcal{A}'} (f_a(y_a) - u_a y_a) \\ &= \sum_{i \in \mathcal{I}} \min_{x^i \in \mathcal{F}^i} (c^i - u)^T x^i + \sum_{a \in \mathcal{A}'} \min_{0 \le y \le \gamma_a} (f_a(y_a) - u_a y_a). \end{aligned}$$

On peut donc exprimer la fonction $L_D(u)$ ainsi :

$$L_D(u) = \sum_{i \in \mathcal{I}} L_D^i(u) + \sum_{a \in \mathcal{A}'} L_D^a(u), \tag{9.1}$$

où

$$L_D^i(u) = \min_{x^i \in \mathcal{F}^i} (c^i - u)^T x^i, \tag{9.2}$$

est un problème de plus court chemin à coût non négatif $(c^i - u)$ et où

$$L_D^a(u) = \min_{0 \le y \le \gamma_a} (f_a(y_a) - u_a y_a), \tag{9.3}$$

suppose la minimisation d'une fonction convexe à une dimension dont la solution peut généralement être donnée analytiquement.

Pour un point u^c envoyé par le programme-maître[3], il est donc possible de calculer $L_D(u^c)$.

comme :

$$\nabla L_D(u) = \sum_{i \in \mathcal{I}} x_a^i - y_a.$$

Soit $x(u^c)$ et $y(u^c)$ les solutions optimales de (9.2) et (9.3) obtenues pour u^c, on déduit la coupe d'optimalité :

$$z \ge \omega^T u + \gamma,$$

où $\omega = -\nabla L_D(u^c)$ et $\gamma = -L_D(u^c) + \nabla L_D^T(u^c) u^c$, avec $\nabla L_D(u) = \sum_{i \in \mathcal{I}} x_a^i(u^c) - y_a(u^c)$.

[3]Dans la partie théorique le point u^c est noté y^c.

De la formulation (9.1) où $L_D(u)$ s'écrit comme la somme de fonctions convexes, on peut déduire un oracle fournissant pour u^c un ensemble de coupes d'optimalité. Bien que nous ne l'ayons pas utilisé, Goffin, Gondzio, Sarkissian et Vial [11] ont montré qu'il permettait de faire converger l'Algorithme ACCPM en beaucoup moins d'itérations.

9.1.3 Problèmes linéaires stochastiques

Description du problème

Dans de nombreux cas on est amené à prendre une décision en fonction d'un avenir incertain. Nous en donnons deux exemples :

- Une entreprise désire ajuster son portefeuille d'actifs et de dettes à court terme, en minimisant le coût net dû au surplus de liquidités et aux déficits. L'ajustement se fait sur un horizon fixé. L'incertitude provient de la méconnaissance, dans le futur, des besoins en liquidités ainsi que certains coûts de transaction. On trouvera une description complète de ce problème dans [12].

- Les choix énergétiques et environnementaux impliquent des immobilisations de capital à long terme (par exemple, des investissements dans de nouvelles technologies) qui exposent les agents économiques à un risque important. De plus, du fait des incertitudes technologiques, économiques et environnementales, l'évaluation conjointe des technologies et du risque devient nécessaire, tant pour l'élaboration d'une politique de l'énergie au niveau régional que pour la gestion des entreprises productrices d'énergie. Grâce à la programmation stochastique, nous disposons d'une méthode d'évaluation de *stratégies*, c'est-à-dire de façon de réagir aux événements [9].

- D'autres exemples peuvent se trouver dans [7].

Dans ces deux cas (comme dans bien d'autres) la programmation stochastique offre un outil permettant de modéliser correctement ces problèmes. L'idée de base consiste à établir une série de scénarios de ce que sera fait "demain" et de les lier au temps présent en les pondérant d'une probabilité. La décision que l'on prendra cherchera ainsi, non seulement à satisfaire au mieux le problème d'aujourd'hui, mais elle prendra aussi en compte ce qui arrivera probablement demain. De plus, elle adoptera souvent une attitude médiane permettant ainsi une reconversion plus facile dans l'avenir en fonction de ce qu'il adviendra réellement. Ce type d'approche est dénommé programmation stochastique à deux étapes (aujourd'hui et demain). Il est possible de généraliser ce concept pour aboutir à un problème stochastique à plusieurs étapes. Le problème à trois étapes consiste à supposer récursivement que l'attitude qui sera adopté demain est aussi conditionnée par un programme stochastique à deux étapes tenant compte d'après demain.

La formulation générale d'un problème linéaire stochastique à deux étapes [2] est la suivante :

$$\begin{aligned} \text{Minimiser} \quad & c_0^T x_0 + \textstyle\int_{\xi \in \Xi} Q(x_0, \xi) \\ \text{s.c.} \quad & A_0 x_0 \geq b_0, \end{aligned} \tag{9.4}$$

où x_0 est la variable de décision de première période et ξ une variable aléatoire de support Ξ. la fonction $Q : \mathbb{R}^n \times \Xi \mapsto \mathbb{R} \cup \{\infty\}$, est définie par :

$$Q(x_0, \xi) = \inf_x \{c^T(\xi)x \mid B(\xi)x \geq b(\xi) - A(\xi)x_0\}.$$

L'argument x dans la minimisation, parfois noté $x(\xi)$, repésente le recours. $Q(x_0, \xi)$ est alors le coût minimum engendré par le recours pour une réalisation donnée de la variable ξ. Il est connu que $Q(x_0, \xi)$, tout comme $\int_{\xi \in \Xi} Q(x_0, \xi)$, sont des fonctions convexes de x_0. On fait les hypotèses suivantes :

Hypothèse : 9.1

1. **Recours réalisable :** *Il existe x_0 tel que $A_0 x_0 \geq b_0$ et l'intégrale $\int_{\xi \in \Xi} Q(x_0, \xi)$ est finie.*
2. **L'échantillon est fini :** $\Xi = \{1, \ldots, I\}$ *est fini et* $P(\xi = i) = p_i > 0$.

La seconde partie de l'hypothèse assure l'équivalence entre le problème (9.4) et le programme convexe suivant :

$$\begin{aligned} \text{Minimiser} \quad & c_0^T x_0 + \sum_{i=1}^{I} f_i(x_0) \\ \text{s.c.} \quad & A_0 x_0 \geq b_0, \end{aligned}$$

avec

$$f_i(x_0) = \min_x \{c_i^T x : \quad B_i x \geq b_i - A_i x_0\}, \qquad i = 1, \ldots, I. \tag{9.5}$$

Les coefficients c_i sont donnés par : $c_i = p_i c(\xi = i)$.

Ce problème peut aussi se formuler sous la forme d'un programme linéaire ayant la structure particulière suivante :

$$\begin{aligned} \text{Minimiser} \quad & c_0^T x_0 + \sum_{i=1}^{I} c^{i^T} x^i \\ \text{s.c.} \quad & A_0 x_0 \geq b_0 \\ & A_i x_0 + B_i x_i \geq b_i, \qquad\qquad i = 1, \ldots, I, \end{aligned} \tag{9.6}$$

dont le dual est :

$$\begin{aligned} \text{Maximiser} \quad & b_0^T u_0 + \sum_{i=1}^{I} b^{i^T} u^i \\ \text{s.c.} \quad & \sum_{i=0}^{I} A_i^T u_i = c_0, \\ & B_i^T u_i = c_i, \qquad\qquad i = 1, \ldots, I \\ & u_i \geq 0, \qquad\qquad i = 0, \ldots, I. \end{aligned} \tag{9.7}$$

Le problème (9.6) est, en accord avec la terminologie traditionnelle, dual bloc angulaire. Il peut donc être résolu par l'approche de Benders [3]. Le problème (9.7) est lui, primal bloc angulaire. Il peut être résolu par la décomposition de Dantzig-Wolfe [6].

Comme nous avons vu à la section 1.2.2, les problèmes (9.6) et (9.7) peuvent être récrits comme la miminisation d'une fonction convexe à laquelle on se propose d'appliquer ACCPM.

Pour les expérimentations numériques nous avons utilisé des problèmes de planification de production sur plusieurs périodes avec une demande stochastique [8] (app0$\times$).

