RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET BPOPULAIRE

MINISTÈRE DE L'ENSIEGNEMENT SUPÉRIEUR ET DE RECHERCHE SCIENTIFIQUE

UNIVERSITÉ FERHAT ABBAS DE SÉTIF

FACULTÉ DES SCIENCES

DÉPARTEMENT DE MATHÉMATIQUES

MÉMOIRE

Présenté par

Amel NOUI

Pour l'obtention du diplôme de

Magister en Mathématiques

Option **: Mathématiques Appliquées**

THÈME

ÉTUDE NUMÉRIQUE D'UNE MÉTHODE PROJECTIVE POUR UN PROGRAMME CONVEXE

Soutenu le :24/05/2012…….....
à **l'UNIVERSITÉ DE SÉTIF**
devant le jury composé de :

Président	**: Pr. BENTERKI Djamel**	**Université Ferhat Abbas Sétif**
Rapporteur	**: MC. MERIKHI Bachir**	**Université Ferhat Abbas Sétif**
Examinateur	**: MC. ZITOUNI Rachid**	**Université Ferhat Abbas Sétif**
Examinateur	**: MC. KEBBICHE Zakia**	**Université Ferhat Abbas Sétif**

Table des matières

Introduction

Depuis longtemps, les spécialistes de la programmation mathématique œuvrent pour parvenir à résoudre des problèmes à grandes dimensions liés à des aspects pratiques intéressants.

Les développements de la programmation linéaire au lendemain de la seconde guerre mondiale a donné satisfaction aux utilisateurs dans différents domaines d'application. Un tel succès est du en grande partie à la méthode du simplexe et ses variantes. On a ainsi élaboré des méthodes dites de linéarisation pour résoudre des programmes non linéaires en faisant appel successivement à des sous programmes linéaires.

Cette approche présente malheureusement des défaillances théoriques mais surtout numériques : problème du chemin optimal, dégénérescence et cyclage. Ceci a entraîné des réticences de la part des utilisateurs mais aussi des spécialistes ; plusieurs questions soulevées sont restées ouvertes. . .

En 1984, Karmarkar publie un algorithme révolutionnaire pour la programmation linéaire, en très peu de temps on a développé des variantes efficaces pour plusieurs classes de programmation convexe voir non convexe.

Notre étude reprend le principe de linéarisation avec les ingrédients de l'approche de Karmarkar pour minimiser une fonction convexe différentiable sur un polyèdre convexe : le programme à résoudre est remplacé par une suite de sous programmes linéaires approximants. L'énorme avantage ici est que l'on connait explicitement la solution optimale de chaque sous programme linéaire, contrairement aux méthodes simpliciales où l'on doit chercher cette solution, ce qui nécessite un volume calculatoire dominant le coût de l'ité-

ration. De plus, il n'y a plus de problème de cheminement ni de dégénérescence dans notre approche.

L'algorithme ainsi élaboré est à complexité arithmétique polynomiale et beaucoup moins complexe.

Nous avons réalisé des expérimentations numériques encourageantes mettant en évidence les performances de notre algorithme, notamment la stabilité et la robustesse qu'on peut constater à travers les problèmes liés à la théorie de l'information.

Le travail réalisé est présenté en trois chapitres :

Le premier est consacré aux rappels fondamentaux de la convexité, la programmation mathématique et les méthodes de points intérieurs récentes.

Le deuxième comprend la description de notre algorithme.

Le dernier chapitre présente les résultats numériques issus de la mise en œuvre de notre algorithme.

Chapitre 1

Rappels d'analyse convexe, optimisation et méthodes de points intérieurs

1.1 Introduction

Dans ce chapitre nous rappelons brièvement quelques notions importantes qui serviront pour le traitement de notre travail. Des développements sur le sujet peuvent être trouvés dans la monographie A. Keraghel [11], M. Minoux [14] et [17].

1.2 Convexité

1.2.1 Définitions générales et propriétés élémentaires

Dans cette partie, nous rappelons les plus importants résultats d'analyse convexe nécessaires pour le développement de ce travail.

Définition 1 *Un sous-ensemble C de $\mathbb{R}^n$ est dit convexe si pour tout x et y appartenant à C, le segment $[x, y]$ est contenu dans C :*

$$\forall \lambda \in [0,1]\,,\ \forall x, y \in C : (1-\lambda)x + \lambda y \in C.$$

Définition 2 *Soit C un sous-ensemble convexe de $\mathbb{R}^n$ et $f : C \longrightarrow \mathbb{R}$ une fonction. On dit que f est **convexe** dans C si :*

$$\forall \lambda \in [0,1]\,,\ \forall x, y \in C :\ f\left[(1-\lambda)x + \lambda y\right] \leq (1-\lambda)f(x) + \lambda f(y).$$

f est dite strictement convexe si l'inégalité ci-dessus est toujours stricte

$$\forall x \neq y, \forall \lambda \in \,]0,1[\,.$$

Définition 3 *Un sous-ensemble C de $\mathbb{R}^n$ est dit affine si*

$$\forall \lambda \in \mathbb{R},\ \forall x, y \in C : (1-\lambda)x + \lambda y \in C.$$

Définition 4 *Une fonction $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ est dite affine sur C si :*

$$\forall \lambda \in \mathbb{R},\ \forall x, y \in C :\ f\left[(1-\lambda)x + \lambda y\right] \leq (1-\lambda)f(x) + \lambda f(y).$$

Définition 5 *un ensemble convexe de la forme*

$$P = \left\{x \in \mathbb{R}^n : Ax = b\right\} /\ A \in \mathbb{R}^{m \times n},\ b \in \mathbb{R}^m$$

est appelé polyèdre convexe.

Définition 6 *un ensemble convexe de la forme*

$$S_n = \left\{ x \in \mathbb{R}^n : \sum_{i=1}^{n} x_i = 1 \right\}$$

est appelé n-simplexe.

1.3 Convexité et Optimisation

1.3.1 Notation

Un programme mathématique est un problème d'optimisation de la forme :

$$(PM) \begin{cases} \min f(x) \\ g_i(x) \leq 0, \quad i = 1 : m \\ h_j(x) = 0, \quad j = 1 : p \\ \qquad x \in C \subseteq \mathbb{R}^n \end{cases}$$

où $f; g_i; h_j$ sont des fonctions définies de $\mathbb{R}^n$ dans $\mathbb{R}$.

Définition 7 *L'ensemble*

$$S = \{x \in C : g_i(x) \leq 0,\ h_j(x) = 0,\ i = 1 : m,\ j = 1 : p\}$$

est appelé ensemble des solutions réalisables.

Définition 8 *Une solution réalisable qui minimise l'objectif sur C est dite solution optimale globale de (PM). On note par $\arg\min_C f(x)$ l'ensemble des solutions optimales globales.*

Définition 9 *Un point $x^* \in C$ est une solution optimale locale de (PM) s'il existe v voisinage de x^* tel que $f(x^*) \leq f(x), \forall x \in v$ et on note $loc\min_C f(x)$ l'ensemble des solutions optimales locales de (PM).*

Nous avons toujours $\arg\min_C f(x) \subseteq loc\min_C f(x)$ et on a l'égalité entre les deux ensembles si (PM) est convexe.

Remarque 1

Le problème d'optimisation (PM) consiste :

- Soit à chercher un point optimal (local ou global).

- Soit, si un tel point n'existe pas on cherche une borne inférieure à f.

- Soit à établir que f est non bornée inférieurement sur C auquel cas on adopte la convention $\inf_C f(x) = -\infty$.

- Lorsque C est vide on pose par convention $\inf_C f(x) = +\infty$.

1.3.2 Classification des problèmes d'optimisation

Les problèmes d'optimisation sont classifiés selon les caractéristiques des fonctions $f; g_i; h_j$ à savoir la convexité et la différentiabilité. A ce propos, (PM) est dit convexe si f et g_i sont convexes et h_j affines. Si ces dernières sont toutes différentiables on dit que (PM) est un programme différentiable. La classe des programmes mathématiques convexes différentiables est le modèle le mieux élaboré, en absence de la convexité ou de la différentiabilité le problème devient difficile à traiter.

1.3.3 Qualification des contraintes

Une contrainte $g_i(x) \leq 0$ est dite active ou saturée en $x^* \in C$ si elle satisfait $g_i(x^*) = 0$ à ce propos, une contrainte d'égalité est saturée par définition. Un point $x^* \in C$ est dit régulier si les gradients des contraintes saturées en x^* sont linéairement indépendants. On dit également que les contraintes sont qualifiées en x^*.

Remarque 2

La résolution complète de (PM) traite dans l'ordre les points suivants :

- L'existence (et éventuellement l'unicité) d'une solution optimale.

- La caractérisation de la solution (il s'agit des conditions d'optimalité)

- L'élaboration d'algorithmes pour calculer cette solution.

1.3.4 Existence et Unicité

Théorème 1 *[1]*

Soit $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ une fonction continue et C un sous-ensemble non vide de $\mathbb{R}^n$. Le problème (PM) admet au moins une solution optimale globale $x^* \in C$. si l'une des deux conditions suivantes est satisfaite :

1. C est compact ***(théorème de Weierstrass)***

2. la fonction f est coercive ($\lim_{\|x\|\to+\infty} f(x) = +\infty$) sur C fermé.

Remarque 3

L'unicité d'une solution éventuelle est en général une conséquence de la convexité de C et la stricte convexité de f.

1.3.5 Conditions d'optimalité

Théorème 2 ***(Karush-Kuhn-Tucher)****[15]*

Si x^* est une solution optimale locale de (PM) satisfaisant l'une des conditions de qualification précédentes, alors il existe des multiplicateurs $\lambda \in \mathbb{R}^m_+$ et $\mu \in \mathbb{R}^p$ tels que :

$$\begin{cases} \nabla f(x^*) + \sum_{i=1}^{m} \lambda_i \nabla g_i(x^*) + \sum_{i=1}^{p} \mu_i \nabla h_i(x^*) = 0 \quad \textit{(optimalité)} \\ \\ \lambda_i g_i(x^*) = 0, \quad i = 1:m \quad \textit{(complémentarité)} \\ h_j(x^*) = 0, \quad j = 1:p \end{cases}$$

Remarque 4

1. *Si la condition de régularité n'est pas satisfaite, les conditions de* ***K-K-T*** *ne s'appliquent pas (l'existence des multiplicateurs dans ce cas n'est pas assurée)*

2. *Si* (PM) *est convexe, les conditions de* ***K-K-T*** *sont à la fois nécessaires et suffisantes pour que* x^* *soit un minimum global.*

1.4 Classes de problèmes en programmation convexe

1.4.1 Programmation Linéaire

La programmation linéaire est l'une des plus importantes techniques d'optimisation utilisées en recherche opérationnelle. Il faut bien sûr éviter de forcer tout modèle à être linéaire. Par contre, un très grand nombre de modèles constituent des extensions de programmes linéaires. Elle peut se définir comme une technique mathématique permettant de résoudre des problèmes de gestion et particulièrement ceux où le gestionnaire doit déterminer, face à différentes possibilités, l'utilisation optimale des ressources de l'entreprise pour atteindre un objectif spécifique comme la maximisation des bénéfices ou la minimisation des coûts.

