MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITÉ MOULOUD MAMMERI DE TIZI-OUZOU

FACULTÉ DES SCIENCES

DÉPARTEMENT DE MATHÉMATIQUES

MÉMOIRE DE FIN D'ÉTUDES EN VUE DE L'OBTENTION

DU DIPLÔME DE MASTER EN MATHÉMATIQUES

SPECIALITE : RECHERCHE OPERATIONNELLE

OPTION : AIDE A LA DECISION

THÈME :

La programmation mathématique

Avec les méthodes des points intérieurs.

Présenté Par :

M^{r} AREZKI RAFIK

Devant le jury composé de :

M^{r} M. MORSLI	Professeur	UMMTO	President
M^{r} M. CHEBAH	M.A.A	UMMTO	promoteur
M^{r} K. KASDI	M.A.A	UMMTO	Examinateur
M^{r} M. OUANES	M.C.A	UMMTO	Examinateur
M^{r} M. AOUANE	M.A.A	UMMTO	Examinateur

Sutenue le 08 .07.2013 à 11h00

Remerciements

Je remercie tout d'abord « DIEU » tout puissant de m' avoir donné la santé et le courage pour effectuer ce projet de fin d'étude, dans les meilleures conditions

Comme je tiens à adresser toute la reconnaissance et ma gratitude à : Mon promoteur : Mr. M. CHEBBAH pour ses précieux conseil et son suivi.

Je remercie les membres du jury d'avoir accepter d'examiner mon travail.

Enfin, je tiens à remercier tous ceux qui ont contribués de prés et de loin à la réalisation de ce travail

Rafik AREZKI

Dédicaces

J'ai l'immense plaisir de dédier ce modeste travail à :

Mes très chers parents :HACENE ET BAYA.

Mes frère :MOULOUD,TOUFIK et MOUNIR.

Mes sœurs :mon oiseau KAHINA,KATIA et NADIA.

MA CHERIE KAHINA

A tous les membres de ma grande famille,oncles, tantes, cousins et cousines.

Mes amis(es)

Table de matière

Table de matière

Introduction générale……1

Chapitre 01 : Notions de bases :

1.1. Introduction :

1.2. Ensembles et fonctions :

1.2.1. Ensembles convexes …….3

1.2.2. Fonctions convexes:……...5

1.2.3. Matrices et vecteurs ……..……6

1.2.4. Convergence des suites dans $\mathbb{R}^n$ et dans $\mathbb{R}$……9

Chapitre 02: Programmation mathématique :

2.1. Introduction ……..……10

2.2. Notions de base ……10

2.3. Classification d'un programme mathématique : ……..…….11

2.4. Qualification des contraintes ……12

2.5. Existence et unicité de la solution ……...12

2.6. Conditions d'optimalité (sans contraintes) …….12

2.7. Conditions d'optimalité : (avec contraintes)…….13

Table de matière

Chapitre 03 : Programmation linéaire :

3.1. Introduction ……..……..……..16

3.2. Dualité……..……16

3.3. Autres définitions ……..…..…..17

3.4. La résolution d'un programme linéaire (PL)..……..…..17

3.4.1. La résolution des programmes linéaires par la méthode du simplexe ……..…… ….18

3.4.1.1. Algorithme du simplexe …….18

3.4.1.2. Critère d'optimalité ……19

3.4.1.3. Exemple numérique :……...…….19

3.4.2. La résolution des programmes linéaires par les méthodes des points intérieurs……...……22

3.4.3. Méthode de Karmarkar (présentation de la méthode)…...22

3.4.3.2. Le calcul de direction et de pas …….…….…...24

3.5. Algorithme de base…….……25

3.5.1. Convergence de l'algorithme…….26

3.6. Généralisation de l'algorithme de Karmarkar sur la programmation linéaire……. …….25

3.7. Tests numériques……..……..28

Chapitre 04 : programmation quadratique :

4.1. Introduction…….31

4.2. Méthodes de résolution d'un problème quadratique ……...31

Table de matière

4.3. Complémentarité linéaire monotone……..32

4.4. Transformation d'un programme quadratique convexe en un programme complémentaire linéaire…….32

4.5. Transformation d'un programme complémentaire linéaire monotone en un programme quadratique convexe…….34

4.6. La résolution d'un problème de complémentarité linéaire avec les méthodes des trajectoires centrales …….35

4.6.1. Présentation de la méthode…….35

4.6.1.3. Facteur de centralité……39

4.6.1.4. Concept de proximité……..39

4.6.1.5. Algorithme de la trajectoire centrale pour la programmation quadratique……..……....39

4.6.1.6. La convergence de la méthode……..40

4.6.2. Méthode de trajectoire centrale avec poids

4 .6.2.1. Présentation de la méthode …….40

4.6.2.2. Facteur de centralité ……42

4.6.2.3. Concept de proximité…….42

4.6.2.4. Calcul du pas de déplacement ……42

4.6.2.5. Algorithme de la trajectoire centrale avec poids pour la programmation quadratique ……..…43

4.6.2.6. Convergence de l'algorithme …….44

4.6.2.7. Le calcul d'une solution initiale strictement réalisable ……..…44

4.7. Tests numériques ……....45

Table de matière

Chapitre 05 : la programmation non linéaire convexe avec les méthodes des points intérieurs sous contraintes linéaires :

5.1. Introduction ……..50

5.2. Méthode de Karmarkar pour la résolution du programme non linéaire convexe sur un polyèdre

5.2.1. Présentation de la méthode…….50

5.2.2. Algorithme de Karmarkar pour la programmation non linéaire convexe …….55

5.2.3. Convergence de l'algorithme …….55

5.2.4. Test numériques …….56

5.3. la méthode de trajectoire centrale :

5.3.1. présentation de la méthode ……...58

5.3.2. Algorithme de trajectoire centrale pour la programmation convexe……...61

5.3.3. Méthode de trajectoire centrale avec poids……... 62

5.3.3.1. Présentation de la méthode……..…62

5.3.3.2. Le calcul de la direction de descente……63

5.3.3.4. Calcul d'une solution initiale strictement réalisable ……65

5.3.3.5. Calcul du pas de déplacement .....……66

5.3.3.6. Algorithme de trajectoire centrale avec poids pour la programmation convexe sur un polyèdre…….67

5.3.3.7. Convergence de l'algorithme……..67

5.3.4. Tests numériques…….....69

Table de matière

Chapitre 06 : définition et implémentation de logiciel MATLAB

6.1. Définition de logiciel MATLAB……..….74

6.2. Description de la fenêtre MATLAB ……..…….…..….74

6.2.1. La barre de titre ……..……74

6.2.2. La barre de menu …….…..……75

6.2.3. La barre d'outils ……...…….……..…..76

6.2.4. La fenêtre de commande ……..….76

6.3. Méthode de travail ……..…….76

6.3.1. Edition et sauvegarde des fichiers MATLAB …...76

6.3.2. Aide en ligne ……76

6.3.4. Création de fichiers de commandes et de fonctions utilisateur …….77

6.3.4.1. Fichiers de commande (”script files”) ……...77

6.3.4.2. Fonctions ……...…….77

6.4. Exemples d'illustration ……..78

Conclusion générale …….93

Annexe……..……..94

Bibliographies……..……..95

Introduction générale

Introduction générale:

La **recherche opérationnelle** peut être définie comme l'ensemble des méthodes et techniques rationnelles orientées vers la recherche de la meilleure façon d'opérer des choix en vue d'aboutir au résultat visé ou au meilleur résultat possible.

Dès le XVII^{e} siècle, des mathématiciens comme **Blaise Pascal** tentent de résoudre des problèmes de décision dans l'incertain avec l'espérance mathématique. D'autres, au XVIII^{e} et XIX^{e} siècle, résolvent des problèmes combinatoires. Au début du XX^{e} siècle, l'étude de la gestion de stock peut être considérée comme étant à l'origine de la recherche opérationnelle moderne avec la formule du lot économique (dite formule de **Wilson**) proposée par **Harris** en 1913.

Mais ce n'est qu'avec la Seconde Guerre mondiale que la pratique va s'organiser pour la première fois et acquérir son nom. En 1940, **Patrick Blackett** est appelé par l'état-major anglais à diriger la première équipe de recherche opérationnelle, pour résoudre certains problèmes tels que l'implantation optimale de radars de surveillance ou la gestion des convois d'approvisionnement. Le qualificatif « **opérationnelle** » vient du fait que la première application d'un groupe de travail organisé dans cette discipline avait trait aux opérations militaires. La dénomination est restée par la suite, même si le domaine militaire n'est plus le principal champ d'application de cette discipline.

Après la deuxième guerre mondiale, la méthode du simplexe s'est imposée comme la seule méthode efficace pour la résolution des programmes linéaires (PL).

EN 1979, **Kachian** a découvert le premier algorithme de type point intérieur projectif de complexité polynomiale, pour des instances des grandes dimensions l'algorithme s'avéra inefficace. En 1984 **Karmarkar** a crée un algorithme qui révolutionna toutes les recherches dans les domaines des algorithmes de complexité polynomiale, grâce à son algorithme **Karmarkar** a palie aux insuffisances de l'algorithme de **Kachian**. L'apparition de l'algorithme de **Karmarkar** a permis le développement des méthodes de points intérieurs qui se sont révélées comme une véritable alternative à la méthode du simplexe.sur tout après la découverte par **Klee & Minty** en 1972 d'un exemple prouvant que le simplexe peut aller jusqu'à une complexité exponentielle.