Nous avons aussi résolu un problème d'affectation dans le domaine aérien (problèmes 750.$\times$). Ce problème est décrit dans le livre de Dantzig [5]. Il présente l'intérêt d'offrir la possibilité de faire varier le nombre de blocs sur un large interval (1 à 750). Le problème 750.1 est donc identique au problème 750.750 excepté que dans le 1$^{\text{er}}$ cas on considère qu'il n'est constitué que d'un seul bloc tandis que dans le second de 750 blocs.

Les caractéristiques de ces problèmes se trouvent dans le tableau 9.3.

Problème	Taille	Dim. de x_0	Card. de I	Dimension de B_i^T	Dimension de A_0^T
App0110	268×129	28	3	80×40	28×9
App0112	372×155	12	3	120×50	12×5
App0210	4000×1892	280	3	1240×620	280×32
App0212	5640×1974	60	3	1860×651	60×21
750.1	7517×3754	17	1	7500×3750	17×4
750.2	7517×3754	17	2	3750×1875	17×4
750.10	7517×3754	17	10	750×375	17×4
750.50	7517×3754	17	50	150×75	17×4
750.75	7517×3754	17	75	100×50	17×4
750.125	7517×3754	17	125	60×30	17×4
750.250	7517×3754	17	250	30×15	17×4
750.750	7517×3754	17	750	10×5	17×4

Tableau 9.3: Problèmes linéraires stochastiques : caractéristiques.

Description de l'oracle

En définissant les ensembles Y_i suivant[4] :

$$\begin{aligned} Y_i &= \{x_0 \in \mathbb{R}^n | \exists x_i : A_i x_0 + B_i x_i \geq b_i\} \\ &= \{x_0 \in \mathbb{R}^n | f_i(x_0) < +\infty\}, \qquad i = 1, \ldots, I, \end{aligned}$$

et par dualité de (9.5) les fonctions $f_i(x_0)$:

$$f_i(x_0) = \max_{u_i}\{(b_i - A_i x_0)^T u_i | B_i^T u_i = c_i, u_i \geq 0\}, \qquad i = 1, \ldots, I, \tag{9.8}$$

il est possible de préciser l'oracle. Soit x_0^k tel que $A_0 x_0^k \geq b_0$, le point envoyé par le programme-maître. La réponse de l'oracle est :

- **Coupe de réalisabilité :** S'il existe un (ou plusieurs) ensemble Y_i tel que $x_0^k \notin Y_i$, l'oracle renvoie un (ou plusieurs) plan séparant x_0^k de Y_i. Pour un point x_0^k et un ensemble Y_i tel que $x_0^k \notin Y_i$, le $i^{\text{ème}}$ sous-problème (9.5) est non réalisable ou, de façon équivalente, le $i^{\text{ème}}$ sous-problème (9.8) est non borné. En utilisant la formulation (9.8) il est possible d'extraire un rayon extrême u_i^r.

[4]La première partie de l'hypotèse assure que $\cap_{i=0}^I Y_i$ avec $Y_0 = \{x_0 : A_0 x_0 \geq b_0\}$ est non vide.

Ce vecteur satisfait les conditions : $\alpha B_i^T u_i^r = c_i$ ($\forall \alpha > 0$) et $(b_i - A_i x_0)^T u_i^r > 0$. La deuxième condition permet de générer la coupe :

$$g_i^T (x_0 - x_0^k) \leq d_i,$$

avec $g_i = -A_i^T u_i^r$ et $d_i = -(b_i - A_i x_0^k)^T u_i^r$. En effet, on obtient alors l'inégalité :

$$b_i^T u_i^r - x_0^T A_i^T u_i^r \leq 0.$$

En posant $\gamma_k = b_i^T u_i^r$ et $\Omega_k = A_i^T u_i^r$ on trouve les composantes de $\tilde{G}$ comme nous les avons définies avec la première inégalité de (4.7).

- **Coupe de réalisabilité :** Si, par contre, x_0^k appartient à l'intersection des ensembles Y_i, l'oracle va renvoyer un plan de support de chacune des fonctions f_i. Soit u_i^p le point extrême optimum de f_i pour x_0 fixé, le plan de support s'écrit :
$$f_i(x_0^k) + g_i^T (x_0 - x_0^k) \leq z_i,$$
où $g_i = -A_i^T u^p$ et $f_i(x_0^k) = (b_i - A_i x_0^k)^T u_i^p$. En Posant $\gamma_k = b_i^T u_i^p$ et $\Omega_k = A_i^T u_i^p$ on trouve les composantes de $\tilde{G}$ comme nous les avons définies avec la deuxième inégalité de (4.7).

On remarquera que tout problème ayant la structure des problèmes (9.6) ou (9.7) peut être résolu par ACCPM en utilisant cet oracle.

On a vu que le programme-maître doit envoyer un point x_0 tel que $A_0 x_0 \geq b_0$. Cela implique que, pour ce type de problème, l'algorithme doit être légèrement modifié. Avant toute opération il faut intégrer la matrice A_0 dans $\tilde{G}$ comme s'il s'agissait de contraintes de type J (c.à.d. contraintes de réalisabilité). Puis on commence l'algorithme normalement (c.à.d. on envoie à l'oracle un point x_0 au millieu des boîtes et satisfaisant ces contraintes).

9.1.4 Problèmes Maxquad

Description du problème

Ce problème nous a été fourni par Lemaréchal [13]. Il s'agit du problème minmax suivant :

$$f(u) = \max_{i=1,\ldots,5} \{u^T A^i u - u^T b^i\},$$

où $u \in \mathbb{R}^{10}$.

Description de l'oracle

L'objectif est de déterminer le minimum d'une fonction convexe non contrainte $f(u)$. Soit j la forme quadratique qui définit $f(u^c)$, la coupe d'optimalité est :

$$z \geq \omega^T u + \gamma,$$

où $\omega = 2A^j u^c - b^j$ et $\gamma = f(u^c) - (2A^j u^c - b^j)u^c$.

9.1.5 Problèmes de dimensionnement optimal de structures métalliques

Description du problème

Les problèmes de dimensionnement optimal de structures métalliques [14] sont issus de modèles étudiant le déplacement de ces structures lorsqu'elles sont soumises à l'influence de forces extérieures. Les structures étudiées sont localisées dans l'espace à deux dimensions. Elles ont m barres fixées entres elles par leurs extrémités, appelées nœuds. La structure est soumise à une force extérieure $f \in \mathbb{R}^n$ appliquée sur un ou plusieurs nœuds et la déformation qu'elle induit est descrite par un vecteur $x \in \mathbb{R}^n$ indiquant le déplacement de ceux-ci.

Soient $A_i \in \mathbb{R}^{n\times n}$ la matrice symétrique semi-définie positive indiquant la raideur[5] de la $i^{\text{ème}}$ barre et t_i son volume. L'objectif est de trouver une structure topologique dont les volumes t_i sont optimaux afin qu'elle soit la plus rigide possible (pour un chargement donné). Le problème a la formulation suivante :

$$\begin{array}{ll} \min\limits_{t\in\mathbb{R}^m} & \frac{1}{2} f^T x \\ \text{s.c.} & A(t)x = f \\ & t^T \mathbf{1} = v \\ & l \leq t \leq h, \end{array}$$

où la matrice $A(t) = \sum_{i=1}^m t_i A_i$, est appelée matrice de raideur globale et v est le volume de matière aloué pour la structure. Malheureusement, non seulement ce problème à un grand nombre de variables ($n + m$ où $m \gg n$) mais il est de plus

[5]Chaque barre a une constante de raideur.

non convexe en (x, t). Cependant, Ben-Tal et Bendsøe [4] ont montré l'équivalence de celui-ci avec le problème suivant défini dans $\mathbb{R}^{n+1}$:

$$\max_{x\in\mathbb{R}^n,\lambda\in\mathbb{R}} f^T x - \lambda v + \sum_{i=1}^{m} \min\left\{(\lambda - \frac{1}{2}x^T A_i x)l_i, (\lambda - \frac{1}{2}x^T A_i x)h_i\right\}.$$

Ce problème est concave et de plus petite dimension que le premier. Cependant, il n'est pas différentiable.