Un problème de programmation linéaire mis sous la forme standard définie comme suit :

$$(PL)\begin{cases} \min c^t x \\ Ax = b \\ x \geq 0 \end{cases}$$

où $c \in \mathbb{R}^n,\ b \in \mathbb{R}^m,\ A$ une matrice réelle de type (m, n)

Au problème (PL), on associe son problème dual :

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

Proposition 1 *[1]*

Un programme linéaire réalisable et borné (objectif borné) possède au moins une solution optimale située sur la frontière du domaine réalisable.

Définitions 1

1. *Un point x est un sommet de S si et seulement s'il est une solution réalisable.*

2. *Une base de A est une sous matrice B formée de m colonne linéairement indépendantes de la matrice A.*

3. *La solution du base associée à B est le point $x = (x_B, x_N)$ de $\mathbb{R}^n$ tel que*
$x_B = B^{-1}b,\ x_N = 0$.

4. *Une solution de base qui vérifie $x_B \geq 0$ est dite solution de base réalisable.*

5. *Un sommet est dit non dégénéré si $x_B > 0$. Il est dégénéré dans le cas contraire (au moins une composante de x_B est nulle).*

Proposition 2 *[1]*

Si un programme linéaire possède une solution optimale alors au moins un sommet du domaine réalisable est une solution optimale.

1.4.2 Programmation quadratique

La programmation quadratique est connue par ses applications multiples dans plusieurs domaines. Souvent, on fait intervenir des programmes quadratiques comme procédures intermédiaires pour des programmes non linéaires, c'est le cas entre autre des méthodes de programmation quadratique successives (SQP).

Sans perte de généralité, on peut présenter un programme quadratique sous la forme suivante :

$$(PQ)\begin{cases} \min\left[f(x) = \dfrac{1}{2}x^tQx + c^tx\right] \\ Ax = b \\ x \geq 0 \end{cases}$$

où Q est une matrice carré symétrique d'ordre n, $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$ et $A \in \mathbb{R}^{m \times n}$ de plein rang ($rgA = m < n$).

Rappelons que l'ensemble des contraintes est un polyèdre convexe et fermé, la fonction objectif est infiniment différentiable.

Remarque 5

Si la matrice Q est définie positive, le problème (PQ) est stricte convexe, et on montre, dans ce cas que la solution si elle existe, est unique.

1.4.3 Programmation non linéaire

La programmation non linéaire est la recherche de l'optimum d'une fonction non linéaire sur un sous ensemble convexe d'un espace donné.

On présente les problèmes non linéaires systématiquement sous la forme :

$$(PNL) \begin{cases} \min f(x) \\ g(x) = 0 \\ h(x) \leq 0 \end{cases}$$

où f est une fonction de $\mathbb{R}^n$ dans $\mathbb{R}$ et les fonctions des contraintes g et h sont définies de $\mathbb{R}^n$ dans $\mathbb{R}^m$ et $\mathbb{R}^p$ respectivement.

Ainsi la programmation non linéaire se présente comme étant une généralisation de programmation quadratique et de la programmation linéaire. Bien que cette généralisation soit évidente de nos jours, cet esprit d'unification n'est apparu qu'après 1984, date de l'introduction de l'algorithme de Karmarkar [7] pour la programmation linéaire. Les recherches qui ont suivi ont pu bâtir des ponts entre deux mondes qui étaient jusqu'à alors tout à fait étrangers l'un à l'autre, nous parlerons depuis d'une révolution [15].

1.5 Méthodes de points intérieurs

Historiquement, les méthodes de points intérieurs sont liées directement aux premiers développements de la programmation non linéaire avec contraintes au milieu des années 50, elles sont largement utilisées au cours des années 60, en forme de méthodes de fonctions barrières pour résoudre des programmes non linéaires. Leur utilisation pour la programmation linéaire n'a pas reçu autant d'enthousiasme à cause de la dominance totale de la méthode du simplexe à cette époque. Durant les années 70, elles sont supplantées par des alternatives, à priori plus performantes telle que les méthodes du Lagrangien augmenté. Au début des années 80, les méthodes barrières sont presque universellement regardées comme un chapitre clos de l'optimisation.

Considérons le programme linéaire standard (primal) :

$$(P)\begin{cases} \min c^T x \\ Ax = b,\ x \geq 0 \end{cases}$$

auquel on associe traditionnellement le programme dual standard :

$$(D)\begin{cases} \max b^T y \\ A^T y + s = c,\ s \geq 0 \end{cases}$$

Rappelons qu'un point $x^0 \in \mathbb{R}^n$ est dit intérieur pour (P) si $x^0 > 0$. Si de plus il vérifie $Ax^0 = b$, on dit qu'il est réalisable intérieur. Étant données, une solution réalisable primale x et une solution réalisable duale (y, s), le saut de dualité est tout simplement :

$$c^T x - b^T y = s^T x \geq 0$$

Il y a principalement trois catégories de méthodes de points intérieurs : Les méthodes affines, celles de réduction du potentiel et celles de trajectoire centrale.

a/ Méthodes affines

L'idée remonte à Dikin (1967), puis a été reprise et développée par plusieurs chercheurs au milieu des années 80. Il s'agit pratiquement de l'algorithme de Karmarkar sans fonction potentielle, et où à la place de la transformation projective, on utilise une transformation affine.

Propriétés et caractéristiques

C'est une méthode primale, performante en pratique, quoique sensible au choix initial.

Elle est simple au sens qu'elle ne demande ni transformation projective ni fonction potentielle, ni même la préparation du problème (forme canonique de Karmarkar).

La démonstration de la convergence est relativement simple dans le cas non dégénéré, elle est fastidieuse le cas échéant.

Il n'y a pas de résultat de polynomialité, on pense plutôt que l'algorithme est d'une complexité exponentielle.

C'est sans doute la raison pour laquelle elle est restée inconnue avant les investigations de Karmarkar.

Variantes et extensions

Il existe une version affine duale opérant sur le programme dual (D) et une autre primale duale opérant simultanément sur le programme primal (P) et sur le programme dual (D) pour laquelle on établit la polynomialité dans un cas très spécial.

b/ Méthodes de réduction du potentiel

Ces méthodes sont le fruit direct d'une grande partie des études acharnées menées par plusieurs chercheurs vers la fin des années 80.

Elles sont typiquement désignées pour trouver des solutions améliorantes du pro-

gramme non linéaire suivant :

$$\begin{cases} \min f(x,y,s) = \{q\ln(c^t x - b^t y) - \sum_{i=1}^{n} \ln(x_i)\} \\ Ax = b, \quad x \geq 0 \\ A^t y + s = c, \quad s \geq 0 \end{cases}$$

où la fonction objectif $f(x,y,s)$ est appelée fonction potentiel et q son paramètre. C'est ce type de fonction que Karmarkar introduit dans son papier [7] avec $q = 1$.

La fonction potentiel joue un grand rôle dans le développement des méthodes de points intérieurs et toutes les méthodes de ce type visent à ramener la fonction potentiel à $-\infty$. L'algorithme de Karmarkar appliqué au programme linéaire sous forme standard utilise une fonction de la forme : $q\ln(c^t x - z) - \sum_{i=1}^{n} \ln(x_i)$, où $q = n+1$ et z est une borne inférieure de la valeur optimale de l'objectif. Depuis 1987, les chercheurs introduisent des fonctions du potentiel de types primales duales, parmi les quelles, celle de Tanabe, Todd et Ye [18] définie par : $\Phi_\rho(x,s) = q\ln(x^t s) - \sum_{i=1}^{n} \ln(x_i s_i)$, pour $q > n$. Cette fonction a joué un rôle très important dans le développement des algorithmes de réduction du potentiel après 1988. Les algorithmes correspondants à ces méthodes possèdent une complexité polynomiale, ils nécessitent $O(\sqrt{n}\,|\ln\varepsilon|)$ itérations pour réduire le saut de dualité ($x^t s \leq \varepsilon$, ε est une précision donnée).

Propriétés et caractéristiques

Ces méthodes sont loin d'être aussi simples que les méthodes affines ;

Elles sont plus attractives pour au moins deux raisons :

Leur complexité polynomiale ;

Leur compatibilité avec les techniques de recherche linéaire sur le pas de déplacement, ce qui entraîne un comportement numérique rivalisant.

Elles sont primales-duales ;

Les transformations projectives ne sont pas nécessaires ni en théorie ni en pratique ;

Elles n'ont pas reçu beaucoup d'attention au niveau des simulations numériques, peut être à cause des difficultés algorithmiques rencontrées au début, mais aussi à cause de leur nature de premier ordre.

Variantes

Il existe plusieurs méthodes de réduction de potentiel, de différents types (primales, duales, primales duales) ; certaines utilisent la fonction ci-dessus, pour d'autres on ajoute le terme $-\sum_{i=1}^{n} \ln s_j$, histoire de gagner quelques propriétés théoriques supplémentaires ; le volume calculatoire de l'itération par contre augmente. Toutes les méthodes visent cependant à ramener la fonction potentiel à $(-\infty)$.

c/ Méthodes de trajectoire centrale (TC)

Elles ont été introduites à la même époque que les méthodes de réduction du potentiel, et pleinement développées au début des années 90. Elles possèdent les propriétés théoriques les plus esthétiques : complexité polynomiale, convergence asymptotique super-linéaire et caractère algorithmique newtonien. Ces qualités de confort, placent cette classe de méthodes au centre de l'intérêt primordial des chercheurs pour résoudre effectivement des programmes mathématiques contraints. La trajectoire centrale du programme linéaire (P) est définie à partir de la version barrière paramétrisée de (P) :

$$(P_\mu) : \min \left[c^T x - \mu \sum_{j=1}^{n} \ln(x_j) \quad : \ \mu > 0 \ , \ Ax = b \ , \ x > 0 \quad \right]$$

où $\mu > 0$ est un paramètre fixé.