Les méthodes de points intérieurs partent d'un point intérieur au domaine des solutions réalisables, puis au moyen d'une stratégie fixée (trajectoire centrale,...) déterminent une valeur approchée de la solution optimale. Parmi les avantages de ces méthodes par rapport à la méthode du simplexe, on peut citer par exemple la complexité polynomiale, la convergence super linéaire et leur bon comportement numérique.

En 1994, **Den Hertog** a classé les méthodes de points intérieurs en quartes catégories :

- Méthodes de chemin central.
- Méthodes affines et mise à l'échelle.
- Méthodes projectives.
- Méthodes affines avec potentiel.

Introduction générale

Depuis quelques années, les chercheurs essaient de généraliser le principe de ces méthodes, d'abord pour la programmation quadratique, puis pour la programmation non linéaire.

Notre travail est consacré à l'étude numérique de deux méthodes des points intérieurs la méthode de **Karmarkar** qui est de type projectif, et la trajectoire centrale avec et sans poids de type **chemin central**, pour la programmation linéaire, et leur extension dans le domaine de la programmation quadratique, non linéaire sous contraintes linéaires.

Le mémoire est organisé comme suit :

Les deux premiers chapitres introductifs serrent à une introduction générale. Celle ci contient un survol rapide : éléments d'analyse convexe et d'algèbre, ensembles, vecteurs, matrices, dérivées, suites, convergence des suites, la continuité des fonctions, la convexité des ensembles et fonction convexe, par la suite : la programmation mathématique, classification d'un problème mathématique, conditions d'optimalité dans la programmation linéaire, la programmation quadratique, et la programmation non linéaire...etc.

Le troisième chapitre est réservé d'abord pour la présentation et la résolution avec la méthode du simplexe à la programmation linéaire, puis à la méthode de **Karmarkar**, dont on a présenté les principes et les résultats de convergence étudiés par **Karmarka**r [2], on a ainsi montré la convergence polynomiale de cet algorithme suivie par des tests numériques sur différents exemples.

Le quatrième chapitre est réservé pour la méthode de trajectoire centrale sans et avec poids appliquées à la programmation de la complémentarité linéaire comme formulation unificatrice de la programmation linéaire et la programmation quadratique convexe. D'une part, on a présenté le principe et les résultats de convergence étudiée par **A. Keraghel** et **Z. Kebbiche** [3], et on a soulevé les inconvénients de cette méthode dont le problème majeur est celui de l'initialisation. Pour cela, dans la deuxième partie, en s'inspirant de l'algorithme de **Ye-Lustig(annexe)**, on a pu calculer une solution primale-duale strictement réalisable. On a ainsi montré la convergence polynomiale de l'algorithme en question suivie par des tests numériques sur différents exemples.

Le cinquième chapitre dans la première partie, on s'est intéressé à l'extension de la méthode de **Karmarka**r pour la programmation non linéaire convexe à contraintes linéaires. Dés premier temps, on a fait une synthèse générale contenant le principe pour la méthode de **Karmarkar**, et les techniques de linéarisation combinées avec l'approche projective, puis des tests numériques qu'on a exécutés sur différents exemples.

dans la deuxième partie, On a fait une synthèse générale contenant le principe sur la méthode de trajectoire centrale (sans et avec poids), le problème barrière perturbé (sans et avec poids) et sa résolution via l'approche newtonienne. Signalons que l'opération la plus coûteuse dans l'algorithme obtenu réside dans la direction qui nécessite le calcul de l'inverse de la matrice ($\nabla^2 f(x) + X^{-1}S$), à chaque itération. L'algorithme décrit est sanctionné par des tests numériques qu'on a exécutés sur différents exemples.

A la fin, dans le dernier chapitre, on terminera par définir et implémenter le logiciel de programmation **MATLAB**, a fin de concrétiser notre étude.

1.1. Introduction :

Dans ce chapitre introductif, on va présenter quelques notions de bases essentielles d'analyse et d'algèbre sur la programmation mathématique, qui nous permettent de résoudre de nombreux problèmes. Et on va introduire une notion qui joue un rôle très important : celle de la convexité des fonctions et des ensembles:

1.2. Ensembles et fonctions :

1.2.1. Ensembles convexes :

Définition 1.2.1.1:(point intérieur**)**

Soit $D \subseteq \mathbb{R}^n \; et \; y \epsilon Y$:

On dit que y est intérieur à D s'il existe un voisinage de y contenu dans D.

D'une manière équivalente, y est intérieur à D s'il existe $\varepsilon > 0$ tel que :

$$z \in Y \;,\; \forall z \; tel \; que \;\; \|z - y\| \leq \varepsilon \; .$$

L'ensemble des points intérieurs à Y est appelés l'intérieur de D et noté $\boldsymbol{int}(Y)$.

Exemple :

L'ensemble $A = \{x / \|x\| \leq 1\}$ a pour intérieur la boule unité $B_1(0) = \{x / \|x\| < 1\}$.

Définition 1.2.1.2. : (Sous ensembles ouverts**)**

Un sous ensemble $\boldsymbol{S}$ est dit ouvert s'il coïncide avec son intérieur c'est-a-dire si :

$$\boldsymbol{S = int(S)}.$$

Exemple :

$B_r(a) = \{p \epsilon \mathbb{R}^n \, / \|p - a\| < r\}$ est un ensemble ouvert.

Définition 1.2.1.3. : (Sous ensembles fermé**)**

Un sous-ensemble $\boldsymbol{S} \subseteq \mathbb{R}^n$ est dit fermé si son complémentaire est ouvert.

Exemple:

$B_r(a) = \{p \epsilon \mathbb{R}^n / \|p - a\| \leq r\}$ est un ensemble fermé.

Définition 1.2.1.4. : (Ensemble compact**)**

On dit que $D \subseteq \mathbb{R}^n$ est borné si :

$$\forall x \in D, \exists r > 0 \; tel \; que : \|x\| < r \Leftrightarrow x \in B_r(0) .$$

Définition 1.2.1.5 :

Un ensemble compact de $\mathbb{R}^n$ est un ensemble fermé borné sur $\mathbb{R}^n$.

Définition 1.2.1.6 : (Convexité des ensembles**)**

Un ensemble $D \subseteq \mathbb{R}^n$ est dit convexe si et seulement si :

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

Définition 1.2.1.7 :

Un ensemble $D \subseteq \mathbb{R}^n$ est dit affine si et seulement si :

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

Définition 1.2.1.8:

On appelle hyperplan dans $\mathbb{R}^n$ un ensemble de la forme :

$\pi = \{X \in \mathbb{R}^n / \langle C, X \rangle = \alpha\}$. $\langle , \rangle$:est le produit cartésien.

Où $C \neq 0$. Dans l'espace $\mathbb{R}^n$; un hyperplan défini deux demi-espaces

$$\{X \in \mathbb{R}^n / \langle C, X \rangle \leq \alpha\}. \text{ Et } \{X \in \mathbb{R}^n / \langle C, X \rangle \geq \alpha\}.$$

Définition 1.2.1.9 :

Un ensemble $D \subseteq \mathbb{R}^n$ est un polyèdre si et seulement si :

$$D = \{ x \epsilon \mathbb{R}^n / \; A x \leq b, \text{Où } A \epsilon \mathbb{R}^{n*m}, x \epsilon \mathbb{R}^n, b \epsilon \mathbb{R}^m \} .$$

Définition 1.2.1.10 : (la projection d'un point sur un ensemble convexe)

On appelle projection d'un point x_0 sur un ensemble convexe fermé D un point p_{x_0} tel que :

$$\|p_{x_0} - x_0\| = min_{x\epsilon D}\|x - x_0\|.$$

On dit que $\|p_{x_0} - x_0\|$ est la distance minimale entre x_0 et sa projection en D.

1.2.2. Fonctions convexes:

Définition 1.2.2.1 :

On appelle une fonction de plusieurs variables et à valeurs réelles une fonction :

$$f: \mathbb{R}^n \longrightarrow \mathbb{R}: pour\ x = (x_1, \dots., x_n)^t \in \mathbb{R}^n, f(x) = f(x_1, \dots, x_n) \in \mathbb{R}.$$

Définition 1.2.2.2:

Une fonction $f: \mathbb{R}^n \longrightarrow \mathbb{R}$ est continue au point $x = (x_1, \dots, x_n)^t \in \mathbb{R}^n$ si et seulement si pour tout $\varepsilon > 0$, il existe $\alpha > 0$ tel que pour tout

$y = (y_1, \dots, y_n)^t \in \mathbb{R}^n$:

$$\|y - x\| \leq \alpha \Rightarrow |f(y) - f(x)| \leq \varepsilon.$$

Définition 1.2.2.3 : (fonctions convexes**)**

Une fonction $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ est dite convexe sur un ensemble D convexe si elle vérifie :

$$\begin{cases} \forall\ x, y \in D. \\ \forall\ \lambda \in [0,1]. \\ f\left((1-\lambda)x + \lambda y\right) \leq (1-\lambda)f(x) + \lambda f(y). \end{cases}$$

Définition 1.2.2.4 : (fonctions affines**)**

f est dite affine si elle vérifie :

$$\begin{cases} \forall\, x, y \in D. \\ \forall\ \lambda \in \mathbb{R}. \\ f\left((1-\lambda)x + \lambda y\right) = (1-\lambda)f(x) + \lambda f(y). \end{cases}$$

1.2.3. Matrices et vecteurs :

Une matrice A Є $\mathbb{R}^{m*n}$ est notée A= (a_{ij}) ,1 ≤ i ≤ m, 1≤ j ≤ n.