Les problèmes sur lesquels nous avons travaillé n'ont pas de bornes sur les volumes t_i, c'est-à-dire $l_i = 0$ et $h_i = v$. Ben-Tal et Bendsøe [4] ont montré que le problème se réduisait alors au problème minmax suivant :

$$\min_{x\in\mathbb{R}^n} \max_{i=1,\dots,m} \left\{\frac{v}{2}x^T A_i x - f^T x\right\}.$$

Les structures que nous avons utilisées sont rectangulaires. Elles sont composées de $n_x \times n_y$ nœuds équidistants formant un graphe complet. Les nœuds du coté gauche du rectangle sont fixes ; la force est verticale et appliquée au milieu du coté droit. L'origine du repère de l'espace est le nœud en bas à gauche. Le déplacement se fait suivant l'axe x et y. La dimension du problème pour ACCPM est donc 2 fois le nombre de nœuds (déplacement dans les 2 dimensions de chacun d'eux). On déduit donc que le plus gros des problèmes (T1) consiste à trouver le minimum d'une fonction convexe non différentiable dans l'espace $\mathbb{R}^{342}$.

Les problèmes que nous avons résolus nous ont été fournis par Zowe leur description se trouve dans [14] et dans le tableau 9.4.

Problème	Nombre de nœuds	Nombre de barres	Valeur du minimum
T3	5×11= 55	934	-5.45431
T2	9×9= 81	2040	-23.8751
T1	9×19=171	8958	-21.5556

Tableau 9.4: Problèmes de structures métalliques : caractéristiques.

Description de l'oracle

L'objectif est de déterminer le minimum d'une fonction convexe non contrainte :

$$g(u) = \max_{i=1,\ldots,m} \left\{ \frac{v}{2} x^T A_i x) - f^T x \right\}.$$

Soit j la forme quadratique qui définit $g(u^c)$, la coupe d'optimalité est :

$$z \geq \omega^T u + \gamma,$$

où $\omega = vA_j u^c - f$ et $\gamma = g(u^c) - (vA_j u^c - f)u^c$.

9.2 Etude des différents paramètres de la méthode

La mise en œuvre de l'algorithme ACCPM nécessite de définir un certain nombre de paramètres. Etudions maintenant l'influence de ceux-ci sur le comportement de la méthode.

Les performances de l'algorithme sont caractérisées par :

- Le nombre d'itérations extérieures, c'est-à-dire le nombre d'appels à l'oracle.
- Le nombre d'itérations intérieures, c'est-à-dire la somme des itérations effectuées dans le module projectif (nombre de factorisations).
- Le temps total.

L'algorithme a été programmé en langage C (excepté la factorisation de Cholesky qui provient de LAPACK (Andersen et al. [1]) et qui est écrite en FORTRAN 77). Les tests ont été effectués sur une station de travail Power PC (IBM) type 7011, modèle 25T, de 64Mb de mémoire vive. Le programme a été compilé par les compilateurs AIX XL C++ et FORTRAN 77 avec le niveau d'optimisation -O (option par défaut).

Les problèmes ont été résolus avec une précision relative de 10^{-6} sur l'objectif.

9.2.1 Précision dans le calcul de la projection

Dans une toute première version écrite avec Matlab [18], nous avions utilisé la factorisation QR. Par la suite, afin de diminuer le temps de calcul, la factorisation de Cholesky a été choisie. Cette factorisation, bien que plus rapide ($1/3n^3$ flops[6] par rapport à $3n^3$ flops), entache la projection d'une erreur souvent importante. Elle rend alors nécessaire le calcul permettant de corriger la direction de Newton. Ce calcul nécessite deux fois la multiplication de $\tilde{G}$ par un vecteur et la résolution de deux systèmes. Soit m le nombre de colonnes introduites (c.à.d. $\kappa = 1 + 2n + m$), la correction coûte $4n(m+3) + 2n^2$ flops. Nous verrons par la suite qu'en pratique il faut en moyenne kn itérations à l'algorithme pour converger ($5 < k < 10$). Cette correction n'est donc pas très coûteuse puisque qu'elle est effectuée en $O(n^2)$ flops tandis que le calcul de la direction se fait en $O(n^3)$ opérations (le produit $\tilde{G}\tilde{X}\tilde{G}^T$: $2(m+3)n^2 \leq kn^3$, sa factorisation en $1/3n^3$ plus quelques opérations en $O(n^2)$). De plus, le calcul de cette correction n'est pas toujours nécessaire et, dans le but de limiter au maximum le temps de calcul, nous avons décidé de ne le faire que si l'erreur relative sur la direction de Newton par rapport à l'espace nul de $\tilde{G}\tilde{X}$ est supérieure à $\varepsilon(\overline{z} - \underline{z})$ où ε est une petite valeur fixée. Le test est le suivant : Si $\frac{\|\tilde{G}\tilde{X}\tilde{q}\|_\infty}{\|\tilde{q}\|} > \varepsilon(\overline{z} - \underline{z})$ alors on applique la correction.

Le tableau 9.5 indique, pour différentes valeurs de ε, le nombre d'itérations intérieures ainsi que le temps total, pour le problème de multiflots NDO148 version non linéaire et le problème stochastique 750 partagé en 75 blocs.

Ce tableau indique, tout d'abord, que la correction de la direction de Newton $\tilde{q}$ est indispensable. Lorsqu'elle n'est pas intégrée le nombre d'itérations intérieures (itérations consommées pour trouver le centre analytique approché avec comme critère d'arrêt $\|q\| < 1$) croît de manière démesurée. On déduit donc que cette correction doit être faite. En revanche, il n'est peut être pas nécessaire d'effectuer cette correction lorsque la précision de la solution courante est encore faible. Nous avons pris comme seuil de déclenchement de cette correction : $\frac{\|\tilde{G}\tilde{X}\tilde{q}\|_\infty}{\|\tilde{q}\|} > \varepsilon * (\overline{z} - \underline{z})$, avec $10^{-6} \leq \varepsilon \leq 10^{-4}$. Il est important de noter cependant que cette constatation n'a aucun fondement théorique. De plus, comme cette correction est peu coûteuse et que certains problèmes génèrent d'importantes instabilités numériques, il est souvent plus sûr, lorsque l'on a aucune idée sur la "difficulté" du problème, de faire cette correction toutes les itérations.

[6] Deux flops correspondent au travail nécessaire pour réaliser l'opération $x_i = x_i + a_{ik}b_{kj}$.

ε	Multiflots NDO148 non linéaire		Stochastique 750.75	
	Nombre d'itérations intérieures	Temps total	Nombre d'itérations intérieures	Temps total
$+\infty$	1613	17 mn 8.23 s	133	27.41 s
10^{+4}	1613	17 mn 9.01 s	idem	idem
10^{+2}	1613	17 mn 7.47 s	idem	idem
10^{-0}	1136	10 mn 45.76 s	133	27.41 s
10^{-2}	1016	9 mn 14.63 s	97	25.14 s
10^{-4}	1011	9 mn 13.18 s	60	22.97 s
10^{-6}	1004	9 mn 18.75 s	56	22.80 s
10^{-8}	idem	idem	56	22.81 s
10^{-10}	idem	idem	56	22.83 s
0	1004	9 mn 18.75 s	56	23.03 s

Tableau 9.5: Problème NDO148 non lin. & 750.75 : performances en fonction de ε.

9.2.2 Précision dans le calcul du centre analytique

Nous avons vu qu'il n'est pas nécessaire de calculer un centre analytique exact pour que l'algorithme fonctionne. L'objectif est de posséder un point u dual réalisable pour pouvoir l'envoyer à l'oracle. Cependant, l'atout de cet algorithme repose sur le fait qu'un point central prend en compte, de façon globale, l'information qu'il possède sur le problème. Un compromis doit donc être trouvé entre le calcul d'un centre analytique exact (très coûteux en nombre d'itérations intérieures et donc en temps de calcul) et le calcul d'un centre analytique approché avec pour seule exigence que le point u soit dual réalisable. Ce dernier, bien que relativement aisé à déterminer (peu d'itérations intérieures par itérations extérieures) risquerait d'entrainer plus d'itérations extérieures (et donc au total, plus d'itérations intérieures) pour déterminer l'optimum.