Ce problème est convexe différentiable, à contraintes linéaires (donc qualifiées partout), les conditions nécessaires et suffisantes d'optimalité au sens de **Karush-Kuhn-Tucker** s'écrivent

$$(1) \begin{cases} Ax = b \ , \ x > 0 \\ A^T y + s = c \ , \ s > 0 \\ XSe - \mu e = 0 \end{cases}$$

X et S sont les matrices diagonales dont les éléments diagonaux sont respectivement x_i et s_i, $i = 1$ à n.

L'ensemble

$$\Gamma = \{(x(\mu), y(\mu), s(\mu)) \ / \ \mu > 0\}$$

est appelé **trajectoire centrale de** (P)

La stratégie des méthodes de trajectoire centrale consiste à chercher des solutions approchées pour le système non linéaire (1), suivant la trajectoire centrale, i.e., en obtenant une suite décroissante du saut de dualité (ou d'une manière équivalente du paramètre μ) qui tend vers zéro à la limite).

A ce propos, on peut envisager un bon nombre de méthodes pour résoudre approximativement le système (1) dont évidemment celle de Newton.

Pour mesurer **la qualité** de la solution trouvée, on introduit la notion de facteur de centralisation ou de proximité défini par :

$$\delta(x, s, \mu) = \left\| \frac{s^t x}{\mu} - 1 \right\|$$

La trajectoire centrale est alors l'ensemble des points réalisables (x, s) de proximité nulle pour un certain μ

Propriétés et caractéristiques

Ce sont des méthodes primales-duales dans lesquelles le saut de dualité est réduit à chaque itération d'un montant fixe jusqu'à convergence après $o(\sqrt{n})$ itérations.

Les performances sont comparables à celles des méthodes affines, pourvu que le point initial soit au voisinage de la trajectoire centrale.

Elles se généralisent facilement au cas non linéaire (convexe), puisqu'il suffit d'écrire les conditions d'optimalité et de définir la trajectoire centrale de la même manière que pour le cas linéaire.

Variantes

Méthodes de prédicteurs-correcteurs

Les méthodes de trajectoire centrale sont aussi considérées comme des variantes des méthodes dites de prédicteurs-correcteurs, lancées comme méthodes à petits pas, avec de bonnes propriétés théoriques, pour évoluer en méthodes à grands pas assez pratiques, quoique la structure des algorithmes se complique et le coût de l'itération augmente.

Méthodes de trajectoire centrale avec poids

Le problème des méthodes de trajectoire centrale est d'ordre pratique. Si le point réalisable initial n'est pas au voisinage de la trajectoire centrale, l'algorithme ne démarre pas. L'introduction de pondérations apporte une amélioration considérable. Techniquement, cela correspond à relaxer l'équation :

$$x_i(\mu)\, s_i(\mu) = \mu, \qquad \forall\, i$$

par

$$x_i\, s_i = \omega_i,\ \forall\, i\,, \quad \omega_i > 0$$

Méthodes de point intérieur non réalisables

Cette approche est développée comme une alternative qui élimine carrément la phase d'initialisation dans les méthodes de trajectoire centrale : on démarre d'un point intérieur non nécessairement réalisable et on tente de réaliser à la fois faisabilité et optimalité. Vraissemblablement, il s'agit (selon les auteurs) de techniques plus performantes que celles des points réalisables.

Ces propos (douteux), n'ont jamais été confirmés numériquement. Des tests préliminaires effectués par différents auteurs ont montré le contraire, en mettant en évidence la supériorité des méthodes réalisables. De même, la complexité arithmétique des méthodes réalisables est considérablement supérieure, et le choix du pas de déplacement est très délicat.

Chapitre 2

Minimisation d'une fonction convexe sous contraintes linéaires par une méthode projective

2.1 Introduction

Ce chapitre est consacré aux développements théoriques des idées fondamentales proposées par A. Keraghel, dans le but de mettre au point une méthode de points intérieurs de type projectif pour minimiser une fonction non linéaire sur un polyèdre convexe. C'est une extension de la méthode de Karmarkar (1984), pour la programmation linéaire que nous rappelons brièvement dans ce chapitre.

On s'intéresse à la résolution des problèmes de la forme :

$$(P)\begin{cases} \min f(x) \\ Ax = b \\ x \geq 0 \end{cases}$$

où $f : \mathbb{R}^n \to \mathbb{R}$ est une fonction non linéaire, convexe et différentiable, A une matrice de type (m, n), $b \in \mathbb{R}^m$ constant.

En mathématique, cette formulation généralise plusieurs classes de problèmes mathématiques à savoir : La programmation linéaire, la programmation quadratique, la programmation fractionnaire ..... De plus, plusieurs problèmes pratiques peuvent être modélisés sous cette formulation (économie, théorie des jeux, problème du consommateur.....).

2.2 Méthode projective de Karmarkar

En 1984, Karmarkar ouvre un nouveau volet dans la programmation linéaire et les méthodes de points intérieurs. L'algorithme de Karmarkar est, donc certainement, l'un des progrès les plus significatifs dans le domaine de la programmation mathématique.

La méthode de Karmarkar est destinée à résoudre les problèmes de la forme :

$$(p.l.r)\begin{cases} \min c^t x = 0 \\ Ax = 0 \\ x \in S_n = \{x \in \mathbb{R}^n, e_n^t x = 1, x \geq 0\} \end{cases}$$

où $c \in \mathbb{R}^n$ (vecteur coût), A une matrice réelle de type $(m \times n)$ de plein rang $(rg(A) = m < n)$ et $e_n = (1, 1, ..., 1)^t \in \mathbb{R}^n$.

Ayant comme solution strictement réalisable de $(p.l.r)$ le centre du simplexe S_n : $x^0 = \dfrac{e_n}{n}$

Remarque 6 *si la valeur optimale est connue à priori mais non nulle l'égalité* $e_n^t x = 1$ *permet de se ramener à un objectif nul. En effet, soit* x^* *une solution optimale du problème et* z^* *la valeur optimale de l'objectif . Alors :*

$$c^t x^* = z^* = z^* e_n^t x^* \Longrightarrow (c - z^* e_n)^t x = (c^{'})^t x = 0$$

On minimise alors l'objectif $(c^{'})^t x$ *au lieu de* $c^t x$, *où* $c_i' = c_i - z^* e_n, i = 1...n$.
Pour le système de contraintes : $Ax = b, b \neq 0$, *on se ramène facilement à un système homogène, il suffit d'écrire :*

$$Ax = be_n^t x \Longrightarrow (A - be_n^t)x = 0.$$

2.2.1 Description de l'algorithme de base

L'algorithme de Karmarkar part d'un point x^0 et construit une suite de points intérieurs x^k qui converge vers une solution optimale x^* en un temps polynômial. A chaque itération, on ramène l'itéré x^k au centre du simplexe S_n par la transformation projective T_k définie comme suit :

$$T_k : x \in S_n \rightarrow T_k(x) = y \in S_n$$

avec

$$T_k(x) = \frac{D_k^{-1} x}{e_n^t D_k^{-1} x} = y \quad \text{et} \quad T_k^{-1}(y) = \frac{D_k y}{e_n^t D_k y}$$

où $D_k = diag\{x^k\}$ et on teste à chaque fois la mesure d'optimalité $\left(c^t x^k \leq \varepsilon\right)$ où ε est la précision recherchée.

Pour plus de détails sur la description de la méthode de Karmarkar, une présentation très détaillée est décrite dans [10].

Algorithme de Karmarkar

Initialisation : $\varepsilon > 0$ est une précision donnée, $x^0 = \frac{e_n}{n}$, $k = 0$

Tant que : $c^t x^k > \varepsilon$ **faire**

Construire $D_k = diag\left\{x^k\right\}$, $A_k = AD_k$, $B_k = \begin{bmatrix} A_k \\ e_n^t \end{bmatrix}$

Calculer $p_k = (I - B_k^t(B_kB_k^t)^{-1}B_k)D_kc$

$$= (I - A_k^t(A_kA_k^t)^{-1}A_k - \frac{1}{n}e_ne_n^t)D_kc,$$

$$d_k = \frac{p_k}{\|p_k\|}$$

Calculer $y^k = \frac{e_n}{n} - \alpha r d_k$, $r = \frac{1}{\sqrt{n(n-1)}}$, $0 < \alpha < 1$

Prendre $x^{k+1} = T_k^{-1}(y^k) = \frac{D_ky^k}{e_n^tD_ky^k}$, $k = k + 1$

Fin tant que.

Fin algorithme.

2.2.2 Fonction potentiel et convergence

Pour établir la convergence de l'algorithme, Karmarkar introduit la fonction dite fonction potentiel définie par :

$$P(x) = n\ln(c^tx) - \sum_{i=1}^{n}\ln(x_i)$$

Karmarkar montre que la décroissance de $P(x)$ implique celle de c^tx , comme l'indique le lemme suivant :

Lemme 1 *[10]*
Soit x^k le $k^{ième}$ itéré de l'algorithme, alors :

$$\frac{c^tx^k}{c^tx^0} \leq (\exp(P(x^k) - P(x^0)))^{\frac{1}{n}}$$

Théorème 3 *[10]*

Si $0 < \alpha \leq 0.25$ *, alors en partant de* $x^0 = \frac{1}{n}e_n$*, après* $O(nq+n\ln n)$ *itérations l'algorithme trouve un point réalisable* x *tel que :*

$$\begin{cases} c^t x = 0 \\ \quad ou \\ \dfrac{c^t x}{c^t x^0} \leq 2^{-q} \end{cases}$$

où q *est une précision fixée.*

2.3 Généralisation de l'algorithme de base

Soit le problème linéaire sous forme standard :

$$(P) \begin{cases} \min c^t x = z^* \\ Ax = b \\ x \geq 0 \end{cases} \tag{2.1}$$

A est une matrice de type (m, n), $c \in \mathbb{R}^n$ est le vecteur coût et $b \in \mathbb{R}^m$ constant.

On suppose que :

H1) Les lignes de la matrice A sont linéairement indépendantes.

H2) On dispose d'un point x^0 strictement réalisable ($Ax^0 = b, x^0 > 0$).