Le $j^{ème}$ vecteur colonne de A est $a_j = Ae_j$ /$A = (a_1, a_2, \dots a_n).e_j$:est le vecteur unitaire.

Un système de n vecteurs $u_m \in \mathbb{R}^m$ est noté sous forme matricielle :

$U = (u_1, \dots . u_n) \in \mathbb{R}^{m*n}$ Avec u_j le $j^{ème}$vecteur colonne U.

Définition 1.2.3.1 : (Matrice transposée**)**

La matrice transposée $A^t = A'$, d'une matrice A est la matrice obtenue en inter-changeant les lignes et les colonnes .soit $a'_{ij} = a_{ji}$.

Remarques :

Soit A une matrice carrée, on a :

1) $A^tA\ et\ AA^t$ sont symétriques.

2) On a $(A^t)^t = A$ et $(AB)^t = B^tA^t$.

Définitions 1.2.3.2 :

Soit A une matrice carrée symétrique d'ordre n.

1. A est dite définie positive si : $x^tAx > 0$, $\forall\ x \neq 0$, $x \in \mathbb{R}^n$.
2. A est dite semi définie positive si : $x^tAx \geq 0$, $x \in \mathbb{R}^n$.
3. A est dite définie négative si :$x^tAx < 0$, $\forall\ x \neq 0$, $x \in \mathbb{R}^n$.
4. A est dite semi définie négative si :$x^tAx \leq 0$, $x \in \mathbb{R}^n$.

Définition 1.2.3.3 :

Une norme matricielle est une application $\|.\| : M_n(\mathbb{k}) \to \mathbb{R}_+$ qui vérifie les propriétés suivantes :

1. $\|A\| = 0 \Leftrightarrow A = 0, et\ \|A\| \geq 0, \forall A \epsilon M_n(\mathbb{k})$
2. $\|\alpha A\| = |\alpha|\|A\|\ ,\ \forall \alpha\ \epsilon\ \mathbb{k}\ .$
3. $\|A + B\| \leq \|A\| + \|B\|, \forall A, B \epsilon M_n(\mathbb{k}).$
4. $\|A \times B\| \leq \|A\| \times \|B\|, \forall A, B \epsilon M_n(\mathbb{k}).$
5. $\|A.x\| \leq \|A\| \times \|x\|, \forall x \epsilon \mathbb{R}^n.$

Exemple :

Les trois normes usuelles sont :

a. $\|A\|_1 = max_{j:\overline{1,n}} \sum_{i=1}^{n} |a_{ij}|$

b. $\|A\|_2 = \sqrt{\rho(A^2)}$; avec :

$\rho(A^2)$:rayon spectral de A. $\|A\|^2 = max_{1 \leq i \leq n}(|\lambda_i|).$

λ_i : Valeur propre de A^2.

c. $\|A\|_\infty = max_i \sum_{j=1}^{n} |a_{ij}|$.

Définition 1.2.3.4: **(**matrice gradient**)**

Soit $G = (g_1, g_2 \dots g_m)^t : \mathbb{R}^n \to \mathbb{R}^m$ tel que $g_i : \mathbb{R}^n \to \mathbb{R}$ est différentiable, pour $i = 1 \dots m$.dans ce cas, G est différentiable, et la fonction $\nabla G(x) : \mathbb{R}^n \to \mathbb{R}^{n \times m}$ est appelé matrice gradient et est définie par :

$$\nabla G(x) = \begin{pmatrix} \frac{\partial g_1}{\partial x_1} & \dots & \frac{\partial g_m}{\partial x_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial g_1}{\partial x_n} & \dots & \frac{\partial g_m}{\partial x_n} \end{pmatrix}.$$

Remarque :

La matrice gradient est souvent utilisée dans sa forme transposée et est appelée matrice Jacobéenne de G .

Définition 1.2.3.5: (matrice Jacobéenne**)**

Soit $G: \mathbb{R}^n \to \mathbb{R}^m$ .la fonction $JG(x): \mathbb{R}^n \to \mathbb{R}^{m\times n}$ est appelée matrice Jacobéenne et est définie par :

$$JG(x) = \nabla G(x)^t = \begin{pmatrix} \nabla g_1(x)^t \\ \vdots \\ \nabla g_m(x)^t \end{pmatrix} = \begin{pmatrix} \frac{\partial g_1}{\partial x_1} & \frac{\partial g_1}{\partial x_2} & \dots & \frac{\partial g_1}{\partial x_n} \\ \frac{\partial g_2}{\partial x_1} & \frac{\partial g_2}{\partial x_2} & \dots & \frac{\partial g_2}{\partial x_n} \\ \vdots & \vdots & \dots & \vdots \\ \frac{\partial g_m}{\partial x_1} & \frac{\partial g_m}{\partial x_2} & & \frac{\partial g_m}{\partial x_n} \end{pmatrix}.$$

Définition 1.2.3.6:(le second ordre**)**

Nous pouvons effectuer la même analyse de différentiabilité faite sur la fonction ∇G dans la section (1.3.4) pour chacune des fonctions $\nabla_i g(x)$ de la définition (2.3.4).

La $j^{\text{ème}}$ dérivée partielle de $\nabla_i g(x)$ est la dérivée seconde de $g(x)$, par rapport aux variable x_i et x_j car : $\frac{\partial\, \nabla_i g(x)}{\partial x_j} = \frac{\partial(\partial\, g(x)/\partial x_i)}{\partial x_j} = \frac{\partial^2\, g(x)}{\partial x_i\, \partial x_j}$.

Définition 1.2.3.7: (Matrice Hessienne**)**

Soit $g: \mathbb{R}^n \longrightarrow \mathbb{R}$ une fonction deux fois différentiable.

La fonction notée $\nabla^2 g(x): \mathbb{R}^n \to \mathbb{R}^{n\times n}$ est appelée matrice Hessienne de g et est définie par :

$$\nabla^2 g(x) = \begin{pmatrix} \frac{\partial^2 g(x)}{\partial x_1^{\,2}} & \frac{\partial^2 g(x)}{\partial x_1\, \partial x_2} & \dots & \frac{\partial^2 g(x)}{\partial x_i\, \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 \mathrm{g}(x)}{\partial x_n\, \partial x_1} & \frac{\partial^2 \mathrm{g}(x)}{\partial x_n\, \partial x_2} & \dots & \frac{\partial^2 g(x)}{\partial x_n^{\,2}} \end{pmatrix}$$

<u>Remarque</u> :

La matrice Hessienne est toujours symétrique.

Notons que la Hessienne de g est la matrice gradient de la matrice ∇g.

1.2.4. Convergence des suites dans $\mathbb{R}^n$ et dans $\mathbb{R}$:

Définition 1.2.4.1 :

On définit une suite dans $\mathbb{R}^n$ par : $(p^k)_{k>0}$ avec : $p^k = (p_1^k, p_2^k, \dots\dots, p_n^k)$ et $p_i^k \in \mathbb{R}$ tel que : $i = 1,2, \dots\dots, n$.

Exemple : $p^k = \left(\frac{1}{k+1}, \frac{2k}{1+k^2}\right)$ une suite dans $\mathbb{R}^2$.

Définition 1.2.4.2 :

Soit $(p_k)_{k>0}$ une suite de points (ou vecteurs) de $\mathbb{R}^n$ **.** on dit que cette suite converge vers une limite $p \in \mathbb{R}^n$ si et seulement si pour tout $\varepsilon > 0$, il existe $k_0$tel que

Quel que soit : $k \geq k_0 \Rightarrow \|p^k - p\| \leq \varepsilon$.

On note alors $\lim_{k\to\infty} p^k = p$ ou encore parfois $p^k \to p$.dans le cas contraire on dit que la suite diverge.

Dans le cas $n = 1$, on a $\|p^k - p\| = |p^k - p|$ et on trouve la définition usuelle de la convergence des suites réelles.

Remarque:

Si (p^k) converge, sa limite est unique.

Définition 1.2.4.3 :

Une suite $(p^k)_{k>0}$ est dite ‘’de Cauchy’’ si et seulement si pour tout $\varepsilon > 0$, il existe k_0 tel que $k, l > k_0 \Longrightarrow \|p^k - p^l\| \leq \varepsilon$.

Remarque :

Dans $\mathbb{R}^n$, p^k suite de Cauchy $\Leftrightarrow$ (p^k) convergente.

- Dans ce chapitre, on a donné quelques définitions et des théorèmes généraux des fonctions et des ensembles, qui vont nous aider à les étudier dans les chapitres prochains.

2.1. Introduction :

La programmation mathématique, et plus particulièrement l'optimisation vise à résoudre des problèmes où l'on cherche à déterminer parmi un grand nombre de solutions candidates celle qui donne le meilleur rendement. Plus précisément, on cherche à trouver une solution satisfaisant un ensemble de contraintes qui minimise ou maximise la fonction objectif donnée. L'application de la programmation mathématique est en expansion croissante et se retrouve dans plusieurs domaines.