Si on se contente d'un centre analytique approché, le test d'arrêt dans les itérations de Newton n'est plus $\tilde{q} = 0$ mais simplement $\|\tilde{q}\|_\infty < 1$. Dans la pratique, nous avons observé que le test $\|\tilde{q}\| < \theta \leq 1$ est plus intéressant. En effet, bien qu'il soit plus exigeant et donc, bien qu'il induise plus d'itérations dans le module projectif, il permet de ne pas trop faire augmenter le nombre d'itérations extérieures. Les tableaux 9.6 et 9.7 indiquent, pour différentes valeurs de θ, le nombre d'itérations intérieures et extérieures effectuées ainsi que le temps total, pour le problème de multiflots NDO148 version non linéaire et le problème Badguy à 100 variables.

θ	Nombre d'itérations extérieures	Nombre d'itérations intérieures	Temps total
1.	851	1004	9 mn 17.20 s
.9	859	1031	9 mn 29.67 s
.8	866	1054	9 mn 47.11 s
.7	869	1082	9 mn 57.81 s
.6	872	1116	10 mn 7.14 s
.5	864	1145	10 mn 18.29 s
.4	859	1181	10 mn 34.50 s
.3	859	1267	11 mn 10.69 s
.2	854	1395	12 mn 10.77 s
.1	861	1708	15 mn 4.34 s
.01	865	1924	16 mn 18.82 s

Tableau 9.6: Problème NDO148 non linéaire : performances en fonction de θ.

Il apparaît clairement qu'il est inutile de prendre pour θ une valeur différente de 1. Chercher à avoir un meilleur centre analytique n'a pour effet que d'augmenter le nombre d'itérations intérieures et donc le temps total (rapport de 1 à 2 entre $\theta = 1.$ et $\theta = .01$ pour ces deux grandeurs). En effet, le nombre d'itérations extérieures reste à peu près invariant.

θ	Nombre d'itérations extérieures	Nombre d'itérations intérieures	Temps total
1.	800	1471	4 mn 21.13 s
.9	800	1566	4 mn 41.27 s
.8	800	1606	4 mn 46.08 s
.7	800	1705	5 mn 3.37 s
.6	800	1940	5 mn 56.89 s
.5	800	2001	6 mn 1.35 s
.4	800	2005	6 mn 1.70 s
.3	800	2007	6 mn 1.82 s
.2	800	2007	6 mn 1.78 s
.1	800	2009	6 mn 2.43 s
.01	800	2722	8 mn 37.44 s

Tableau 9.7: Problème Badguy à 100 variables : performances en fonction de θ.

9.2.3 Précision dans la recherche unidimensionnelle

Après avoir déterminé une direction dans la méthode de Newton il faut connaître le "pas" que l'on va faire dans cette direction. Pour cela on cherche, pour une direction $\tilde{d}$ fixée, le paramètre α pour lequel $\alpha\tilde{d}$ minimise la fonction (ici c'est la fonction potentielle). On fait alors une recherche unidimensionnelle. La question est de savoir avec quelle précision cette recherche doit être faite. Initialement nous avions envisagé d'appliquer la méthode de "dichotomie avec dérivées" pour déterminer ce minimum. Ensuite nous avons décidé d'utiliser l'algorithme de Newton. Le test d'arrêt porte sur la valeur de la dérivée au point courant. On continue la recherche tant que la valeur de la dérivée en α de la fonction potentielle est supérieure à un épsilon (ε) donné.

Les tableaux 9.8 et 9.9 indiquent, pour différentes valeurs de ε, le nombre d'itérations intérieures et extérieures effectuées ainsi que le temps total, pour le problème de multiflots NDO148 version non linéaire et le problème Badguy à 100 variables.

ε	Nombre d'itérations extérieures	Nombre d'itérations intérieures	Temps total
10^{-6}	862	1016	9 mn 30.86 s
10^{-4}	869	1023	9 mn 33.61 s
10^{-2}	871	1026	9 mn 34.61 s
10^{-1}	871	1026	9 mn 34.16 s
10^{-0}	870	1068	9 mn 58.15 s
10^{+1}	865	1856	15 mn 19.59 s

Tableau 9.8: Problème NDO148 non linéaire : performances en fonction de ε.

De ces résultats on peut conclure que la recherche unidimensionnelle semble indispensable. De plus, on remarquera qu'elle est très peu coûteuse en temps de calcul. On a donc intérêt à exiger la plus grande précision possible afin d'espérer faire le moins d'itérations intérieures.

En réalité, lorsque nous avons programmé l'algorithme nous avons utilisé un autre

ε	Nombre d'itérations extérieures	Nombre d'itérations intérieures	Temps total
10^{-6}	800	1471	4 mn 21.86 s
10^{-4}	800	1472	4 mn 21.89 s
10^{-2}	800	1473	4 mn 21.34 s
10^{-1}	800	1482	4 mn 22.48 s
10^{-0}	800	1698	5 mn 11.40 s
10^{+1}	805	7255	25 mn 29.18 s

Tableau 9.9: Problème Badguy à 100 variables : performances en fonction de ε.

critère d'arrêt. Il consiste à exiger que la valeur de la dérivée en fin de recherche soit 10^{-6} fois plus petite qu'au début (une variante de le règle de Wolfe et Powel [17]). Deux garde-fous ont cependant été introduits : on arrête toute recherche lorsque l'on a obtenu une dérivée plus petite que 10^{-6}, et on limite la recherche à une centaine d'itérations.

Enfin, si l'algorithme de Newton ne parvient pas à converger on applique la méthode de "dichotomie avec dérivées". Ce dernier point est indispensable pour la recherche unidimensionnelle lors de l'adjonction de colonnes puisque rien ne prouve que la fonction potientielle à minimiser est convexe en α. Cependant, pour tous les problèmes que nous avons traités cette méthode n'a jamais été activée.

9.2.4 Influence de la taille des boîtes

L'algorihme ACCPM nécessite que l'on localise la solution dans un espace. Cet espace est défini par les bornes l et h appelées boîtes. Celles-ci devront être d'autant plus grandes que l'on n'a pas idée de l'endroit où se trouve cette solution. Qu'elle est l'influence de la taille de ces boîtes sur le nombre d'itérations extérieures ?

La réponse à cette question n'est pas évidente. Pour certains types de problèmes, la méthode ACCPM est peu sensible à la taille des boîtes, pour d'autres elle y est beaucoup plus. Par exemple, nous avons montré [2] que pour les problèmes de programmation géométrique, l'algorithme ACCPM n'est pratiquement pas sensible à la taille des boîtes.

On peut faire la même constatation avec le problème Badguy à 100 variables. Le nombre d'itérations extérieures est de 800 tant que les boîtes restent de taille “raisonnables”. Par contre, si on continue à augmenter les boîtes, le nombre d'itérations intérieures croît rapidement (voir tableau 9.10) et les performances de l'algorithme se dégradent.

Dimension des boîtes	Nombre d'itérations extérieures	Nombre d'itérations intérieures	Temps total
10^{+4}	800	1105	3 mn 54.02 s
10^{+5}	800	1343	4 mn 12.96 s
10^{+6}	800	1523	4 mn 26.85 s
10^{+7}	831	1727	4 mn 53.73 s
10^{+8}	898	1935	5 mn 37.83 s
10^{+9}	876	2120	5 mn 44.44 s

Tableau 9.10: Problème Badguy à 100 variables : performances en fonction de la taille des boîtes.

On constatera de même que ACCPM reste aussi relativement insensible à la dimension des boîtes pour le problème de multiflot NDO148 version non linéaire (voir tableau 9.11).

Par contre, pour le problème de structure métallique à 162 variables (T2), les performances se dégradent très rapidement lorsque la taille des boîtes augmente (voir tableau 9.12).