H3) La valeur optimale z^* de l'objectif est connue au départ.

L'hypothèse **1** est classique. L'hypothèse **2** n'est pas du tout restrictive, car on peut toujours obtenir une solution strictement réalisable ou prouver que le problème n'est pas réalisable moyennant une méthode simple de variable artificielle [9].

Nous signalons qu'il existe différentes méthodes pour transformer un P.L.G à un P.L.R et nous citons entre autres dans [10]

La transformation projective notée T_k est une fonction définie comme suit :

$$T_k : \mathbb{R}^n_+ \longrightarrow S_{n+1} \text{ telle que : } T_k(x) = y = \begin{cases} y_i = \dfrac{\dfrac{x_i}{x_i^k}}{1 + \sum\limits_{i=1}^{n} \dfrac{x_i}{x_i^k}}, i = 1 : n \\ y_{n+1} = 1 - \sum\limits_{i=1}^{n} y_i \end{cases}$$

On a :

$$y_i = \frac{x_i}{x_i^k} y_{n+1} \ , \ i = 1 : n$$

et on a :

$$x = T_k^{-1}(y) = \frac{D_k y[n]}{y_{n+1}}$$

On applique la transformation T_k au programme (2.1), on obtient :

$$\begin{cases} \min \dfrac{c^t D_k y[n]}{y_{n+1}} \\ \dfrac{AD_k y[n]}{y_{n+1}} = b \\ \sum\limits_{i=1}^{n} y_i = 1 \\ y[n] \geq 0, y_{n+1} > 0 \end{cases}$$

Or, si la valeur optimale z^* est connue, on peut écrire :

$$\begin{cases} \min \widehat{c}^t y \\ \widehat{A} y = 0 \\ e_{n+1}^t y = 1 \\ y \geq 0 \end{cases} \tag{2.2}$$

où : $\begin{cases} \widehat{c}_i = c_i x_i^k, \quad i = 1, ..., n \\ \widehat{c}_{n+1} = -z^* \end{cases}$, $y = \begin{pmatrix} y[n] \\ y_{n+1} \end{pmatrix}$ et $\widehat{A} = [AD_k \qquad -b]$.

Notons que toute solution réalisable de (2.1) est transformée par T_k en une solution réalisable de (2.2) et réciproquement, toute solution réalisable y de (2.2) avec $y_{n+1} > 0$ est transformée par T_k^{-1} en une solution réalisable x de (2.1).

2.4 Extension de la méthode de Karmarkar

On considère le problème d'optimisation non linéaire suivant :

$$(P)\begin{cases} \min f(x) \\ Ax = b \\ x \geq 0 \end{cases}$$

où $f : \mathbb{R}^n \to \mathbb{R}$ est une fonction non linéaire, convexe et différentiable, A une matrice de type (m, n), $b \in \mathbb{R}^m$ constant, avec les hypothèses suivantes :

H1) La matrice A est de plein rang.

H2) On dispose d'un point x^0 strictement réalisable ($Ax^0 = b, x^0 > 0$).

H3) La valeur optimale z^* de l'objectif est connue au départ.

Remarque 7 *L'hypothèse **3** reste assez restrictive. Traditionnellement, on la relaxe moyennant des approximations (inférieures ou supérieures) de la valeur optimale : l'analyse théorique devient cependant plus difficile à établir.*

2.4.1 Description de la méthode

Préparation du problème

Au début de chaque itération ($k \geq 0$), on dispose de l'itéré x^k , on introduit alors la transformation projective :

$$T_k : \mathbb{R}^n \longrightarrow S_{n+1} \text{ tel que : } T_k(x) = y = \begin{cases} y_i = \dfrac{\dfrac{x_i}{x_i^k}}{1 + \sum_{i=1}^{n} \dfrac{x_i}{x_i^k}}, i = 1 : n \\ y_{n+1} = 1 - \sum_{i=1}^{n} y_i \end{cases}$$

$$x = T_k^{-1}(y) = \frac{D_k y[n]}{y_{n+1}} \quad \text{où} \quad y[n] = (D_k^{-1}x)y_{n+1} = (y_i)_{i=1}^n, \; D_k = diag(x^k).$$

Le problème (P) se transforme comme suit :

$$\begin{cases} \min \left[f(T_k^{-1}(y)) - z^* = f\left(\dfrac{D_k y[n]}{y_{n+1}}\right) - z^* \right] \\ \dfrac{AD_k y[n]}{y_{n+1}} = b \\ e_{n+1}^t y = 1 \\ y[n] \geq 0,\ y_{n+1} > 0 \end{cases} \tag{2.3}$$

Pour avoir la convexité de la fonction objectif, il convient d'écrire (2.3) sous la forme équivalente :

$$\begin{cases} \min \left[g(y) = y_{n+1}(f(T_k^{-1}(y)) - z^*) \right] \\ A_k y = 0 \\ y \in S_{n+1} \end{cases} \tag{2.4}$$

$A_k = [AD_k \qquad -b], \quad y = \begin{pmatrix} y[n] \\ y_{n+1} \end{pmatrix}.$

Notons ici que la valeur optimale de g est égale à 0 et que le centre du simplexe est réalisable pour (2.4).

Notons aussi que la fonction g est convexe sur l'ensemble $Y = \{y \in \mathbb{R}^{n+1} : A_k y = 0, y \in S_{n+1}\}$, conformément au lemme suivant :

Lemme 2 *[8]*
La fonction f étant convexe sur l'ensemble $X = \{x \in \mathbb{R}^n : Ax = b, x \geq 0\}$, il en est de même pour g sur l'ensemble $Y = \{y \in \mathbb{R}^{n+1} : A_k y = 0, y \in S_{n+1}\}$.

Preuve.

Puisque g est une fonction continue sur Y, il suffit de montrer que :

$$g\left(\frac{1}{2}y + \frac{1}{2}\widehat{y}\right) \leq \frac{1}{2}\left[g(y) + g(\widehat{y})\right], \quad \forall y, \widehat{y} \in Y$$

Nous avons : $\forall y, \widehat{y} \in Y, \quad \exists x, \widehat{x} \in \mathbb{R}^n$ telque : $x = T_k^{-1}(y)$ et $\widehat{x} = T_k^{-1}(\widehat{y})$, et alors :

$$\begin{aligned} g\left(\frac{1}{2}y + \frac{1}{2}\widehat{y}\right) &= \frac{1}{2}(y_{n+1} + \widehat{y}_{n+1}) \left[f\left(\frac{D_k\left(\frac{1}{2}y[n] + \frac{1}{2}\widehat{y}[n]\right)}{\frac{1}{2}(y_{n+1} + \widehat{y}_{n+1})} \right) - z^* \right] \\ &= \frac{1}{2}(y_{n+1} + \widehat{y}_{n+1}) \left[f\left(\frac{y_{n+1}}{y_{n+1} + \widehat{y}_{n+1}} x + \frac{\widehat{y}_{n+1}}{y_{n+1} + \widehat{y}_{n+1}} \widehat{x} \right) - z^* \right] \end{aligned}$$

$$\begin{aligned} g\left(\frac{1}{2}y + \frac{1}{2}\widehat{y}\right) &\leq \frac{1}{2}(y_{n+1} + \widehat{y}_{n+1}) \left[\frac{y_{n+1}}{y_{n+1} + \widehat{y}_{n+1}} f(x) + \frac{\widehat{y}_{n+1}}{y_{n+1} + \widehat{y}_{n+1}} f(\widehat{x}) - z^* \right] \\ &\leq \frac{1}{2} y_{n+1} f(x) + \frac{1}{2} \widehat{y}_{n+1} f(\widehat{x}) - \frac{1}{2}(y_{n+1} + \widehat{y}_{n+1}) z^* \\ &\leq \frac{1}{2} y_{n+1} \left[f(x) - z^* \right] + \frac{1}{2} \widehat{y}_{n+1} \left[f(\widehat{x}) - z^* \right] \end{aligned}$$

Donc : $g\left(\frac{1}{2}y + \frac{1}{2}\widehat{y}\right) \leq \frac{1}{2}\left[g(y) + g(\widehat{y})\right]$. ■

Linéarisation

En linéarisant la fonction g au voisinage du point a (centre du simplexe) et en introduisant une boule de centre a considérée comme voisinage de a :

$$g(y) = g(a) + \langle \nabla g(a), y - a \rangle \quad \text{pour} \quad y \in \left\{ y \in \mathbb{R}^{n+1}, \|y - a\| \leq \alpha \right\}$$

Nous sommes ainsi amenés à la résolution du sous problème :

$$\begin{cases} \min \nabla g(a)^t y \\ A_k y = 0 \\ e_{n+1}^t y = 1, \ y \geq 0 \\ \|y - a\|^2 \leq \alpha^2 \end{cases} \tag{2.5}$$

En tenant compte du lemme 3 ci-dessous, le problème (2.5) peut être écrit sous la

forme suivante :

$$\begin{cases} \min \nabla g(a)^t y \\ A_k y = 0 \\ e_{n+1}^t y = 1 \\ \|y - a\|^2 \leq \alpha^2 \end{cases} \tag{2.6}$$

Lemme 3 *[10]*

Si pour un programme linéaire donné on connait une solution réalisable y^0 *tel que* $(y_i^0 > 0,$ $i = 1...n+1)$, *alors l'ellipsoïde :*

$$E = \left\{ x \in \mathbb{R}^n : \sum_{i=1}^{n} \frac{(x_i - y_i^0)^2}{(y_i^0)^2} \leq \beta^2,\ 0 < \beta < 1 \right\}$$

est dans l'intérieur de l'orthant positif de $\mathbb{R}^{n+1}$.

Preuve.

On suppose le contraire, c'est à dire : il existe $j \in \{1, ..., n\}$ tel que $x_i \leq 0$ alors :

$$\sum_{i=1}^{n} \frac{(x_i - y_i^0)^2}{(y_i^0)^2} \geq \frac{(x_j - y_j^0)^2}{(y_j^0)^2} \geq 1 > \beta^2 \quad \blacksquare$$

Lemme 4 *[10]*

la solution optimale du problème (2.6) *est donnée explicitement par :*

$$y^k = a - \alpha d_k$$

où : $d_k = \dfrac{p_k}{\|p_k\|}$ *et* $p_k = p_{B_k}(\nabla g(a))$ *désigne la projection de* $\nabla g(a)$ *sur* $Ker(B_k)$,

$B_k = \begin{bmatrix} A_k \\ e_{n+1}^t \end{bmatrix}$. *e étant le vecteur de* $\mathbb{R}^{n+1}$ *ayant toutes ses composantes égales à* 1.