Dans ce chapitre, nous allons présenter les problèmes classiques d'optimisation, et leurs conditions d'optimalité et définir quelques notions usuelles.

D'une façon générale un programme mathématique dans $\mathbb{R}^{n}$ est un problème d'optimisation sous la forme suivante :

$$(P1)\ \begin{cases} Min\ f(x) \text{ ou } Max\ f(x) \\ \text{sc} \\ g_i(x) \leq 0\ , i : 1..m \\ h_j(x) = 0\ , j : 1..p \\ x \in s \subseteq \mathbb{R}^{n} \end{cases}$$

Où $f: \mathbb{R}^{n} \to \mathbb{R}$ est une fonction continûment différentiable appelée fonction objectif, et l'ensemble des conditions $\{ g_i(x) \leq 0\ , i : 1..m; h_j(x) = 0\ , j : 1..p, x \in s \subseteq \mathbb{R}^{n} \}$ sont les contraintes du problème (P1) .L'ensemble $D = \{x \in s \subseteq \mathbb{R}^{n} / g_i(x) \leq 0\ , i : 1..m; h_j(x) = 0\ , j : 1..p\}$ est appelé l'ensemble des solutions réalisable du problème (P1).

2.2. Notions de base :

Définition 2.2.1 :

On appelle solution réalisable du problème (P1), tout point vérifiant les contraintes de ce problème (P1) (appartenant à D).

Définition 2.2.2 :

Un point $x^* \epsilon D$ qui minimise la fonction objectif sur D dit solution optimale du problème (P1) si et seulement si :

$$\forall x \epsilon D \, , \, f(x^*) \leq f(x).$$

On note par $\arg \min_D f(x)$, l'ensemble des solutions optimales du problème.

Définition 2.2.3 :

Un point $x^* \epsilon D$ est une solution optimale locale de (P1), s'il existe un voisinage V de x^*, et un $\varepsilon > 0$ tel que :

$$\begin{cases} f(x^*) \leq f(x) \, , \forall x \epsilon V \, (x^*) \\ \text{où} \\ V(x^*) = \{ x \epsilon D \, / \, || x - x^* || < \varepsilon \} \end{cases}$$

(x appartient au voisinage de $V(x^*)$) et on note par $\text{locmin}_D f(x)$ l'ensemble des solutions optimales locales de (P1).

Et de plus si f et D sont convexes, le minimum local est global pour le problème (P1).

Remarques :

Le problème d'optimisation précedent consiste :

- soit à chercher un point optimal (local, global).
- soit, si un tel point n'existe pas on cherche une borne inférieure à la fonction f.
- soit, à établir que f est non borné inférieurement sur D ,auquel cas on adopte la convention $\inf_D f(x) = -\infty$.
- lorsque D est vide on pose par convention $\inf_D f(x) = +\infty$

2.3. Classification d'un programme mathématique :

On classifie le problème (P1) selon des propriétés fondamentales, à savoir la convexité, la différentiabilité et la continuité des fonctions du problème (P1).

Définition 2.3.1 :

- Un programme mathématique est dit convexe si : f, g_i et h_j sont lineaires sur un ensemble D des solutions réalisables convexe.
- Si :f, g_i et h_j sont différentiable le problème est dit différentiables .

Signalons que, les programmes non convexes ou non différentiables sont les plus difficiles à traiter.

2.4. Qualification des contraintes :[13]

Pour que la qualification des contraintes **(QC)** soit vérifiée en tout point x dans D, il suffit que l'une des conditions (**1**) ou (**2**) soit réalisée:

(1)-toutes les contraintes de (P1) sont linéaires, affines (**karlin 1959**) ou convexe.

(2)-toutes les contraintes d'inégalités $g_i(x)$ (I $= 1 \ldots$ m) sont convexes et il existe $\bar{x} \in \mathbb{R}^n$ vérifiant $g_i(\bar{x}) < 0$, $\forall$i $\in$ I et $h_j(x)$ $(j = 1 \ldots p)$ affines, (**Slater 1950**).

-Pour que (QC) soit vérifiée en un point $x° \in D$ il suffit que l'on ait :

(3) Les gradients $\nabla g_i(x°)$(i $= 1 \ldots$ m), $\nabla h_j(x)$, (j: 1 ... p) des contraintes saturées en $x°$ sont linéairement indépendantes (**Fiacco et Mc Cornick 1968)**.

Remarques :

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

- L'existence d'une solution optimale.
- La caractérisation de la solution.
- L'élaboration d'algorithme pour calculer cette solution.

2.5. Existence et unicité de la solution :[14]

Théorème 2.5.1 :(weierstrass)

Si D l'ensemble des solutions, est compact non vide de $\mathbb{R}^n$ et si f est continue sur D alors (PM) admet au moins une solution optimale globale $x^* \in D$.

2.6. Conditions d'optimalité (sans contraintes) :[14](cas sanscotraintes)

Soit le programme non linéaire et non contraintes (**PNC**) :

$$(p1) \quad \begin{cases} Min\, f(x) \\ x \epsilon\, D \subseteq \mathbb{R}^n \end{cases}$$

Où la fonction $f(x)$ est au moins deux fois continûment différentiables.

Définition 2.6.1:

Soit $x^* \in D$, on dit que $y \epsilon\, D$ une direction admissible s'il existe $\alpha > 0$ tel que $x^* + ty \in D, \quad \forall\, t \in [0\ ,\ \alpha]$.

• Soit x^* un minimum local de $f(x)$, on a alors nécessairement pour tout t > 0 assez petit.

$f(x^* + ty) - f(x^*) \geq 0$, $\forall y$ admissible ceci implique :

$\lim_{t \to 0} \frac{f(x^*+ty)-f(x^*)}{t} = \nabla f(x^*)\, y \geq 0$.

En particulier, la direction $y = -\nabla f(x^*)$ est admissible et on déduit :

$$- \|\nabla f(x^*)\|^2 \geq 0 \Rightarrow \|\nabla f(x^*)\|^2 \leq 0 \Rightarrow \nabla f(x^*) = 0.$$

Théorème 2.6.1: (condition nécessaire du 1er ordre) (cas sans contraintes) [9]

Soit x^* minimum local pour (P1), on a alors $\nabla f(x^*) = 0$.

Un point x satisfaisant cette condition est appelé un point stationnaire.

Au deuxième ordre, on obtient :

$$f(x^*) \leq f(x^* + ty) = f(x^*) + t\,\nabla f(x^*)^t y + \frac{1}{2}\, t^2\, y^t\, H f(x^*) y + O(t^2).$$

quand t tend vers 0.

Comme x^*est un minimum local on a :

$$\lim_{t\to 0} \frac{f(x^*+ty)-f(x^*)}{t^2} = \frac{1}{2}\, y^t H f(x^*)\, y + \lim_{t\to 0} \frac{O(t^2)}{t^2} = \frac{1}{2}\, y^t H f(x^*) y \geq 0.$$

Théorème 2.6.2 (condition nécessaire du 2éme ordre) **[14]**

Soit x^* un minimum local pour (P1) on a alors :

$$\begin{cases} \nabla f(x^*) = 0 \\ y^t\, H f(x^*)\, y \geq 0\ ,\ \forall\, y \in D \end{cases}$$

La matrice hessienne au point x^* est donc semi-définie positive. Ces conditions ne sont pas suffisantes.

Théorème 2.6.3 (conditions suffisantes du 2éme ordre) **[14]**

Soit x^* un point tel que :

$$\begin{cases} \nabla f(x^*) = 0 \\ y^t\, H f(x^*) y > 0\ ,\ \forall\, y \in D \end{cases}$$

Alors x^* est un minimum local (strict).

2.7. Conditions d'optimalité : (avec contraintes**)**

Soit le programme mathématique (PM):

$$\begin{cases} Min \quad f(x) \\ g_i(x) \leq 0,\ i = 1,\dots,m \\ h_j(x) = 0,\ j = 1,\dots,p \\ x \in \mathbb{R}^n \end{cases} \qquad \text{(P1)}$$

Où les fonctions f, g_i et h_j sont au moins deux fois continûment différentiables.

Définition 2.7.1: Le lagrangien du programme mathématique (P1) est défini par :

$$L(x;\ \lambda, \mu) = f(x) + \sum_{i=1}^{m} \lambda_i\ g_i\ (x) + \sum_{j=1}^{p} u_j h_j\ (x).$$

Théorème 2.7.1:(Karush-Kuhn-Tucker)[14](avec contraintes)

Soit x^* un minimum local régulier de (PM). Alors, il existe les multiplicateurs $\lambda^* \in \mathbb{R}_+^m$ et $\mu^* \in \mathbb{R}^p$ tels que *:*

$$\begin{cases} \nabla f(x^*) + \sum_{i=1}^{m} \lambda^*{}_i \nabla\ g_i(x^*) + \sum_{j=1}^{p} u^*{}_j \nabla\ h_j(x^*) = 0 \\ Avec \\ (\nabla_x\ L(x^*;\ \lambda^*, \mu^*) = 0);\ (condition\ d'optimalité\ ). \\ \lambda^*{}_i \nabla\ g_i\ (x^*) = 0, (i = 1 \dots m); (condition\ de\ complémentarité). \\ \lambda_i{}^* \geq 0, (\ j = 1 \dots m) \quad (KKT\ ). \\ g_i\ (x^*) \leq 0,\ \ i = 1, \dots, m. \\ h_j\ (x^*) = 0,\ \ j = 1, \dots, p. \end{cases}$$

sont aussi les conditions nécessaires du premier ordre.