Heureusement, il est possible de stabiliser la méthode ACCPM en prenant initialement de petites boîtes et en incorporant à l'algorithme le mécanisme permettant de les augmenter (voir § 7.3). Il faut savoir cependant que toute augmentation perturbe l'ensemble de localisation de façon importante et que cela se traduit par une croissance du nombre d'itérations intérieures par rapport au nombre d'itérations extérieures.

En conclusion la meilleure politique consiste à prendre des boîtes “raisonnables” (souvent 10^2 ou 10^3) et à incorporer le mécanisme permettant exceptionellement de les déplacer.

Dimension des boîtes	Nombre d'itérations extérieures	Nombre d'itérations intérieures	Temps total
10^{+1}	829	946	8 mn 51.92 s
10^{+2}	848	991	9 mn 17.66 s
10^{+3}	851	1004	9 mn 18.91 s
10^{+4}	875	1048	9 mn 45.94 s
10^{+5}	895	1079	10 mn 12.28 s
10^{+6}	903	1105	10 mn 32.39 s

Tableau 9.11: Problème NDO148 non linéaire : performances en fonction de la taille des boîtes.

Dimension des boîtes	Nombre d'itérations extérieures	Nombre d'itérations intérieures	Temps total
10^{+2}	408	595	8 mn 15.41 s
10^{+3}	694	933	17 mn 23.74 s
10^{+4}	942	1252	28 mn 34.55 s
10^{+5}	1229	1599	44 mn 15.39 s
10^{+6}	1558	2104	1 h 25 mn 27.20 s
10^{+7}	1877	2447	2 h 27 mn 55.85 s

Tableau 9.12: Problème T2 : performances en fonction de la taille des boîtes.

9.2.5 Réduction du nombre de coupes

Le test permettant d'éliminer des coupes (où de manière équivalente, des colonnes de $\tilde{G}$) est relativement délicat à mettre en œuvre. D'une part il est très coûteux, d'autre part il est difficile de le rendre à la fois sûr et efficace.

Il est coûteux parce qu'il nécessite pour chaque colonne la résolution d'un système et qu'une fois les colonnes éliminées, il faut rechercher un nouveau point réalisable. Soit m le nombre de colonnes introduites, le test lui même est en $O(mn^2)$ avec $m > n$ et la recherche d'un point réalisable (nécessitant le calcul d'une projection ainsi que sa correction) en $O(n^3)$. Le gain principal que l'on obtient est observé lorsque l'on fait le produit $\tilde{G}\tilde{X}^2\tilde{G}^T$. Ce produit nécessite $2mn^2$ flops. soit $\delta(m)$ le nombre de colonnes éliminées le gain est alors de $\delta 2mn^2$ flops par produit. Il faudra donc faire au moins β (avec $\beta = \frac{n+m}{\delta(m)}$) produits $\tilde{G}\tilde{X}^2\tilde{G}^T$ avant la fin de l'algorithme pour que le test soit rentable. Comme, pour les problèmes intéressants, n est grand et comme d'autre part l'algorithme ne nécessite pas un nombre de produits $\tilde{G}\tilde{X}^2\tilde{G}^T$ très élevé, il apparaît clairement que le nombre de colonnes éliminées doit être grand (avoisiner la dimension n).

De cette remarque on déduit tout d'abord qu'il est nécessaire de réduire l'ellipsoïde extérieur d'un facteur homothétique (c.à.d. d'éliminer aussi les coupes qui l'intersecte). Ce mécanisme est efficace mais dangereux. On peut être amené à éliminer trop de coupes. Il est même possible que, perdant à chaque itération trop d'informations, l'algorithme ne parvienne plus à converger.

La solution que nous avons adoptée consiste à trier les contraintes par ordre de distance au centre analytique décroissant (distance calculée dans la métrique de l'ellipsoïde) et à les supprimer jusqu'a ce qu'il n'en reste que kn (avec $1,5 < k < 2,5$ en fonction du problème). Cette méthode semble relativement robuste.

On remaquera aussi que si on veut pouvoir éliminer n plans de support il n'est pas envisageable de faire le test à toutes les itérations. Nous avons alors décidé de ne le faire que toutes les kn itérations (avec $2 < k < 3$ en fonction du problème).

Malheureusement il n'est pas possible de certifier que ce test sera rentable. Cela dépend grandement de la nature du problème. Si on étudie les performances du problème NDO148 non linéaire, on constate que d'appliquer le test toutes les $2n$

itérations[7] en conservant au moins $2n$ variables donne le meilleur résultat (voir tableau 9.13).

		Nombre d'itérations extérieures	Nombre d'itérations intérieures	Temps total
μ	1.	846	988	9 mn 7.67 s
$\hbar = 3n$	.5	872	1034	7 mn 40.19 s
$m_{min} = 2n$	0.	872	1034	7 mn 40.19 s
$\mu = 0$	$2n$	871	1020	7 mn 33.23 s
$\hbar$	$3n$	872	1034	7 mn 40.19 s
$m_{min} = 2n$	$4n$	874	1022	7 mn 56.45 s
	∞	846	991	9 mn 7.67 s
$\mu = 0$	$1.5n$	988	1179	8 mn 12.02 s
$\hbar = 3n$	$2n$	872	1034	7 mn 40.19 s
m_{min}	$2.5n$	848	989	7 mn 47.22 s

Tableau 9.13: Problème NDO148 non linéaire : performances avec le test d'élimination (réduction de l'ellipsoïde extérieur : $\rho = \mu - (1-\mu)\rho_{max}$; fréquence : $\hbar$; colonnes conservées : m_{min}).

On remarquera que le gain est relativement faible (dans la pratique on observe pas plus de 10 à 30 % de gain en fonction du problème), surtout si on le compare à la complication de l'algorithme et aux instabilités qu'il implique. De plus, dans le cas où l'oracle prend la plupart du temps de calcul (comme dans la majeure partie des problèmes stochastiques testés), ce gain devient négatif. En effet, ce test induit souvent plus d'itérations extérieures (donc plus d'appels à l'oracle) et n'apporte en échange qu'une diminution négligeable du temps de calcul.

[7]Effectuer le test toutes les $3n$ itérations est habituellement meilleur. Pour ce problème, lors du premier test seules 2 contraintes ont été supprimées. Le test a donc pratiquement été inefficace. Si cependant le temps total est plus faible cela provient simplement du fait que la perturbation induite par cette suppression provoque moins d'itérations intérieures.

9.3 Résultats numériques

Les problèmes ont été résolus avec une précision relative de 10^{-6} sur l'objectif. Avant chaque problème la taille des boîtes utilisées est précisée. Les résultats donnés correspondent aux meilleurs réglages possibles des paramètres de l'algorithme eu égard au temps de calcul. Lorsque le test d'élimination permet de diminuer le temps de calcul de manière significative, le nombre de colonnes éliminées ainsi que la valeur des paramètres du test sont donnés.

9.3.1 Problèmes de programmation géométrique

Pour résoudre ces problèmes les boîtes $l = -9$ et $h = +9$ ont été utilisées, excepté pour le problème 10 où, pour des raisons d'instabilités numériques, les boîtes ont été mises à $l = -30$ et $h = +30$. Le test d'élimination a permis de diminuer le temps de calcul total. Il a été appliqué lorsque $3n$ plans ont été introduits et on s'est garanti de garder au moins $2n$ coupes. Les résultats sont donnés dans le tableau 9.14.

Problème	Nombre d'itérations extérieures	Nombre d'itérations intérieures	Nombre de colonnes éliminées	Temps total
1	13	31	0	0.04 s
3	28	49	14	0.09 s
2	33	55	18	0.11 s
4	37	62	23	0.12 s
6	33	56	18	0.11 s
8	54	109	58	0.13 s
9a	64	106	44	0.29 s
9b	64	109	42	0.30 s
9c	67	112	51	0.31 s
7a	67	115	39	0.34 s
7b	68	113	40	0.33 s
10	84	192	58	0.56 s
5	67	160	62	0.56 s
11	100	169	73	1.59 s
12	183	312	317	4.75 s
12b	170	257	194	4.40 s

Tableau 9.14: Problèmes de programmation géométrique : performances.