Preuve.

On pose $x = y - a$, alors on a $B_k x = \begin{bmatrix} A_k \\ e_{n+1}^t \end{bmatrix} (y - a) = 0$, et le problème (2.6) est équivalent à :

$$\begin{cases} \min \nabla g(a)^t x \\ B_k x = 0 \\ \|x\|^2 \leq \alpha^2 \end{cases} \tag{2.7}$$

x^*est une solution de (2.7) si et seulement si $\exists \lambda \in \mathbb{R}^{m+1}, \exists \mu \geq 0$ tel que :

$$\nabla g(a) + B_k^t \lambda + \mu x^* = 0 \tag{2.8}$$

En prémultipliant les deux membres de (2.8) par B_k on trouve :

$$B_k \nabla g(a) + B_k B_k^t \lambda + \mu B_k x^* = B_k \nabla g(a) + B_k B_k^t \lambda = 0$$

(puisque $B_k x^* = 0$)

alors :

$\lambda = -(B_k B_k^t)^{-1}(B_k \nabla g(a))$, en substituant dans (2.8) :

$$x^* = -\frac{1}{\mu}\left(\left[I - B_k^t (B_k B_k^t)^{-1} B_k\right] \nabla g(a)\right) = -\frac{1}{\mu} p_k$$

$$\|x^*\| = \frac{1}{\mu}\|p_k\| = \alpha \Longrightarrow x^* = -\alpha \frac{p_k}{\|p_k\|} = -\alpha d_k$$

et on a : $y^k = y^* = a + x^* = a - \alpha d_k$. ■

Remarque 8

nous appliquons T_k^{-1} pour retourner à la variable initiale, il doit alors commander la réduction de l'objectif du problème initial pour vérifier la convergence de l'algorithme.

2.4.2 Calcul de la projection

Nous avons

$$p_k = (I - A_k^t(A_kA_k^t)^{-1}A_k - \frac{1}{n}e_ne_n^t)\nabla g(a)$$

Posons

$$u_k = (A_kA_k^t)^{-1}A_k\nabla g(a)$$

Donc

$$p_k = \nabla g(a) - A_k^t u_k - \frac{1}{n}e_ne_n^t\nabla g(a)$$

Le calcul de p^k dépend ainsi de la manière par laquelle, on résout le système linéaire :

$$A_kA_k^t u_k = A_k\nabla g(a)$$

dont la matrice $A_kA_k^t$ est symétrique définie positive

On distingue à ce propos, deux types de méthodes :

- Méthodes directes : basées essentiellement sur la factorisation de Cholesky et concernent les systèmes à matrices pleines et de tailles moyennes.
- Méthodes itératives : concernent surtout les systèmes à grande dimension à matrice plutôt creuse, et sont en général de type gradient conjugué.

2.4.3 Calcul du pas de déplacement

Plusieurs alternatives sont proposées dans la littérature entraînant des améliorations considérables pour le calcul du pas de déplacement. Ces alternatives utilisent les méthodes de recherche linéaire. Ces dernières calculent une direction de descente puis l'améliore en résolvant le problème d'optimisation unidimensionnelle suivant :

$$\min_{\alpha}\left\{f(x_k + \alpha d^k) : \alpha \geq 0\right\}$$

2.4.4 Calcul d'une solution initiale strictement réalisable

On considère le problème d'optimisation convexe suivant :

$$(P)\begin{cases} \min f(x) \\ Ax = b \\ x \geq 0 \end{cases}$$

Un point strictement réalisable de (P) est une solution du problème :

$$(R)\begin{cases} Ax = b \\ x > 0 \end{cases}$$

En utilisant la technique de la variable artificielle, le problème (R) est équivalent au problème suivant :

$$(PA)\begin{cases} \min \lambda \\ Ax - \lambda(b - Aa) = b,\ a \in \mathbb{R}^n_+ \\ x > 0,\ \lambda \geq 0 \end{cases}$$

Tel que $a > 0$ est choisi arbitrairement dans l'orthant positif et une variable artificielle.

Posons $y = (x, \lambda)^t, (PA)$ est équivalent au programme linéaire :

$$(PL)\begin{cases} \min c^t y = z^* = 0 \\ \overline{A}y = b \\ y \geq 0 \end{cases}$$

où $c = (0, ...0, 1)^t \in \mathbb{R}^{n+1},\ \overline{A} = [A \quad b - Aa] \in \mathbb{R}^{m\times(n+1)}$.

(PL) vérifie les hypothèses de Karmarkar :

1) $z^* = 0$

2) $y = (a, 1)^t$ solution strictement réalisable de (PA)

3) La matrice A est de plein rang $(rg(A) = m < n + 1)$

La résolution de (R) se ramène à celle de (PL). Plus précisément Karmarkar démontre le théorème suivant :

Théorème 4

$\exists\ \varepsilon_0 > 0$ *telles que les deux propositions suivantes sont équivalentes :*

1- x *solution strictement réalisable de* (R)

2- (PL) *admet une solution optimale*$(x, \lambda)^t$ *telle que* $\lambda \leq \varepsilon_0$

2.4.5 Algorithme

Étape 1.

$\varepsilon > 0$ est une précision donnée.

x^0 est un point strictement réalisable

Étape 2.

Poser $D_k = diag\left\{x^k\right\}$

Construire $A_k = \left[\begin{array}{cc} AD_k & -b \end{array}\right]$, $B_k = \left[\begin{array}{c} A_k \\ e_{n+1}^t \end{array}\right]$

puis $\nabla g(a)$ (le gradient de la fonction transformée au point a)

Étape 3.

Calculer $p_k = (I - B_k^t(B_k B_k^t)^{-1} B_k)\nabla g(a)$

$$= (I - A_k^t(A_k A_k^t)^{-1} A_k - \frac{1}{n} e_n e_n^t)\nabla g(a),$$

Calculer $d_k = \dfrac{p_k}{\|p_k\|}$

Étape 4.

$$\text{Calculer} \quad y^k = a - \alpha r d_k \ , \ r = \frac{1}{\sqrt{n(n+1)}}$$

Étape 5.

$$\text{Calculer} \quad x^{k+1} = T_k^{-1}(y^k) = \frac{D_k y^k\,[n]}{y^k_{n+1}} \quad , \quad k = k+1$$

Étape 6.

Si $\left|f(x^k) - z^*\right| \leq \varepsilon$, **stop**. **Sinon** $k = k+1$ et aller à l'**étape 2**.

Fin algorithme.

2.4.6 Résultats de convergence

Pour contrôler la convergence de l'algorithme, on introduit la fonction potentiel associée au problème (P) définie par :

$$P(x) = (n+1)\ln(f(x) - z^*) - \sum_{i=1}^{n} \ln(x_i).$$

On a :

$$P(x^k) - P(x^0) = \left[(n+1)\ln(f(x^k) - z^*) - \sum_{i=1}^{n} \ln(x_i^k)\right] - \left[(n+1)\ln(f(x^0) - z^*) - \sum_{i=1}^{n} \ln(x_i^0)\right]$$

$$= (n+1)\ln\left[\frac{f(x^k) - z^*}{f(x^0) - z^*}\right] - \left[\sum_{i=1}^{n} \ln(x_i^k) - \sum_{i=1}^{n} \ln(x_i^0)\right]$$

$$\frac{P(x^k) - P(x^0)}{n+1} = \ln\left[\frac{f(x^k) - z^*}{f(x^0) - z^*}\right] - \left[\frac{\sum_{i=1}^{n} \ln(x_i^k) - \sum_{i=1}^{n} \ln(x_i^0)}{n+1}\right]$$

$$\frac{f(x^k)-z^*}{f(x^0)-z^*} = \upsilon(x^k)\exp\left(\frac{P(x^k)-P(x^0)}{n+1}\right)$$

$$\text{avec : } \upsilon(x^k) = \exp\left(\frac{\sum_{i=1}^{n}\ln(x_i^k) - \sum_{i=1}^{n}\ln(x_i^0)}{n+1}\right)$$

Lemme 5 *[8]*

On a

$$g(y^k) < g(a)$$

Preuve.

$$\begin{aligned}
g(y^k) &= g(a) + \left\langle \nabla g(a), y^k - a \right\rangle \quad \text{et} \quad y^k = a - \alpha d_k, \text{ donc} \\
g(y^k) - g(a) &= \langle \nabla g(a), -\alpha d_k \rangle \\
&= \left\langle \nabla g(a), -\alpha \frac{p_k}{\|p_k\|} \right\rangle \\
&= -\frac{\alpha}{\|p_k\|} \langle \nabla g(a), p_k \rangle \\
&= -\frac{\alpha}{\|p_k\|} \|p_k\|^2 \\
&= -\alpha \|p_k\| < 0 \quad \blacksquare
\end{aligned}$$

Lemme 6 *[8]*

Soit y^k la solution optimale du problème (2.5) *alors :*

$$g(y^k) \leq \left(1 - \frac{\alpha}{n+1}\right) g(a)$$

Preuve.

Soit x^* la solution optimale du problème (2.1), et y^* tel que $x^* = T_k^{-1}(y^*)$, on distingue deux cas :

Cas 1) $y^* \in Y \cap B(a,\alpha)$

$$0 \leq g(y^k) \leq g(y^*) = 0 < \left(1 - \frac{\alpha}{n+1}\right) g(a)$$

d'où le résultat.