<u>Remarques</u>:

1- Si les contraintes ne sont pas qualifiées en x^* , les conditions de KKT ne s'appliquent pas (x^* peut être optimal sans vérifier ces conditions).

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

soit $I(x^*)=\{i/i=1\dots m,\ g_i(x^*)=0\}$, l'ensembles des indices des contraintes d'inégalités actives au point x^* .

Définition 2.7.2:

On définit un espace tangent $T(x^*)$ au point x^* d'un problème (PM) par :

$T(x^*)= \{\ y \in \mathbb{R}^n : \nabla\ g_i\ (x^*)^t y \leq 0, i \in I(x^*)$ et $\nabla\ h_j(x^*)^t y = 0, j = 1...p\ \}$.

Théorème 2.7.2.: (Conditions nécessaires du 2ème ordre)[14]

Soit x^*un minimum local régulier de (PM). Alors, il existe des multiplicateurs $\lambda^* \in \mathbb{R}_+^m$ et $\mu^* \in \mathbb{R}^p$ tels que:

- les conditions de KKT sont satisfaites;
- $y^t\ H\ L(x^*; \lambda^*, \mu^*)\ y \geq 0, \forall\ y \in T\ (x^*)$,(**Hessienne** semi définie positive).

Théorème 2.7.3: (Conditions suffisantes du 2ème ordre)[14]

S'il existe des multiplicateurs $\lambda^* \in \mathbb{R}_+^m$ et $\mu^* \in \mathbb{R}^p$ tels que:

- les conditions de KKT sont satisfaites;
- $y^t H L(x^*; \lambda^*, \mu^*) y > 0, \forall y \in T(x^*)$.

$T(x^*)$ est toujours un espace tangent du problème (PM) au point x^*. Alors le point x^* est un minimum local strict de (PM).

3.1. Introduction :

La programmation linéaire est certainement l'un des plus beaux succès de la recherche opérationnelle. Il provient d'une part de la puissance de modélisation qu'elle offre et ce malgré la limite inhérente qu'impose la linéarité des fonctions impliquées, et d'autre part, de la richesse de la théorie qu'elle a initiée et qui a permis le développement d'algorithmes extrêmement efficaces pour sa résolution.

D'une façon générale, on définit un programme linéaire primal (P) sous la forme standard par :

$$(P) \quad \begin{cases} \text{Min } c^t x \text{ ou } Max \\ sc \\ A\,x = b \\ x \geq 0 \end{cases}$$

Où : A est une matrice réelle de type (m; n) supposée de plein rang (rang A = m ≤ n) , $b \in \mathbb{R}^m$ *et* $c \in \mathbb{R}^n$. L'ensemble des solutions réalisables $D = \{x \in \mathbb{R}^n : A\,x = b; x \geq 0\}$ est un polyèdre convexe fermé.

3.2. Dualité [9]:

A chaque programme linéaire est associé un autre programme linéaire appelé dual. Le dual (DU) correspond au primal standard (P) est :

$$(DU) \quad \begin{cases} Max\ b^t y \\ A^t\, y \leq c \\ y \in \mathbb{R}^m \end{cases}$$

Et Les solutions des programmes primal et dual sont liées par les théorèmes de la dualité faible et forte.

Proposition 3.2.1 (Dualité faible): [9]

Si *x* est primal réalisable et *y* est dual réalisable, alors : $b^t y \leq c^t x$.

Proposition 3.2.2 (Dualité forte) :[9]

Si *x* est fini et optimal pour le programme primal(P), alors le programme dual (DU) a une solution *y* optimale telle que : $b^t y = c^t x$.

3.3. Autres définitions :

Définitions 3.3.1:

1) On dit que x est un point extrême de D, s'il ne peut pas s'écrire comme une combinaison convexe stricte de deux points de D.

$$\nexists x_1, x_2 \in D\ , \lambda\ \epsilon]0.1[\quad \text{pour les quels } (1-\lambda)x_1 + \lambda\, x_2 = x \,/\, x_1 \neq x_2.$$

2) Une base de A est toute sous-matrice A_B (matrice de base) formée de m colonnes linéairement indépendantes de A.

3) Soit A_B une base de A. La solution de base associée à A_B est le point :

$X = (x_B, x_H)^t \in \mathbb{R}^n$ tel que : $X_B = A_B^{-1} b$.(composante de base).

$$X_H = 0.\text{(hors base).}$$

Proposition 3.3.1 [9]:

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

3.4. La résolution d'un programme linéaire (PL):

Dans le cas de programmation linéaire, on dispose d'une méthode efficace de résolution : l'algorithme du simplexe, découvert par Dantzig en 1947.

Cet algorithme a connu depuis lors de nombreuses améliorations, et il est utilisé dans la majorité des logiciels commerciaux. Cependant, un nouveau type de méthodes de résolution a fait son apparition en 1984 : les méthodes des points intérieurs. La plupart des idées proviennent du domaine de la programmation non-linéaire. Parmi leurs avantages, citons :

1- Efficacité théorique : il est possible de prouver que ces méthodes s'exécutent en temps polynomial (ce qui n'est pas le cas de l'algorithme du simplexe, de nature exponentielle ,de moyenne polynomiale).

2-Traitement de très grands problèmes : ces méthodes permettent de résoudre des problèmes de très grande taille qu'aucun autre algorithme connu ne pourrait traiter en un temps acceptable.

On s'intéresse dans ce chapitre à la résolution par la méthode du simplexe et puis par la méthode de Karmarkar.

3.4.1. La résolution des programmes linéaires par la méthode du simplexe :

La méthode du simplexe itère d'une base réalisable à une autre base réalisable adjacente (seuls deux indices de la base sont modifies). Le nombre de base réalisable est clairement fini. Dans le cas non dégénéré ($X_B > 0$), ces deux bases correspondent à deux points extrêmes voisins de valeurs différentes. Dans le cas dégénéré (au moins une composante du vecteur X_B est nulle), les deux bases peuvent correspondre au même point extrême, et donc de même valeur.

3.4.1.1.Algorithme du simplexe [9]:

1-Soit X une solution de base réalisable :

$$A\,\mathrm{x} = b$$

$$\Leftrightarrow \quad A_B\,\mathrm{x}_B + A_H\,\mathrm{x}_H = b$$

$$\Leftrightarrow \quad \mathrm{x}_B = A_B^{-1}\,b - A_B^{-1}A_H\,\mathrm{x}_H$$

Et de valeur:

$$Z = C^t X = c_B^t\,\mathrm{x}_B + c_H^t\,\mathrm{x}_H$$

$$= c_B^t(A_B^{-1}\,b - A_B^{-1}A_H\,\mathrm{x}_H) + c_H^t\,\mathrm{x}_H$$

$$= c_B^t A_B^{-1}\,b + (c_H^t - c_B^t A_B^{-1}A_H)\mathrm{x}_H.$$

De façon compacte, on associe à chaque base un dictionnaire (ou encore tableau) :

$$\begin{array}{l} \mathrm{x}_B = A_B^{-1}\,b \quad - A_B^{-1}A_H\,\mathrm{x}_H \\ \hline Z = c_B^t A_B^{-1}\,b \quad + (c_H^t - c_B^t A_B^{-1}A_H) \end{array}$$

On note:

$\Delta_H = -c_H^t + c_B^t A_B^{-1}A_H$. Le vecteur des couts réduits (des estimations) des variables hors bases et on pose :

$y = A_B^{-1}A_H$ est le vecteur des potentiels. Plutôt on peut dire que y solution du système : $A_B y = c_B^t$.

2. changement de base :

a) Choix de variable d'entrée :

Choisir x_{i*} tel que $\Delta_{i*} = \min_{i \epsilon\, j_H}\{\Delta_i < 0\}$;

S'il n'existe pas un tel i, la solution trouvée est optimale.

b) Choix de variable de sortie :

On cherche une composante x_{j*} du vecteur de base X_B pour la quelle:

$$\theta_{j*} = \text{Min } \frac{(A_B^{-1}b)j}{(A_B^{-1}a^{i*})j} : \{ j \epsilon j_B \text{ et pour } (A_B{}^{-1}a_i)_j > 0 \}$$

S'il n'existe pas un tel j^*, le programme linéaire est non borné.

3. Le pivotage :

(a) Mettre à jour la base: la variable $x_i{}^*$ entre dans la base et la variable$x_j{}^*$ sort de la base.

(b) Obtenir le nouveau dictionnaire par pivotage. Aller en 2.

3.4.1.2. Critère d'optimalité [7]: on dit $X = (x_B, x_H)^t$ est optimale si seulement si :

Le vecteur des estimations $\Delta_H \geq 0$ est positif (cas de maximisation).