On remarquera qu'il faut en moyenne, pour ce type de problème, $7.8n$ itérations pour déterminer l'optimum avec une précision de 10^{-6}. De plus, il faut en moyenne 1.82 itérations dans le module projectif pour trouver le nouveau centre analytique approché.

9.3.2 Problèmes de multiflots

Pour résoudre ces problèmes les boîtes $l = 0$ et $h = 10^3$ ont été utilisées. Le test d'élimination a permis de diminuer le temps de calcul total. Il a été appliqué lorsque

$3n$ plans ont été introduits et on s'est garanti de garder au moins $2n$ coupes.

Problème	Nombre d'itérations extérieures	Nombre d'itérations intérieures	Nombre de colonnes éliminées	Temps total
NDO22 lin.	78	113	25	0.53 s
NDO22 non lin.	145	194	94	1.25 s
NDO148 lin.	473	572	151	3 mn 46.11 s
NDO148 non lin.	887	1050	449	7 mn 38.64 s

Tableau 9.15: Problèmes de multiflots : performances.

On remarquera qu'il faut en moyenne $3.37n$ ($6.29n$) itérations pour déterminer l'optimum avec une précision de 10^{-6} dans le cas linéaire (non linéaire). De plus, il faut en moyenne 1.84 (respectivement 1.26) itérations dans le module projectif pour trouver le nouveau centre analytique approché dans le cas linéaire (ou non linéaire).

9.3.3 Problèmes linéaires stochastiques

Pour résoudre ces problèmes les boîtes $l = 0$ et $h = 10^2$ ont été utilisées. De plus, nous avons inclu le test vérifiant avant d'inclure une contrainte dans $\tilde{G}^T$ qu'elle n'y avait pas déjà été introduite. Les résultats sont donnés dans le tableau 9.16.

On remarquera le faible nombre d'itérations extérieures par rapport à la dimension du problème (dim. de x_0). Il faut réaliser cependant qu'à chaque appel à l'oracle, m' (m' = card de I) plans de support sont intégrés (excepté si certains d'entre eux sont éliminés). On remarquera aussi que plus le nombre m' est important plus le rapport " nombre d'itérations intérieures sur nombre d'itérations extérieures" croît. La pertubation entrainée par l'ajout de plus d'un plan de support rend plus difficile la recherche du nouveau centre de analytique. Enfin, discrétiser à l'extrême l'objectif (par exemple le problème 750.750) n'est pas nécessairement rentable. En effet, la taille de la matrice $\tilde{G}$ commence à devenir importante non seulement en nombre de lignes mais aussi en nombre de colonnes, puisqu'à chaque itération, 750 colonnes sont ajoutées. Ainsi, même si l'on s'efforce de gérer correctement les contraintes de convexité, le gain réalisé en itérations extérieures ne parvient pas à surpasser la perte, en temps de calcul, dans le programme-maître. La figure (9.1) illustre ces diverses remarques.

Problème	Nombre d'itérations extérieures	Nombre d'itérations intérieures	Nombre de colonnes éliminées	Temps total
App0110	45	87	17	1.43 s
App0112	34	72	82	0.56 s
App0210	549	723	0	44 mn 18.88 s
App0212	54	104	81	39.58 s
750.1	50	93	29	8 mn 48.44 s
750.2	37	74	39	3 mn 48.61 s
750.10	22	53	98	44.58 s
750.50	15	53	290	13.24 s
750.75	13	52	355	10.99 s
750.125	13	58	652	11.74 s
750.250	12	59	1126	15.48 s
750.750	13	79	4232	1 mn 15.19 s

Tableau 9.16: Problèmes linéaires stochastiques : performances.

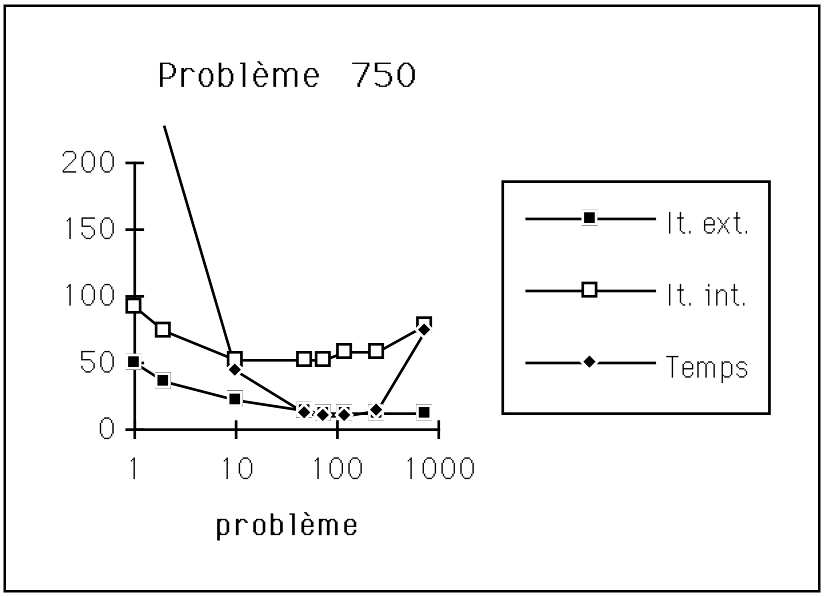

Figure 9.1: Comportement de l'algorithme en fonction de la discrétisation du problème 750.

9.3.4 Problèmes Badguy

Pour résoudre ces problèmes les boîtes $l = -10^2$ et $h = 10^2$ ont été utilisées. Le test d'élimination a permis de diminuer le temps de calcul total. Il a été appliqué lorsque $3n$ plans ont été introduits et on s'est garanti de garder au moins $2n$ coupes.

Dimension	Nombre d'itérations extérieures	Nombre d'itérations intérieures	Nombre de colonnes éliminées	Temps total
10	80	148	57	0.45 s
20	160	295	107	1.67 s
30	240	443	157	4.68 s
40	320	589	207	10.70 s
50	400	737	257	21.29 s
60	480	884	307	36.75 s
70	560	1031	357	59.76 s
80	640	1177	407	1 mn 32.62 s
90	720	1325	457	2 mn 17.14 s
100	800	1470	507	3 mn 17.37 s
110	880	1618	557	4 mn 38.08 s
120	960	1767	607	6 mn 16.89 s
130	1040	1913	657	8 mn 22.31 s
140	1120	2059	707	10 mn 50.10 s
150	1200	2207	757	13 mn 56.98 s
160	1280	2353	807	17 mn 38.35 s
170	1360	2499	857	22 mn 16.64 s
180	1440	2647	907	27 mn 3.40 s
190	1520	2793	957	33 mn 11.27 s
200	1600	2940	1007	39 mn 50.85 s

Tableau 9.17: Problèmes Badguy : performances.

On remarquera que le nombre d'itérations pour ce type de problème est toujours de $8n$. La figure (9.2) indique l'évolution du temps de calcul total en fonction de la dimension du problème.

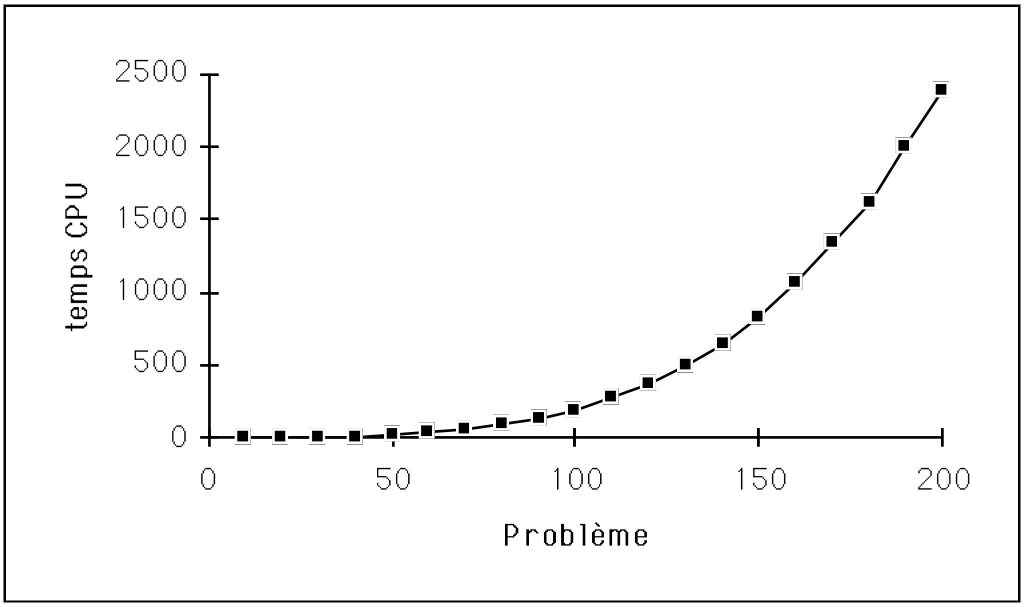

Figure 9.2: Temps de calcul pour les problèmes badguy.