Cas 2) $y^* \notin Y \cap B(a,\alpha)$

$y^* \notin Fr\left(B(a,\alpha)\right) \cap [a,y^*]$, on a :

$$Fr\left(B(a,\alpha)\right) \cap [a,y^*] = \begin{cases} \left\{y'^k\right\} & \text{(un point réalisable quelconque)} \\ \left\{y^k\right\} & \text{(une solution optimale)} \end{cases}$$

a) $Fr\left(B(a,\alpha)\right) \cap [a,y^*] = \left\{y'^k\right\}$

$$\exists\ \theta \in]0,1[: \begin{cases} y'^k = \theta y^* + (1-\theta)a \\ \left\|y'^k - a\right\| = \alpha \end{cases}$$

$$\left\|y'^k - a\right\| = \alpha \Longleftrightarrow \theta \left\|y^* - a\right\| = \alpha$$

$$\left\|y'^k - a\right\| = \frac{\alpha}{\theta}$$

et on a : $\|y^* - a\|^2 = \|y^*\|^2 + \|a\|^2 - 2a^t y^*$

$$\|y^*\|^2 = \sum_{i=1}^{n+1} (y_i^*)^2 \leq \left(\sum_{i=1}^{n+1} y_i^*\right)^2 = (e_{n+1}^t y^*)^2 = 1$$

$$\|a\|^2 = \left\|\frac{1}{n+1} e_{n+1}\right\|^2 = \frac{1}{n+1}$$

$$a^t y^* = \frac{1}{n+1} e_{n+1}^t y^* = \frac{1}{n+1}$$

$$\|y^* - a\|^2 \leq \|y^*\|^2 + \|a\|^2$$

Alors : $\|y^* - a\|^2 \leq 1 + \frac{1}{n+1} \leq 1 + (n+1) \leq (n+1)^2$

On a aussi : $\left\|y'^k - a\right\|^2 = \left(\frac{\alpha}{\theta}\right)^2$

$$\left(\frac{\alpha}{\theta}\right)^2 \leq (n+1)^2 \Longrightarrow \frac{\alpha}{\theta} \leq n+1 \Longrightarrow \theta \geq \frac{\alpha}{n+1}$$

Puisque g est convexe et $g(y^*) = 0$,

alors : $g(y'^k) \leq \theta g(y^*) + (1-\theta)g(a) \leq (1 - \frac{\alpha}{n+1})g(a)$

b) $Fr\left(B\left(a,\alpha\right)\right) \cap [a, y^*] = \left\{y^k\right\}$

Par une démonstration analogue, on trouve le même résultat. ■

Théorème 5 *[9]*

A chaque itération de l'algorithme, la fonction potentiel se réduit d'une valeur constante δ *telle que :*

$$P(x^{k+1}) \leq P(x^k) - \delta \quad o\grave{u} \quad \delta = \alpha - \frac{\alpha^2}{2(1-\alpha)^2}$$

Preuve.

$$P(x^{k+1}) - P(x^k) = (n+1)\ln\left(\frac{f(x^{k+1}) - z^*}{f(x^k) - z^*}\right) - \sum_{i=1}^{n} \ln\left(\frac{x_i^{k+1}}{x_i^k}\right)$$

$$= (n+1)\ln\left(\frac{g(y^k)}{g(a)}\right) - \sum_{i=1}^{n+1} \ln(y_i^k)$$

$$P(x^{k+1}) - P(x^k) \leq (n+1)\ln\left(1 - \frac{\alpha}{n+1}\right) - \sum_{i=1}^{n+1} \ln(y_i^k)$$

$$\leq -\alpha + \frac{\alpha^2}{2(1-\alpha)^2}$$

On a utilisé le résultat démontré par Karmarkar [9],[5] :

$$-\sum_{i=1}^{n+1} \ln(y_i^k) \leq \frac{\alpha^2}{2(1-\alpha)^2}$$

Donc : $P(x^{k+1}) \leq P(x^k) - \delta$ où $\delta = \alpha - \frac{\alpha^2}{2(1-\alpha)^2}$. ■

Si $\alpha \in [0.27, 0.36]$, alors $P(x^{k+1}) < P(x^k) - 0.2$, donc $P(x^{k+1}) < P(x^0) - (k+1)0.2$.

Théorème 6 *[9]*

Sous les hypothèses :

1. Il existe une solution réalisable x^0 *telle que :* $x^0 \geq 2^{-L}e_n$.

2. La solution optimale x^* *vérifie* $x^* \leq 2^L e_n$.

De plus pour tout x *réalisable on a :* $-2^{3L} \leq z^* \leq f(x) \leq 2^{3L}$.

L'algorithme trouve une solution optimale après $O(nL)$ *itérations.* L *est le nombre total des bits nécessaires*

Preuve.

Puisque $\dfrac{f(x^k) - z^*}{f(x^0) - z^*} = \upsilon(x^k) \exp\left(\dfrac{P(x^k) - P(x^0)}{n+1}\right)$

avec $\upsilon(x^k) = \exp\left(\dfrac{\sum_{i=1}^{n} \ln(x_i^k) - \sum_{i=1}^{n} \ln(x_i^0)}{n+1}\right)$

Par les hypothèses (1) et (2) on a : $\upsilon(x^k) \leq 2^{2L}$ dans la région de réalisabilité, alors :

$$f(x^k) - z^* \leq 2^{2L}(f(x^0) - z^*) \exp\left(\frac{P(x^k) - P(x^0)}{n+1}\right),$$

$$f(x^k) - z^* \leq 2^{3L}2^{2L} \exp\left(\frac{-k\delta}{n+1}\right) \leq 2^{-4L}$$

(Généralement, on dit que (P) est réalisable si on trouve $x^k \in X$, sachant que $f(x^k) - z^* \leq 2^{-4L}$)

On a $k \geq h\,(n+1)\,L$ avec $h \in \mathbb{R}_{+.}^{*}$. ■

Chapitre 3

Expérimentations numériques

3.1 Introduction

Nous présentons dans ce chapitre des expérimentations numériques issues de la mise en œuvre de notre algorithme.

Le programme est réalisé en $C++$, les tests sont effectués sur Intel(R), Pintium(R) 4 CPU 1.70GHz, 1.72MHz de 288 Mo de RAM avec une précision de 10^{-6} idéale pour les méthodes de points intérieurs. Le temps d'exécution et le nombre d'itérations sont enregistrés à titre indicatif.

3.2 Problèmes Quadratiques

les problèmes de 1 à 7 ci-dessous sont des programmes quadratiques convexes standards, c'est-à-dire de la forme :

$$\begin{cases} \min \left[f(x) = \frac{1}{2} x^t Q x + c^t x \right] \\ Ax = b \\ x \geq 0 \end{cases}$$

La programmation quadratique est une classe de problèmes très intéressante pour les raisons suivantes :

1- Elle joue un rôle fondamental dans les développements théoriques et pratiques des méthodes d'optimisation, elle constitue en particulier une classe de problèmes tests à travers lesquels, on peut estimer l'efficacité des méthodes.

2- La programmation quadratique convexe constitue un modèle idéal pour les études comparatives des méthodes de points intérieurs développées jusqu'à présent, notamment les extensions de l'algorithme de Karmarkar.

3- La programmation quadratique est un outil mathématique largement utilisé dans l'étude et la résolution de plusieurs problèmes d'optimisation.

4- Les programmes quadratiques sont fréquemment utilisés comme sous problèmes dans des programmes non linéaires, de plus la programmation quadratique est le seul moyen qui permet une liaison très étroite entre deux domaines très différents à savoir l'optimisation continue et l'optimisation combinatoire dans laquelle les variables sont entières.

3.3 Problème lié à la théorie de l'information

Le problème 8 est un problème lié à la théorie d'information (Eriksson, 1980). La théorie de l'information se préoccupe des systèmes d'information, des systèmes de com-

munication et de leur efficacité.

Le fondateur principal de la théorie de l'information est Claude shannon avec son article "*A Mathematical Theory of Communications*" publié en 1948. C'est un domaine large qui couvre aussi le codage de l'information

La mesure quantitative de redondance d'un texte,

La compression de données,

La cryptographie.

3.4 Éxpérimentations numériques

Exemple 1

$$\begin{cases} \min\left[x_1^2 + x_2^2 + 2x_3\right] \\ s.c \\ x_1 + x_2 + x_3 = 6 \\ -x_1 + x_2 + x_3 = 4 \\ x_1 \geq 0, x_2 \geq 0, x_3 \geq 0 \end{cases}$$

La valeur optimale exacte est $z^* = 10$

La solution exacte est : $x^* = \begin{pmatrix} 1 & 1 & 4 \end{pmatrix}^t$

Résolution

La solution optimale	La valeur optimale	Nombre d'itérations	Temps d'exécution
$\begin{pmatrix} 1.000001 \\ 1.000402 \\ 3.999597 \end{pmatrix}$	10.000000	6	0.010 s

Exemple 2

$$\begin{cases} \min\left[x_1^2 + x_2^2 - 2x_1 - 4x_2\right] \\ s.c \\ -x_1 + x_2 = 1 \\ x_1 + x_2 + x_3 = 2 \\ x_1 \geq 0, x_2 \geq 0, x_3 \geq 0 \end{cases},$$

La valeur optimale exacte est $z^* = -4.5$

La solution exacte est : $x^* = \begin{pmatrix} 0.5 & 1.5 & 0 \end{pmatrix}^t$

Résolution

La solution optimale	La valeur optimale	Nombre d'itérations	Temps d'exécution
$\begin{pmatrix} 0.500000 \\ 1.500001 \\ 0.000000 \end{pmatrix}$	-4.5	1	$0.000\ s$

Exemple 3

$$\begin{cases} \min\left[2x_1^2 + 2x_2^2 - 2x_1x_2^2 - 2x_1^2x_2 - 4x_1 - 6x_2\right] \\ s.c \\ x_1 + x_2 + x_3 = 2 \\ x_1 + 5x_2 + x_4 = 5 \\ x_1 \geq 0, x_2 \geq 0, x_3 \geq 0, x_4 \geq 0 \end{cases},$$

La valeur optimale exacte est $z^* = -7.15594$

La solution exacte est : $x^* = \begin{pmatrix} 1.12904 & 0.77421 & 0.09675 & 0 \end{pmatrix}^t$

Résolution

La solution optimale	La valeur optimale	Nombre d'itérations	Temps d'exécution
$\begin{pmatrix} 1.130533 \\ 0.772858 \\ 0.096585 \\ 0.005214 \end{pmatrix}$	-7.15593	24	$0.016\ s$

Exemple 4

$$\begin{cases} \min\left[x_1^2 + x_2^2 - 3x_1 - 10x_2\right] \\ s.c \\ -x_1 + x_2 + x_3 = 2 \\ 2x_1 + 3x_2 + x_4 = 11 \\ x_1 \geq 0, x_2 \geq 0, x_3 \geq 0, x_4 \geq 0 \end{cases}$$

La valeur optimale exacte est $z^* = -23$

La solution exacte est : $x^* = \begin{pmatrix} 1 & 3 & 0 & 0 \end{pmatrix}^t$

Résolution

La solution optimale	La valeur optimale	Nombre d'itérations	Temps d'exécution
$\begin{pmatrix} 1.000002 \\ 3.000000 \\ 0.000009 \\ 0.000003 \end{pmatrix}$	-23	3	$0.000\ s$