3.4.1.3. Exemple numérique :

Soit le problème de maximisation suivant :

$$(P1) \begin{cases} \text{Max } x_1 + 2x_2 \\ \text{S.c} \\ x_1 + x_2 \leq 2 \\ -x_1 + x_2 \leq 1 \\ x_1, x_2 \geq 0 \end{cases}$$

Après l'addition des variables d'écart, on obtient le problème canonique suivant :

$$(P2) \qquad \begin{cases} \text{Max } x_1 + 2x_2 \\ \text{S.c} \\ x_1 + x_2 + x_3 = 2 \\ -x_1 + x_2 + x_4 = 1 \\ x_1, x_2, x_3, x_4 \geq 0 \end{cases}$$

3.4.1.3 La résolution du problème (P2):

1) Dressons le premier tableau de simplexe suivant :

	C	1	2	0	o	
	B	X_1	X_2	X_3	X_4	θ
X_3	2	1	1	1	0	2
X_4	1	-1	1	0	1	1 →
	Δ_j	-1	-2 ↑	0	0	

Soit la solution de base réalisable donnée :

$X^t_0=(x_B, x_H)$ =(0,0,2,1) avec J_B {indice de base}={3,4}et J_H{indice hors base}={1,2}.$\exists\ \Delta_1, \Delta_2 < 0$, donc X_0 n'est pas optimale.

1- la recherche de la variable d'entrée :

on a pour $i^* \in J_H$ ={1,2}
$\Delta_{i*} = \{\Delta_i < 0\} = \Delta_2$, i^* =2 ce qui fait, x_2 va entrer dans la base.

2- la recherche de la variable de sortante de la base :

$\theta_{j*} = \text{Min}\ \frac{(A_B^{-1}b)j}{(A_B^{-1}a^i)j} = \text{Min}\{2,1\} = \theta_4$:{pour j$\epsilon$$j_B$ et $(A_B{}^{-1}a_I)_j$ strictement positifs }
donc x_4 qui va sortir de la base .Et on fait le pivotage.

2)Dressons le deuxième tableau du simplexe :

	C	1	2	0	O	
	B	X_1	X_2	X_3	X_4	θ
X_3	1	2	0	1	-1	1/2 →
X_2	1	-1	1	0	1	/
	Δ_j	-3 ↑	0	0	2	

D'après la première itération, on obtient alors :

X_1=(0,0,1,1) avec Z_1=2 et J_B {indice de base}={2,3}et J_H{indice hors base}={1,4}.

on a $\Delta_1, \Delta_4 < 0$, donc X_1 n'est pas optimale.

On passe à une autre itération :

1-*La recherche de la variable d'entrée :*

on a pour $i^* \in J_H = \{1,4\}$

$\Delta_{i*} = \{\Delta_i < 0\} = \Delta_1$, donc $i^* = 1$ ce qui fait, x_1 va entrer dans la base.

2-La recherche de la variable de sortie :

$\theta_{j*} = \text{Min } \frac{(A_B^{-1}b)j}{(A_B^{-1}a^i)j} = \text{Min}\{1/2,1\} = \theta_3$:{pour $j \epsilon j_B$ et $(A_B{}^{-1}a_i)_j$ strictement positifs } donc x_3 qui va sortir de la base .

Après le pivotage, on trouve le tableau suivant :

3) Dressons le troisième tableau suivant :

	C	1	2	0	O	
solution	B	X_1	X_2	X_3	X_4	θ
x_1	½	1	0	1/2	1/2	/
X_2	3/2	0	1	-1	2	/
	Δ_j	0	0	3/2	1/2	

Δ_3, Δ_4 sont positifs; parfois, on dit que le tableau est optimale, donc $X_2 = (1/2, 3/2, 0, 0)$ est la solution optimale pour le problème(P2) avec une valeur $Z_2 = 7/2$.ce qui nous donne :$X = (1/2, 3/2)$ la solution optimale pour le problème (P1)avec $Z = 7/2$.

3.4.2. La résolution des programmes linéaires par la méthode des points intérieurs :

La question de l'efficacité théorique (complexité) avec laquelle les programmes linéaires peuvent être résolus est restée longtemps ouverte, jusqu'aux travaux khachian (1979) qui a proposé un algorithme polynomial utilisant la méthode dite des ellipsoïde. Néanmoins, en dépit de l'intérêt théorique de cette méthode, son instabilité numérique n'a jamais permis de résoudre en pratique des programmes linéaires de plus d'une ou des centaines de variables.

Le progrès majeur pour concilier l'efficacité théorique (polynomialité) et l'efficacité pratique a été accompli par Karmarkar (1984). La capacité de l'algorithme de Karmarkar (ou de nombreuses variantes développées depuis sous le nom <<méthodes de points intérieurs >>) à résoudre des problèmes de taille considérables (jusqu'à des millions de variables) a été un sujet controverse jusqu'à la fin des années 80, mais elle est aujourd'hui couramment reconnue.

3.4.3. **Méthode de Karmarkar** (méthode de base):

Présentation de la méthode :

En 1984**,** Karmarkar a proposé un algorithme prometteur pour la programmation linéaire, il possède des propriétés théoriques attractives et un bon comportement numérique.

Considérant le problème linéaire suivant :

$$(PLS) \qquad \begin{cases} Min\ c^t x \\ s.c \\ A\,x = 0 \\ \sum_{i=1}^{n} x_i = 1 \\ x \geq 0 \end{cases}$$

Cette forme n'est pas restrictive puisque tout programme linéaire peut se ramener à cette forme (en classe).

Définition 3.4.3.1 : On définit le simplexe S_n de dimension n-1 contenu dans $\mathbb{R}^n$ par :

$$S_n = \{x \in \mathbb{R}^n : x \geq 0,\ e_n^t x = 1\}.$$

- Le problème (PLS) supposé sous les hypothèses suivantes :

Karmarkar a proposé des hypothèses qui doivent être vérifiées pour la résolution du problème (PLS) précédent suivantes :

Hypothèses[3] :(dites hypothèses de Karmarkar)

(Hypothèse 01) : La matrice A est de plein rang (rang A = m ≤ n).

(Hypothèse 02) : On dispose d'un point x_0 strictement réalisable ($Ax_0 = 0$; $x_0 > 0$):

(Hypothèse 03) : 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. Pour l'hypothèse(3), n'est que technique, si la valeur optimale est 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:

$$c^t x^* = z^* = z^* e^t x^* \Rightarrow (c^t - z^* e^t)\, x^* = 0.$$

- L'algorithme démarre de la solution initiale $x^0 = a$, et construit une suite de points intérieurs qui converge vers une solution optimale du problème en un temps polynômial. Dans le but de ramener facilement l'objectif à zéro, on le minimise localement sur une sphère inscrite dans la région réalisable.

Donc, à chaque itération k, l'itéré $x^k > 0$ est ramené au centre de S_n par :

<u>la transformation projective [2]</u> T_k définie par la fonction $T_k(x): \mathbb{R}_+^n \to S_n$:

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

Avec : $T_k(x) = \frac{D_K^{-1} x}{e_n^t D_K^{-1} x} = y, T_k^{-1}(x) = \frac{D_k y}{e_n^t D_k y}$ Où D_k=diag $\{ x^k \}$.

Définition 3.4.3.1:

D_k est la matrice diagonale :

$$D_k = \begin{pmatrix} x_k^1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & x_k^n \end{pmatrix}$$

La transformation T_k applique le simplexe S_n en lui-même, en même temps l'itéré $x^k > 0$ est envoyé au centre de S_n, le transformé du programme linéaire (PLS) est le programme fractionnaire suivant :

$$(PNL) \qquad \begin{cases} \text{Min } \frac{c^t D_k y}{e_n^t D_k y} \\ \frac{A D_k y}{e_n^t D_k y} = 0 \\ e_n^t y = 1 \\ y \geq 0 \end{cases} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(1)$$

Les hypothèses (Hyp2) et (Hyp3) de Karmarkar permettent d'écrire le programme (1) non linéaire comme le programme linéaire suivant :

$$(PLS) \quad \begin{cases} \text{Min } c^t D_k y \\ A D_k y = 0 \\ e_n^t y = 1 \\ y \geq 0 \end{cases} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(2)$$

3.4.3.2. Le calcul de direction et de pas :

On a besoin du lemme pour simplifier la résolution de ce dernier problème(2).

Lemme 3.4.3.1 [4]: 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 :

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

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

Preuve: Supposons le contraire, c'est à dire : il existe j ∈ {1 ... n + 1} tel que: y ≤ 0

alors : $\sum_{i=1}^{n+1} \frac{(y_i - y^0{}_i)^2}{(y^0{}_i)^2} \geq \frac{(y_j - y^0{}_j)^2}{(y^0{}_j)^2} \geq 1 \geq \beta^2, 0 < \beta < 1.$

- D'après le lemme (1), si on ajoute au problème (3) la contrainte : $y \in \mathbb{R}^n / \|y - \alpha\| \leq \alpha r$, Avec : 0< α <1 et $r = \frac{1}{\sqrt{n(n-1)}}$ .Alors la contrainte de positivité y≥0 devient redondante. On la remplace et on obtient le problème :

$$(PLS) \qquad \begin{cases} \text{Min } c^t D_k\, y \\ \text{A } D_k\, y = 0 \\ e_n^t y = 1 \\ \|y - a\|^2 \le (\alpha r)^2 \end{cases} \quad \dots\dots\dots\dots\dots\dots\dots\dots (3)$$

Définition 3.4.3.3: On définit la sphère par :

$$S(a, \alpha r) = \{\, x \in \mathbb{R}^n,\ e_n^t x = 1,\ \|y - a\| \le \alpha r \ \}$$

Remarque: Le problème (3), c'est la minimisation d'une fonction linéaire sur une sphère de rayon αr (inscrite $\frac{\alpha}{\sqrt{n(n-1)}}$)centrée au centre du simplexe a.