9.3.5 Problèmes Maxquad

Pour résoudre ces problèmes les boîtes $l = -10^2$ et $h = 10^2$ ont été utilisées. Le test d'élimination a permis de diminuer le temps de calcul total. Il a été appliqué lorsque $3n$ plans ont été introduits et on s'est garanti de garder au moins $2n$ coupes. Les résultats sont donnés dans le tableau 9.18.

Paramètre epsreg	Nombre d'itérations extérieures	Nombre d'itérations intérieures	Nombre de colonnes éliminées	Temps total
1	149	195	123	0.45 s
10^{-1}	159	210	134	0.47 s
10^{-2}	158	207	134	0.47 s
10^{-3}	158	208	134	0.48 s
10^{-5}	156	206	134	0.48 s
10^{-10}	153	203	123	0.50 s

Tableau 9.18: Problèmes Maxquad : performances.

9.3.6 Problèmes de dimensionnement optimal de structures métalliques

Pour résoudre ces problèmes les boîtes $l = -10$ et $h = 10$ ont été utilisées. Pour être certain d'obtenir l'optimum nous avons incorporé l'utilitaire permettant d'augmenter et de diminuer les boîtes. En procédant de cette manière l'algorithme converge plus rapidement que si nous avions pris directement les boîtes $l = -10^3$ et $h = 10^3$. Ceci s'explique par le fait que la force s'applique en un point unique à une extrémité de la structure tandis que l'autre extrémité est fixe. Comme la variable sur laquelle on optimise mesure le déplacement des nœuds, même si celui-ci est relativement important près du point où la force est appliquée, il s'estompe rapidement dès que l'on s'en éloigne. Pour la majeure partie des nœuds, le déplacement est donc faible et des boîtes à $l = -10$ et $h = 10$ sont amplement suffisantes. Les performances de l'algorithme se trouvent dans le tableau 9.19.

Problème	Nombre d'itérations extérieures	Nombre d'itérations intérieures	Temps total
T3	115	228	30.47 s
T2	301	683	7 mn 55.07 s
T1	375	918	1 h 9 mn 46.19 s

Tableau 9.19: Problèmes de structures métalliques : performances.

On remarquera le très faible nombre d'itérations extérieures nécessaire pour déterminer l'optimum (à peine supérieur au nombre de variables pour les problèmes T1 et T3).

9.4 Comparaison avec l'algorithme de Dantzig-Wolfe

L'algorithme de Dantzig-Wolfe ne diffère de ACCPM que dans la proposition u qu'il envoie à l'oracle. Nous avons vu qu'il détermine le minimum de la relaxation linéaire tandis que l'algorithme ACCPM envoie un centre analytique approché.

La critique qui est faite à la méthode de Dantzig-Wolfe est son aspect oscillatoire. Les propositions faites à l'oracle ayant tendance à être très différentes les unes des autres l'algorithme convergerait lentement.

Cette affirmation est cependant à nuancer comme le démontre les expériences sur la programmation linéaire stochastique rapporté dans [2] .

Par exemple, en utilisant le générateur de problèmes stochastiques proposé par Mehdi [16], nous avons construit 5 problèmes (voir tableau 9.20) dont les performances sont à peine moins bonnes avec l'algorithme de Dantzig-Wolfe qu'avec l'algorithme ACCPM (voir tableau 9.21).

Problem	Taille	dim. de x_0	card. de I	dimension de B_i^T	densité
m1	3020×8000	20	100	30×80	2%
m2	4040×8000	40	100	40×80	2%
m3	5060×8000	60	100	50×80	2%
m4	6090×8000	90	100	60×80	2%
m5	6153×8000	153	100	60×80	3%

Tableau 9.20: Problèmes de Mehdi : caractéristiques.

Problème	ACCPM		Dantzig-Wolfe	
	nb iter. extérieures	temps total	nb iter. extérieures	temps total
m1	16	1 mn 47.0 s	15	1 mn 37.1 s
m2	19	3 mn 14.8 s	20	3 mn 15.9 s
m3	20	4 mn 26.1 s	22	5 mn 17.7 s
m4	21	7 mn 31.6 s	23	9 mn 57.7 s
m5	28	17 mn 40.7 s	?	?

Tableau 9.21: Problèmes de Mehdi : performances.

On notera cependant que le rapport est toujours en notre faveur et qu'il a tendance à l'être d'autant plus que la dimension de x_0 (variable de première décision) est importante.

On aboutit aux mêmes conclusions avec les problèmes stochastiques App0× (voir tableau 9.22).

Problème	ACCPM		Dantzig-Wolfe	
	nb iter. extèrieures	temps total	nb iter. extèrieures	temps total
app0110	36	2.36 s	55	4.56 s
app0112	12	0.40 s	2	0.23 s
app0210	539	1 h 56 mn	2544	36 h 7 mn
app0212	23	37.14 s	6	28.36 s

Tableau 9.22: Problèmes stochastiques App0× : performances.

Lorsqu'un seul plan de support est généré, l'algorithme ACCPM est hautement plus performant que la méthode de Dantzig-Wolfe. Par exemple, nous avons cherché, sans succès, à résoudre le problème de multiflots à 148 variables version non linéaire par l'algorithme de Dantzig-Wolfe, tandis que seules 7mn 38.64s ont été nécessaires avec ACCPM.

Par contre pour le même problème, en utilisant une version désagrégée de l'oracle [11], Dantzig-Wolfe converge plus rapidement que ACCPM. La raison est simple. Lorsqu'un grand nombre de coupes est envoyé à chaque itération, un algorithme qui consiste à chercher le minimum de la relaxation linéaire (Dantzig-Wolfe) converge plus vite qu'un algorithme qui va proposer un point central. En effet, quand toute l'information est présente (ce qui peut survenir rapidement) le premier va déterminer immédiatement l'optimum. L'algorithme ACCPM, par contre, continue à faire plusieurs itérations. A chacune d'elles il envoie une proposition et ne reçoit qu'une information relative à la variation de la borne supérieure (toutes les coupes étant présentes). Il peut alors mettre un grand nombre d'itérations à converger vers le point que Dantzig-Wolfe a immédiatement déterminé. Une solution consiste à augmenter de manière importante le poids mis sur l'objectif par le calcul d'un centre analytique pondéré (voir § 5.8). Nous avons vu que si μ_0 est le poids mis sur l'objectif et si le poids mis sur les autres coupes est égale à l'unité, alors le saut de dualité s'écrit : $x^T s = \frac{\kappa s_0}{\mu_0}$. Il est donc d'autant plus petit que μ_0 est grand et l'algorithme va probablement converger plus rapidement. Cette solution a été testée avec succès pour les problèmes app0×. Nous avons constaté qu'il est possible, en augmentant le poids mis sur l'objectif, d'obtenir un nombre d'itérations comparable à celui obtenu par Dantzig-Wolfe. Cependant, nous avons aussi constaté que le calcul du centre analytique approché nécessite alors plus d'itérations intérieures. Bien que ce soit conforme à l'étude de la complexité (voir page 113), ceci implique que, pour les problèmes où Dantzig-Wolfe se comporte moins bien que l'algorithme ACCPM, l'adjonction de poids dans l'algorithme ne fait que d'augmenter le temps de calcul.