Exemple 5 $(m = 3,\ n = 5)$

$$Q = \begin{pmatrix} 2 & -1 & -1 & 1 & 0 \\ -1 & 6 & -2 & 2 & 0 \\ -1 & -2 & 14 & -4 & 0 \\ 1 & 2 & -4 & 4 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} ;\ c = \begin{pmatrix} 9 \\ -26 \\ 10 \\ -10 \\ 0 \end{pmatrix}$$

$$A = \begin{pmatrix} 1 & 1 & 1 & -1 & 0 \\ 2 & 1 & 1 & -1 & -1 \\ 0 & 0 & -1 & 1 & 0 \end{pmatrix} ; \qquad b = \begin{pmatrix} 4 \\ 3 \\ 1 \end{pmatrix}$$

La valeur optimale exacte est $z^* = -53$

La solution exacte est : $x^* = \begin{pmatrix} 0 & 5 & 0 & 1 & 1 \end{pmatrix}^t$

Résolution

La solution optimale	La valeur optimale	Nombre d'itérations	Temps d'exécution
$\begin{pmatrix} 0.000000 \\ 5.000000 \\ 0.000000 \\ 1.000000 \\ 1.000023 \end{pmatrix}$	-53	3	$0.015\ s$

Exemple 6 $(m = 3,\ n = 5)$

$$Q = \begin{pmatrix} 20 & 1.2 & 0.5 & 0.5 & -1 \\ 1.2 & 32 & 1 & 1 & 1 \\ 0.5 & 1 & 14 & 1 & 1 \\ 0.5 & 1 & 1 & 15 & 1 \\ -1 & 1 & 1 & 1 & 16 \end{pmatrix}; \quad c = \begin{pmatrix} 1 \\ -1.5 \\ 2 \\ 1.5 \\ 3 \end{pmatrix}$$

$$A = \begin{pmatrix} 1 & 1.2 & 1 & 1.8 & 0 \\ 3 & -1 & 1.5 & -2 & 1 \\ -1 & 2 & -3 & 4 & 2 \end{pmatrix}; \quad b = \begin{pmatrix} 9.31 \\ 5.45 \\ 6.60 \end{pmatrix};$$

La valeur optimale exacte est $z^* = 172.733$

Résolution

La solution optimale	La valeur optimale	Nombre d'itérations	Temps d'exécution
$\begin{pmatrix} 2.632979 \\ 0.702611 \\ 1.398617 \\ 2.464023 \\ 1.083801 \end{pmatrix}$	172.733	110	0.016 s

Exemple 7 $(m = 3,\ n = 10)$

$$Q = \begin{pmatrix} 30 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 21 & 0 & 1 & -1 & 1 & 0 & 1 & 0.5 & 1 \\ 1 & 0 & 15 & -0.5 & -2 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & -0.5 & 30 & 3 & -1 & 1 & -1 & 0.5 & 1 \\ 1 & -1 & -2 & 3 & 27 & 1 & 0.5 & 1 & 1 & 1 \\ 1 & 1 & 1 & -1 & 1 & 16 & -0.5 & 0.5 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0.5 & -0.5 & 8 & 1 & 1 & 1 \\ 1 & 1 & 1 & -1 & 1 & 0.5 & 1 & 24 & 1 & 1 \\ 1 & 0.5 & 1 & 0.5 & 1 & 0 & 1 & 1 & 39 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 11 \end{pmatrix}; \quad c = \begin{pmatrix} -0.5 \\ -1 \\ 0 \\ 0 \\ -0.5 \\ 0 \\ 0 \\ -1 \\ -0.5 \\ -1 \end{pmatrix}$$

$$A = \begin{pmatrix} 1 & -1 & 1.9 & 1.25 & 1.2 & 0.4 & -0.7 & 1.06 & 1.5 & 1.05 \\ 1.3 & 1.2 & 0.15 & 2.15 & 1.25 & 1.5 & 0.4 & 1.52 & 1.3 & 1 \\ 1.5 & -1.1 & 3.5 & 1.25 & 1.8 & 2 & 1.95 & 1.2 & 1 & -1 \end{pmatrix}; \; b = \begin{pmatrix} 11.651 \\ 16.672 \\ 21.295 \end{pmatrix}$$

La valeur optimale exacte est $z^* = 264.148$

Résolution

La solution optimale	La valeur optimale	Nombre d'itérations	Temps d'exécution
$\begin{pmatrix} 0.914792 \\ 0.647217 \\ 1.850417 \\ 1.908323 \\ 1.205878 \\ 2.777260 \\ 1.103696 \\ 1.556887 \\ 0.813803 \\ 0.786502 \end{pmatrix}$	264.148	2	0.016 s

Exemple 8 ***Problème lié à la théorie d'information (Eriksson, 1980)***

Voici une classe intéressante de problèmes convexes à contraintes linéaires :

.

$$\min_x \left[f(x) = \sum_{i=1}^{n} x_i \ln \frac{x_i}{a_i},\ Ax = b,\ x \geq 0 \right]$$

On considère maintenant le problème suivant :

$$\begin{cases} \min \left[f(x) = \sum_{i=1}^{n} x_i \ln \frac{x_i}{a_i} \right] \\ s.c \\ x_i + x_{i+m} = b_i,\ i = 1 : m,\ où\ n = 2m \\ x \geq 0 \end{cases}$$

Cas 1 : ($\mathbf{a}_i = 1$, $b_i = 6$)

La solution exacte est $x_i^* = 3, \quad i = 1 : n$

La valeur optimale exacte est $z^* = 3n \ln 3$

1) n = 10

z^*	$f(opt)$	Itérations	Temps d'exécution
32.95836	32.9584	4	0.010 s

La solution optimale trouvée est :

i	1	2	3	4	5	6	7
x_i^*	3.000000	3.000000	3.000000	3.000000	3.000000	3.000000	3.000000

i	8	9	10
x_i^*	3.000000	3.000000	3.000000

2) n = 20

z^*	$f(opt)$	Itérations	Temps d'exécution
65.9167	65.9168	4	0.016 s

La solution optimale trouvée est :

i	1	2	3	4	5	6	7
x_i^*	3.000000	3.000000	3.000000	3.000000	3.000000	3.000000	3.000001

i	8	9	10	11	12	13	14
x_i^*	3.000001	3.000001	3.000001	3.000000	3.000000	3.000000	3.000000

i	15	16	17	18	19	20
x_i^*	3.000000	3.000000	3.000000	3.000001	3.000001	3.000000

3) n = 30

z^*	$f(opt)$	Itérations	Temps d'exécution
98.8751	98.875	4	0.031 s

La solution optimale trouvée est :

i	1	2	3	4	5	6	7
x_i^*	3.000000	2.999999	2.999999	2.999999	2.999999	2.999999	2.999999

i	8	9	10	11	12	13	14
x_i^*	2.999999	2.999999	2.999999	2.999999	2.999999	2.999999	2.999999

i	15	16	17	18	19	20	21
x_i^*	3.000000	2.999999	2.999999	2.999999	2.999999	2.999999	2.999999

i	22	23	24	25	26	27	28
x_i^*	2.999999	2.999999	2.999999	2.999999	2.999999	2.999999	2.999999

i	29	30
x_i^*	2.999999	2.999999

4) n = 50

z^*	$f(opt)$	Itérations	Temps d'exécution
164.7918	164.792	5	0.032 s

La solution optimale trouvée est :

i	1	2	3	4	5	6	7
x_i^*	3.000001	3.000000	3.000000	3.000000	3.000000	3.000000	3.000000

i	8	9	10	11	12	13	14
x_i^*	3.000000	3.000000	3.000000	3.000000	3.000000	2.999999	2.999999

i	15	16	17	18	19	20	21
x_i^*	2.999999	3.000000	3.000000	3.000000	3.000000	3.000000	3.000000

i	22	23	24	25	26	27	28
x_i^*	3.000000	3.000000	3.000000	3.000001	3.000000	3.000000	3.000000

i	29	30	31	32	33	34	35
x_i^*	3.000000	3.000000	3.000000	3.000000	3.000000	3.000000	3.000000

i	36	37	38	39	40	41	42
x_i^*	3.000000	3.000000	2.999999	2.999999	2.999999	3.000000	3.000000

i	43	44	45	46	47	48	49
x_i^*	3.000000	3.000000	3.000000	3.000000	3.000000	3.000000	3.000000

i	50
x_i^*	3.000000

Taille $(m \times n)$	Val. opt. exacte	$f(opt)$	Itérations	Temps d'exécution
50×100	329.5836	329.584	5	0.100 s
100×200	659.1673	659.167	5	0.751 s
150×300	988.7510	988.751	5	2.724 s

Cas 2 : ($\mathbf{a}_i = 2$, $b_i = 4$)

La solution exacte est $x_i^* = 2, \quad i = 1 : n$

La valeur optimale exacte est $z^* = 2n \ln 1 = 0$

1) n = 10

z^*	$f(opt)$	Itérations	Temps d'exécution
0	0	1	0.011 s

La solution optimale trouvée est :

i	1	2	3	4	5	6	7
x_i^*	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000

i	8	9	10
x_i^*	2.000000	2.000000	2.000000

3) n = 30

z^*	$f(opt)$	Itérations	Temps d'exécution
0	0	1	0.016 s

La solution optimale trouvée est :

i	1	2	3	4	5	6	7
x_i^*	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000

i	8	9	10	11	12	13	14
x_i^*	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000

i	15	16	17	18	19	20	21
x_i^*	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000

i	22	23	24	25	26	27	28
x_i^*	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000

i	29	30
x_i^*	2.000000	2.000000

4) n = 50

z^*	$f(opt)$	Itérations	Temps d'exécution
0	0	1	0.032 s

La solution optimale trouvée est :

i	1	2	3	4	5	6	7
x_i^*	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000

i	8	9	10	11	12	13	14
x_i^*	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000

i	15	16	17	18	19	20	21
x_i^*	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000

i	22	23	24	25	26	27	28
x_i^*	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000

i	29	30	31	32	33	34	35
x_i^*	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000

i	36	37	38	39	40	41	42
x_i^*	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000

i	43	44	45	46	47	48	49
x_i^*	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000

i	50
x_i^*	2.000000

Taille $(m \times n)$	Val. opt. exacte	$f(opt)$	Itérations	Temps d'exécution
50×100	0	0	1	0.020 s
100×200	0	0	1	0.060 s
150×300	0	0	1	0.201 s