Théorème 3.4.3.1:[2] La solution optimale du problème (3) est donnée explicitement par : $y^k = a - \alpha r d^k$ Où $d^k = \frac{p^k}{\|p^k\|}$

$$avec : \quad p^k = pB_k(D_k c),\ B_k = \begin{pmatrix} A_k \\ e_n^t \end{pmatrix}.$$

- A chaque itération, on revient à la variable initiale x_0 en appliquant la transformation inverse T_k^{-1} et ainsi de suite jusqu'a ce que le test d'optimalité ($c^t x \le \varepsilon$) se réalise où $\varepsilon > 0$ est une précision donnée.

3.5. Algorithme de base[2] :

Début algorithme :

1-Initialisation : $\varepsilon > 0$ est une précision donnée, $x^0 = a = \frac{1}{n} e_n$, $k = 0$.

2-Itération : $k = 1,2, \dots$

Pas 1 :

Construire : $D_k = \text{diag}\{x^k\}$, $A_k = AD_k$, $B_k = \begin{pmatrix} A_k \\ e_n^t \end{pmatrix}$.

Calculer : $p_k = (I - B_k^t (B_k B_k^t)^{-1} B_k) D_k c$, $d^k = \frac{p^k}{\|p^k\|}$.

Calculer : $y^k = a - \alpha r d^k$ avec : $0 < \alpha < 1$ *et* $r = \frac{1}{\sqrt{n(n-1)}}$.

Pas 2 :

Prendre : $x_{k+1} = T_k^{-1}(y^k) = \frac{D_k y^k}{e_n^t D_k y^k}$.

La condition d'arrêt : si ($c^t x^{k+1} \le \varepsilon$) alors la solution trouvée x_{k+1} est optimale

Sinon : aller en 02 avec $k := k + 1$ et retourner au pas 1.

Fin algorithme.

3.5.1.Convergence de l'algorithme[18]: on définit le potentiel de Karmarkar comme suit : $p(\mathrm{x}) = \sum_{j=1}^{n} \ln(\frac{c'\mathrm{x})}{\mathrm{x}_j})$. Le potentiel de Karmarkar intervient ici à la deuxième étape(condition d'arrêt). En effet Karmarkar montre que si le potentiel évolué en x^k assez petit, alors $z = c'\mathrm{x}$ sera arbitrairement proche de 0.

Dans le théorème suivant, Karmarkar montre que la convergence de son algorithme est réalisée en O(nq + n ln n) itérations pour :$0 < \alpha \leq 1/4$.

3.5.1. Théorème [2]: Si $0 < \alpha \leq 1/4$; alors en partant de $x_0 = \frac{e_n}{n}$; l'algorithme trouve après O(nq + n ln n)itérations un point réalisable x tel que :

1)- $c^t x = 0$.

2)-$\frac{c^t x}{c^t x_0} \leq \varepsilon = 2^{-q}$ où q est une précision fixée.

Remarque :

Dans ce que précède, on a parlé de la résolution des programme linéaire par Karmarkar de la classe de (PLS), dont les quels la valeur optimale du problème z^* est nulle, et le membre à droite des contraintes du problème b est nul, ce qui nous pousse à généraliser cette dernière sur la programmation linéaire dont la valeur optimale z^* et le membre a gauche des contrainte b quelconques.

3.6.Généralisation de l'algorithme de Karmarkar sur la programmation linéaire :

Soit le programme linéaire sous forme standard :

$$(P)\begin{cases} Min\ c^t x = z^* \\ Ax = b \\ x \geq 0 \end{cases} \quad \ldots\ldots\ldots\ldots\ldots (1)$$

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

Définition 3.6.1 :

On définit le simplexe S_{n+1} de dimension n contenu dans $\mathbb{R}^n$ par :

$$S_{n+1} = \{x \epsilon \mathbb{R}^{n+1} : x \geq 0, e_{n+1}^t x = 1\}$$

On suppose les hypothèses suivantes (hypothèses de Karmarkar):

(Hypothese 01): La matrice A est de plein rang ($\mathrm{rg\ A = m \leq n}$)

(Hypothese 02): On dispose d'un point x^0 strictement réalisable $(Ax^0 = b, x^0 > 0)$.

(Hypothese 03): La valeur optimale z^*de l'objectif est connue au départ.

Remarque :

A l'hypothèse 2, pour le système de contraintes : $Ax = b, b \neq 0$, on se ramène facilement à un système homogène .il suffit d'écrire :

$$Ax = be_{n+1}^t x \Rightarrow (A - be_{n+1}^t)x = 0$$

De plus pour : z^* la valeur optimale de l'objectif. Alors, on peut trouver le système précédent : $c^t x^* = z^* = z^* e^t x^* \Rightarrow (c^t - z^* e^t) x^* = \hat{c}x = 0$.

Soient la transformation projective notée $T_k(x)$ une fonction [11][3]:

$T_k(x): \mathbb{R}_+^n \to S_{n+1}$ définie par : $T_k(x) = y$, Avec :

$$\begin{cases} y_i = \frac{x_i/x_i^k}{1+\sum_{i=1}^n x_i/x_i^k} \\ y_{n+1} = 1 - \sum_{i=1}^n y_i \end{cases} \quad \text{.....(2)}$$

On a :

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

Ou encore : $y[n] = (D_k^{-1}x)y_{n+1}$,Où $y[n]$ désigne les n première composantes de y .

Remarque : La transformation $T_k(x)$ est univoque, et donnant sa transformation inverse par : $x = T_k^{-1}(y) = \frac{D_k y[n]}{y_{n+1}}, D_k = diag(x^k)$.

- On applique la transformation $T_k(x)$ au programme(1), on obtient :

$$\begin{cases} min \frac{c^t D_k y[n]}{y_{n+1}} = z^* \\ A\frac{D_k y[n]}{y_{n+1}} = b \\ \sum_{i=1}^{n+1} y_i = 1 \\ y[n] \geq 0, y_{n+1} > 0 \end{cases} \quad \text{.....(3)}$$

Son problème équivalent :

$$\begin{cases} \text{Min } c^t D_k y[n] - z^* y_{n+1} = 0 \\ AD_k y[n] - b y_{n+1} = 0 \\ \sum_{i=1}^{n+1} y_i = 1 \\ y[n] \geq 0, y_{n+1} > 0 \end{cases} \quad \text{.....(4)}$$

Ou encore :

$$\begin{cases} Min\ \hat{c}^t y \\ \hat{A}y = 0 \\ e_{n+1}^t y = 1 \\ y \geq 0 \end{cases} \text{.....(5)}$$

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

Notons que toute solution réalisable de (1) est transformée par T_k en une solution réalisable de (5) et réciproquement, toute solution réalisable y de (5) avec $y_{n+1} > 0$ est transformée par T_k^{-1} en une solution réalisable de x de (1).

3.7. Tests numériques :

Nos tests d'applications sot réalisés su un PC portable Acer sous Windows 7 en utilisant le langage MATLAB.

Les exemples traités sont de la forme canonique suivante :

$$(PL) \qquad \begin{cases} Min\ c^t\ x \\ sc \\ A\ x = b \\ x \geq 0 \end{cases}$$

La précision est comprise entre $10^{-3}\ et\ 10^{-6}$.

Exemple 01 : Soit le programme linéaire :

$$\begin{cases} \text{Min} \quad -4x_1 - 5x_2 \\ sc \\ 2x_1 + x_2 \leq 8 \\ x_1 + 2x_2 \leq 7 \\ x_2 \leq 3 \\ x_1, x_2 \geq 0 \end{cases}$$

Après 85 itérations, on obtient la solution optimale $x^* = (3\ ,\ 2)^t$ qui donne la valeur optimale du problème $z^* = -22$.

Exemple 02 : Soit le problème linéaire suivant :

$$\begin{cases} Min\ 3x_1 - x_2 + x_3 \\ 2x_1 + x_2 - x_4 \leq 0 \\ x_3 + x_5 - x_6 \leq 0 \\ x_1 + x_2 + x_3 + x_4 + x_5 + x_6 \leq 1 \\ x_1, x_2, x_3, x_4, x_5, x_6 \geq 0 \end{cases}$$

Après 208 itérations, on obtient la solution optimale suivante :

$x^* = (0\ ,\ 0.5\ ,\ 0\ ,\ 0.5\ ,\ 0\ ,\ 0)$, avec la valeur optimale : $z^* = -0.5$.

Exemple 03: Soit le problème linéaire suivant :

$$\begin{cases} Min\ x_1 + x_2 \\ sc \\ 2x_1 + x_2 + x_3 = 2 \\ -2x_1 + 4x_2 + x_4 = 0 \\ x_1, x_2, x_3, x_4 \geq 0 \end{cases}$$

On obtient, après 43 itérations, la solution optimale suivante: $x^* = (0.0000, 0.0000, 2.0000, 0.0000)$, avec : $z^* = 0$. La valeur optimale du problème.