Pour conclure, observons que lorsque le problème est "simple" (c'est-à-dire s'il est de petite taille ou si la solution est triviale), l'algorithme de Dantzig-Wolfe donne de très bons résultats. Dans les autres cas par contre, ACCPM converge nettement plus vite et permet parfois d'obtenir un résultat que Dantzig-Wolfe n'aurait pas pu déterminer (au moins en un temps raisonnable). Remarquons de plus que l'algorithme ACCPM, dans sa version de base, converge toujours et que son comportement est relativement prévisible.

Bibliographie

[1] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov and D. Sorensen, *LAPACK Users' Guide*, SIAM, Philadelphia, **1992**.

[2] O. Bahn, O. du Merle, J. L. Goffin et J. P. Vial, "A cutting plane method from analytic centers for stochastic programming", à paraître dans *Mathematical programming.*

[3] J. F. Benders, "Partitioning procedures for solving mixed-variables programming problems", *Numerische Mathematik* **4** (1962) 238-252.

[4] A. Ben-Tal et M. P. Bensøe, "E new method for optimal truss design. Tech.", *Techical report*, Technion, Israel Inst. Of Technologiy, 1991.

[5] G.B. Dantzig, "Linear programming and extensions", (1963), Princeton University Press.

[6] G. B. Dantzig et P. Wolfe, "The decomposition algorithm for linear programming", *Econometrica* **29** (1961) 767-778.

[7] Y. Ermoliev, R. J-B Wets, *Numerical techniques for stochastic optimization*, Springer-Verlag, New York (1988).

[8] L. Escudero, P.K. Kamesam, A.J. King et R.J.-B. Wets, "Production Planning via Scenario Modelling", *Annals of Operations Research* **43** (1993) 311-355.

[9] E. Fragnière et A. Haurie, "MARKAL-Geneva : A Model to assess Energy-Environnement Choices for a swiss Canton", miméo, Genève (1994).

[10] E. M. Gafni et D.P. Bertsekas, "Two-metric projection methods for constrained optimization", *SIAM Journal on Control and Optimization* 22, **6** (1984) 936-964.

[11] J. L. Goffin, J. Gondzio, R. Sarkissian et J. P. Vial, “Solving nonlinear multicommodity flow problems by the analytic center cutting plane method”, Université de Genève, Suisse (1994).

[12] J. G. Kallberg, R. W. White et W. T. Ziemba, “Short term financial planning under uncertainty”, *Management Science* 6, **28** (1982) 670-682.

[13] C. Lemaréchal, *Non-smooth optimization*, Pergamon Press, Oxford, Great Britain (1978).

[14] M. Kŏcvara et J. Zowe, “Codes for truss topology design : a numerical comparison”, *Techical report*, 1991.

[15] K. O. Kortanek et Hoon No, “A second Order affine scaling algorithm for the geometric programming dual with logarithmic barrier”, *Optimization* **23** (1992) 303-322.

[16] D. Medhi, ”Parallel bundle-based decomposition for large-scale structured mathematical programming problems”, *Annals of Operations Research* **22** 1990 101-127.

[17] M. Minoux, *Programmation mathématique : théorie et algorithmes*, **1** et **2** Dunod (Bordas), Paris (1983).

[18] C. B. Moler, J. Little, S. Bangert et S. Kleiman, *Pro-MatlabUser's Guide*, (MathWorks, Sherborn, M.A., 1987).

Conclusion

De nombreuses applications, dans des domaines aussi divers que l'Economie, les Mathématiques ou les Sciences de l'ingénieur, conduisent à la recherche du minimum d'une fonction convexe. L'optimisation convexe non-différentiable offre un cadre mathématique permettant de résoudre ces problèmes.

Si des méthodes efficaces ont été trouvées dans le cas où la fonction est différentiable, des difficultés apparaisent lorsque l'on introduit des contraintes ou lorsque la fonction n'est pas continûment différentiable. Cela est dû à deux raisons : la direction de recherche n'est plus nécessairement une direction de descente ; le critère d'arrêt n'est plus valable.

Pour contourner ces difficultés de nombreuses méthodes ont été développées. Cette thèse fait un rapide tour d'horizon de ces différents algorithmes en se consacrant sur trois méthodes principales : la méthode des sous-gradients, la méthode des ellipsoïdes et les méthodes de plans coupants. S'il est possible de montrer que les meilleures variantes du deuxième chapitre (méthode des sous-gradients et méthode des ellipsoïdes) possèdent une complexité théorique optimale, dans la pratique, les algorithmes du troisième chapitre (méthode de plans coupants) se comportent mieux.

On peut regrouper certaines méthodes de plans coupants en une famille appelée méthode des centres. Les méthodes appartenant à cette famille présentent deux particularités : elles définissent un ensemble contenant nécessairement une solution optimale et elles proposent comme solution, un point appartenant à l'intérieur de cet ensemble. Elles procèdent alors itérativement en générant, à chaque itération, un point dit "central" et en recevant, en retour, une coupe leur indiquant comment réduire l'ensemble de localisation. L'objectif, en proposant un point "central", est de faire décroitre au maximum le volume de l'ensemble contenant la solution optimale.

L'algorithme ACCPM (Analytic Center Cutting Plan Method) proposé par Goffin, Haurie et Vial appartient à la famille des méthodes des centres. Il découle des récentes recherches sur les méthodes des points intérieurs. Le point central est appelé : centre analytique. Il est défini comme le maximum de la somme des logarithmes des variables d'écart associées aux différentes coupes.

L'objet de cette thèse a été de mettre en œuvre l'algorithme ACCPM. Nous avons vu tout au long de ce document que de nombreuses techniques ont été élaborées pour que, d'un concept théorique, puisse naître une méthode utilisable dans la pratique. Après avoir décrit les fondements théoriques (chapitre 4 et 5) nous avons donné les procédures essentielles sans lesquelles l'algorithme ne peut fonctionner (chapitre 6). Nous y trouvons, entre autre, comment calculer une borne inférieure, comment ajouter une coupe et comment limiter les instabilités numériques. Dans le chapitre 7 nous avons décrit un ensemble de techniques permettant d'améliorer les performances de l'algorithme et plus particulièrement comment gérer de manière efficace l'introduction d'un ensemble de coupes. Nous y décrivons aussi comment éliminer des coupes et comment initialiser l'algorithme (Phase I). Dans le chapitre 8, enfin, nous donnons les principales procédures à mettre en œuvre. Ce chapitre permet non seulement d'avoir un aperçu global de la méthode mais il devrait être utile à toute personne qui désirerait programmer l'algorithme ACCPM. Pour finir, dans un dernier chapitre nous décrivons les différents problèmes sur lesquel nous avons testé l'algorithme en indiquant les performances et le comportemant de la méthode en fonction des paramètres choisis (comme par exemple la qualité du centrage).

Au vu des résultats numériques, on peut conclure que les développements que nous avons apportés à l'algorithme ACCPM en font une méthode fiable, robuste et performante pour l'optimisation convexe non différentiable. Elle est fiable, puisqu'elle permet de déterminer une solution satisfaisant parfaitement les contraintes avec une évaluation précise de sa qualité (saut de dualité connu). Elle est robuste, puisqu'elle a toujours permis de déterminer l'optimum, et cela quelque soit la nature du problème traité. Enfin, elle performante puisqu'elle se compare de manière relativement favorable aux autres méthodes développées. On notera aussi son caractère relativement prévisible qui nous permet de pouvoir estimer le nombre d'itérations qui seront nécessaires pour déterminer l'optimum, en fonction de la dimension du problème à résoudre.

Nous pensons avoir maintenant en main, un outil nous permettant d'aborder des problèmes de très grande taille comme ceux générés par la programmation stochastique. Le chemin est balisé : il suffit de gérer avec soin la structure de données et les comunications entre le programme-maître et l'oracle. Un développement particulièrement prometteur concerne la mise en œuvre sur machines parallèles. La décomposition s'y prête naturellement mais le parallélisme pourrait aussi porter sur les opérations matricielles du programme-maître.