Cas 3 : ($\mathbf{a}_i = 10$, $b_i = 1$)

La solution exacte est $\quad x_i^* = 0.5, \quad i = 1 : n$

La valeur optimale exacte est $\quad z^* = 0.5n \ln \dfrac{0.5}{10} = 0.5n \ln 0.05$

1) n = 10

z^*	$f(opt)$	Itérations	Temps d'exécution
-14.9786	-14.9787	1	$0.010\ s$

La solution optimale trouvée est :

i	1	2	3	4	5	6	7
x_i^*	0.500000	0.500000	0.500000	0.500000	0.500000	0.500000	0.500000

i	8	9	10
x_i^*	0.500000	0.500000	0.500000

2) n = 20

z^*	$f(opt)$	Itérations	Temps d'exécution
-29.9573	-29.9573	1	$0.010\ s$

La solution optimale trouvée est :

i	1	2	3	4	5	6	7
x_i^*	0.500000	0.500000	0.500000	0.500000	0.500000	0.500000	0.500000

i	8	9	10	11	12	13	14
x_i^*	0.500000	0.500000	0.500000	0.500000	0.500000	0.500000	0.500000

i	15	16	17	18	19	20
x_i^*	0.500000	0.500000	0.500000	0.500000	0.500000	0.500000

3) n = 30

z^*	$f(opt)$	Itérations	Temps d'exécution
-44.9359	-44.936	1	0.010 s

La solution optimale trouvée est :

i	1	2	3	4	5	6	7
x_i^*	0.500000	0.500000	0.500000	0.500000	0.500000	0.500000	0.500000

i	8	9	10	11	12	13	14
x_i^*	0.500000	0.500000	0.500000	0.500000	0.500000	0.500000	0.500000

i	15	16	17	18	19	20	21
x_i^*	0.500000	0.500000	0.500000	0.500000	0.500000	0.500000	0.500000

i	22	23	24	25	26	27	28
x_i^*	0.500000	0.500000	0.500000	0.500000	0.500000	0.500000	0.500000

i	29	30
x_i^*	0.500000	0.500000

4) n = 50

z^*	$f(opt)$	Itérations	Temps d'exécution
-74.8933	-74.8933	1	0.010 s

La solution optimale trouvée est :

i	1	2	3	4	5	6	7
x_i^*	0.500000	0.500000	0.500000	0.500000	0.500000	0.500000	0.500000

i	8	9	10	11	12	13	14
x_i^*	0.500000	0.500000	0.500000	0.500000	0.500000	0.500000	0.500000

i	15	16	17	18	19	20	21
x_i^*	0.500000	0.500000	0.500000	0.500000	0.500000	0.500000	0.500000

i	22	23	24	25	26	27	28
x_i^*	0.500000	0.500000	0.500000	0.500000	0.500000	0.500000	0.500000

i	29	30	31	32	33	34	35
x_i^*	0.500000	0.500000	0.500000	0.500000	0.500000	0.500000	0.500000

i	36	37	38	39	40	41	42
x_i^*	0.500000	0.500000	0.500000	0.500000	0.500000	0.500000	0.500000

i	43	44	45	46	47	48	49
x_i^*	0.500000	0.500000	0.500000	0.500000	0.500000	0.500000	0.500000

i	50
x_i^*	0.500000

Taille $(m \times n)$	Val. opt. exacte	$f(opt)$	Itérations	Temps d'exécution
50×100	-149.7866	-149.786	1	0.010 s
100×200	-299.5732	-299.573	1	0.060 s
150×300	-449.3598	-449.359	1	0.210 s

Exemple 9 *(Problème de la boite)*

Minimiser la surface des boites (à 6 faces planes) de volume V donné.

Puisque les dimensions de la boite est strictement positives, on effectues le changement de variable

$d_{x_1} = \exp(x_1),\ d_{x_2} = \exp(x_2)\,,\ d_{x_3} = \exp(x_3)$, *le problème revient alors à minimiser la fonction* $f(x) = \exp(x_1 + x_2) + \exp(x_2 + x_3) + \exp(x_1 + x_3)$ *sous la contrainte* $x_1 + x_2 + x_3 = \ln(V)$.

En ecrivons la condition nécéssaire de Lagrange, on obtient : $x_1 = x_2 = x_3 = \frac{1}{3}\ln(V)$ *et* $\lambda^* = -2\exp(\frac{2}{3}\ln(V))$. *Dans ce cas, la condition nécéssaire de Lagrange est suffisante, puisque* $x^* = (\frac{1}{3}\ln(V),\ \frac{1}{3}\ln(V),\ \frac{1}{3}\ln(V))$ *est un minimum globale de la fonction*

$$(x_1, x_2, x_3) \longrightarrow f(x) + \lambda^*(x_1 + x_2 + x_3 - 1), \quad \textit{(fonction convexe différentiable)}$$

On prend par exemple le volume $V = 8, \quad \ln(V) = 2.0794$

Soit le programme suivant :

$$\begin{cases} \min\left[\exp(x_1 + x_2) + \exp(x_2 + x_3) + \exp(x_1 + x_3)\right] \\ s.c \\ x_1 + x_2 + x_3 = 2.0794 \\ x_1 \geq 0, \quad x_2 \geq 0, \quad x_3 \geq 0, \end{cases}$$

avec : La valeur optimale exacte est $z^* = 12$

La solution exacte est $x^* = (0.6931\ ,\ 0.6931\ ,\ 0.6931)^t$

Résolution

La solution optimale	La valeur optimale	Nombre d'itérations	Temps d'exécution
$\begin{pmatrix} 0.6931 \\ 0.6931 \\ 0.6931 \end{pmatrix}$	11.9997	1	0.000 s

Exemple 10

$$\begin{cases} \min\left[(x_1 - x_2 + 1)^2 + \exp(x_1 - x_2)\right] \\ s.c \\ 2(x_1 - x_2) + x_3 = 1 \\ x_1 \geq 0, x_2 \geq 0, x_3 \geq 0 \end{cases}$$

La valeur optimale exacte est $z^* = 2$

Résolution

La solution optimale	La valeur optimale	Nombre d'itérations	Temps d'exécution
$\begin{pmatrix} 4.242750 \\ 3.993952 \\ 0.502701 \end{pmatrix}$	2.00003	92	0.032 s

Conclusion

Le travail que nous avons réalisé dans ce mémoire concerne essentiellement l'aspect algorithmique et numérique d'une méthode de point intérieur élaborée par A. Keraghel [10] (pour la programmation linéaire), pour minimiser une fonction convexe différentiable sous contraintes linéaires.

L'aspect théorique a été réalisé en grande partie avec Z. Kebbiche [9].

L'algorithme que nous avons mis au point et implanté a donné des résultats encourageants mettant en évidence la rivalité de cette approche.

L'extension de cet algorithme pour une classe de programme plus général est une perspective très intéressante....

Bibliographie

[1] S. Bazarra, H. D. Sherali and C. M. Shetty, "*Nonlinear Programming, Theory and algorithms*", Second edition (1993).

[2] M. Bouhtou, "*Méthode de points intérieurs pour l'optimisation des systèmes de grande taille*", Thèse de Doctorat. Université de Paris Dauphine (1993).

[3] R. H. Byrd, G. C. Gilbert et J. Nocedal, "*A trust region method based on interior point techniques for nonlinear programming*". Math. Program. 89, pp.149-185. (2000)

[4] R. Fletcher, S. Leyffer, "*Nonlinear Programming without a penalty function*", Numerical Analysis Report NA/171, Department of Mathematics, University of Dundee, Dundee, Scotland.(1997)

[5] A. Hanachi, "É*tude théorique et numérique d'une méthode de linéarisation de type Karmarkar pour la programmation quadratique convexe*", Thèse de Magister, institut de Mathématique, Université Ferhat Abbas, Sétif (1997).

[6] A. Kadiri, "*Analyse numérique des méthodes de points intérieurs pour les problèmes de complémentarité et la programmation quadratique convexe.*", Thèse de Doctorat, INSA de Rouen.(2001)

[7] N. Karmarkar, "*A new polynomial-time algorithm for linear programming*", Combinatorica 4, 373-395 (1984).

[8] Z. Kebbiche, "É*tude et extensions d'algorithmes de points intérieurs pour la programmation non linéaire*". Thèse de Doctorat de Mathématiques appliquées, Université Ferhat Abbas, Sétif (2007)

[9] Z. Kebbiche, A. Keraghel & A. Yassine, "*Extension of a projective interior point method for linearly constrained convex programming*", Applied Mathematics and Computation Vol. 193, Issue 2, 1 November (2007), 553-559.

[10] A. Keraghel, "É*tude adaptative et comparative des principales variantes dans l'algorithme de Karmarkar*". Thèse de Doctorat de Mathématiques appliquées, Université Joseph Fourier, Grenoble, France (1989).

[11] A. Keraghel, "*Analyse convexe* : Théorie fondamentale et exercices", Éditions Dar El'Houda, Ain Mlila, Algérie (2001).

[12] A. Leulmi, "*Une procédure améliorante d'une méthode projective en programmation linéaire*". Thèse de Magister, Université de Skikda.

[13] B. Merikhi, "É*tude comparative de l'extension de l'algorithme de Karmarkar et des méthodes simpliciales pour la programmation quadratique convexe*". Mémoire de Magister, Université de Sétif (1994)

[14] M. Minoux, "*Programmation mathématique* : Théorie et algorithmes", Tome I, Dunod , Paris (1983).

[15] J. Nocedal & S. J. Wright, "*Numerical optimization*". Springer series in operations research.

[16] M. Ouriemchi, A. Yassine, "*An interior point algorithm for nonlinear programming*". soummis à Control et Cybernetics review (2003).

[17] M. Ouriemchi, "*Résolution des problèmes non linéaires par les méthodes de points intérieurs.* Théorie et algorithmes.", Thèse de Doctorat de Mathématiques appliquées, Université du Havre, France (2005).

[18] M. J. Todd, B. P. Burell, "*An extension of Karmarkar's algorithm for linear programming using dual variables*", Algorithmica 1 (1986) 409-424.

[19] S. J. Wright, "*Primal-dual interior point methods*", SIAM. Philadelphia, (1997).

[20] Y. Ye, E. Tse, "*An extension of Karmarkar's projective algorithm for convex quadratic programming*", Mathematical Programming 44 (1989) 157-179.