Exemple 04 : Soit le problème linéaire suivant :

$$\begin{cases} Min\ 3x_1 + 2x_2 + x_3 + 3x_4 \\ sc \\ x_1 - x_2 + x_3 + x_4 = 3 \\ 2x_1 + x_2 - x_3 + x_4 = 4 \\ x_1 + x_2 + x_4 + x_5 = 5 \\ x_1, x_2, x_4, x_5 \geq 0 \end{cases}$$

On obtient, après 97 itérations la solution optimale suivante :

$x^* = (2.3333 \quad 0.0000 \quad 0.6667 \quad 0.0000 \quad 2.6667)$, avec : $z^* = 7.6667$.

Exemple 05 : Soit le problème linéaire suivant :

$$\begin{cases} Min\ 4x_1 + x_2 + 2x_3 \\ sc \\ 2x_1 + 3x_2 + x_3 + 2x_4 = 2 \\ 3x_1 - 2x_3 + x_4 = 0 \\ x_1, x_2, x_3, x_4 \geq 0 \end{cases}$$

La solution optimale trouvée est : $x^* = (0.0268 \quad 0.3679 \quad 0.0564 \quad 0.6553)^t$, obtenue en 8 itérations, et donne $z^* = 0.6666$.

Exemple 06 : Soit le problème linéaire suivant :

$$\begin{cases} Min\ -x_1 - x_2 \\ sc \\ -2x_1 + x_2 \leq 1 \\ x_1 + x_2 \leq 3 \\ x_1, x_2 \geq 0 \end{cases}$$

La solution optimale trouvée est :

$x^* = (1.3884, 1.6112)^t$, obtenue en 17 itérations, et donne $z^* = -3$.

Commentaire :

A travers les tests numériques effectuées, on remarque l'efficacité pratique de l' algorithme en termes d'infériorité du nombre pratique du nombre d'itération qui est très réduit, et la réduction de la durée d'exécution.

4.1. Introduction :

La programmation quadratique est connue pour ses applications multiples dans plusieurs domaines. son importance provient du fait que plusieurs problèmes réels sont quadratiques c'est le cas de nombreux problèmes en particuliers ceux de la physique et de l'économie…. et souvent, qui intervient comme procédures intermédiaires pour des programmes non linéaires, c'est le cas entre autre des méthodes de programmation quadratique successives (SQP). En ramenant des programmes non linéaires avec de transformation ou formulation adéquate aux problèmes de programmations quadratiques successives où la fonction objectif à minimiser est quadratique sous des contraintes linéaire.

Le problème quadratique en général peut s'écrire comme suit :

$$(PQ) \qquad \begin{cases} Min\ F(x) = \left(\frac{1}{2}x^t Qx + c^t x\right) \\ Ax = b \\ x \geq 0 \end{cases}$$

Où Q est une matrice symétrique d'ordre *n* ($Q \in \mathbb{R}^{n*n}$), $b \in \mathbb{R}^m$, $c, x \in \mathbb{R}^n$ et $A \in \mathbb{R}^{m*n}$ de plein rang *(rgA=m ≤ n)*.

Remarque : 1) L'ensemble des contraintes $D=\{x \in \mathbb{R}^n, Ax = b, x \geq 0\}$ est un polyèdre convexe et fermé, la fonction objectif est infiniment différentiable.

2) (PQ) est convexe si et seulement si *F* est convexe auquel cas la matrice Q est semi-définie positive, car les contraintes sont linéaires (affines), donc elles sont convexes.

4.2. Méthodes de résolution d'un problème quadratique:

On s'intéresse dans ce chapitre à la résolution par complémentarité linéaire qui génère le domaine de la programmation linéaire et le domaine de la programmation quadratique convexe avec une méthode des points intérieurs de type barrière logarithmique, trajectoire centrale avec ou sans poids, en définissant leurs propriétés attractives, notamment la complexité polynomiale et la convergence super linéaire, ce qui devrait donner lieu à comportement numérique prometteur.

4.3. Complémentarité linéaire monotone :

L'importance de la complémentarité peut être mesurée par le rôle crucial qu'elle joue dans la résolution de plusieurs problèmes dans différents domaines: programmation linéaire, programmation quadratique convexe ....

Le problème de complémentarité linéaire monotone s'énonce comme suit :

Déterminer les points $x, y \in \mathbb{R}^n$ tels que :

$$\text{(PLC)} \qquad \begin{cases} y = Mx + q \\ x^t y = 0 \\ (x, y) \geq 0 \end{cases}$$

Où : M est une matrice carré d'ordre n semi-définie positive ($M \in \mathbb{R}^{n*n}$) et $q \in \mathbb{R}^n$.

4.4. Transformation d'un programme quadratique convexe en un programme complémentaire linéaire :

Soit le programme quadratique convexe :

$$\text{(PQ)} \qquad \begin{cases} Min\ F(x) = \left(\frac{1}{2} x^t Q x + c^t x\right) \\ Ax \leq b \\ x \geq 0 \end{cases}$$

Q est une matrice symétrique d'ordre *n*, et semi-définie positive. $b \in \mathbb{R}^m$, $c, x \in \mathbb{R}^n$ et $A \in \mathbb{R}^{m*n}$ de plein rang *(rang A=m≤ n).*

Comme le problème (PQC) est convexe et les contraintes sont linéaires alors les conditions de ***KKT*** sont nécessaires et suffisantes et s'écrivent comme suit :

$x \in \mathbb{R}^n$ est une solution optimale de (PQC) si et seulement si $\exists y \in \mathbb{R}^m_+, \lambda \in \mathbb{R}^n_+$ tels que [4] [3] :

$$\begin{cases} C + Qx + A^t y - \lambda = 0 \\ y^t(b - Ax) = 0 \\ -\lambda^t x = 0 \\ y \geq 0, \lambda \geq 0, x \geq 0. \end{cases}$$

$$\Leftrightarrow \qquad \begin{cases} \lambda = C + Qx + A^t y \\ \mu = b - Ax \\ \lambda^t x = 0, y^t \mu = 0 \\ x \geq 0, y \geq 0, \lambda \geq 0, \mu \geq 0 \end{cases}$$

$$\Leftrightarrow \quad \begin{cases} \begin{pmatrix}\lambda\\ \mu\end{pmatrix} = \begin{pmatrix} Q & A^t \\ -A & 0\end{pmatrix}\begin{pmatrix}x\\ y\end{pmatrix} + \begin{pmatrix}c\\ b\end{pmatrix} \\ \left\langle \begin{pmatrix}\lambda\\ \mu\end{pmatrix} \middle| \begin{pmatrix}x\\ y\end{pmatrix} \right\rangle = 0 \\ \begin{pmatrix}\lambda\\ \mu\end{pmatrix} \geq 0, \begin{pmatrix}x\\ y\end{pmatrix} \geq 0 \end{cases}$$

En posant : $\omega = \begin{pmatrix}\lambda\\ \mu\end{pmatrix}, z = \begin{pmatrix}x\\ y\end{pmatrix}$, $q = \begin{pmatrix}c\\ b\end{pmatrix}$

$$M = \begin{pmatrix} Q & A^t \\ -A & 0\end{pmatrix}$$

On obtient bien un PCL :

$$\begin{cases} \text{Trouver } z \in \mathbb{R}^{m+n} \text{ tel que :} \\ \omega = Mz + q \geq 0, z \geq 0 \\ \omega^t z = 0 \end{cases}$$

Exemple : soit le problème quadratique suivant :

$$\begin{cases} Min\ x_1^2 + x_2^2 + 2x_1 + 3x_2 \\ sc \\ 5x_1 + 3x_2 <= 4 \\ x_1 - x_2 <= 5 \\ (x_1, x_2) \geq 0 \end{cases} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(1)$$

Qui est équivalent au problème suivant :

$$\begin{cases} Min\ (x_1, x_2)\begin{pmatrix}2 & 0\\ 0 & 2\end{pmatrix}\begin{pmatrix}x_1\\ x_2\end{pmatrix} + (2,3)\begin{pmatrix}x_1\\ x_2\end{pmatrix} \\ sc \\ 5x_1 + 3x_2 \leq 4 \\ x_1 - x_2 \leq 5 \\ (x_1, x_2) \geq 0 \end{cases}$$

Son problème équivalent en programmation de complémentarité linéaire est le suivant :

$$\begin{cases} \text{Trouver } z \in \mathbb{R}^{m+n} \text{ tel que :} \\ \omega = Mz + q \geq 0, z \geq 0 \\ \omega^t z = 0 \end{cases}$$

Avec :

$$\omega = \begin{pmatrix}\lambda\\ \mu\end{pmatrix}, z = \begin{pmatrix}x\\ y\end{pmatrix} \text{ et } q = \begin{pmatrix}2\\ 3\\ 4\\ 5\end{pmatrix} \text{ avec : } M = \begin{pmatrix} 1 & 0 & 5 & 1 \\ 0 & 1 & 3 & -1 \\ -5 & -3 & 0 & 0 \\ -1 & 1 & 0 & 0 \end{pmatrix}.$$

4.4.1. Dans le cas du programme linéaire :

On a un programme linéaire sous la forme :

$$(PL) \qquad \begin{cases} \text{Min } c^t x \\ sc \\ A\,x \leq b \\ x \geq 0 \end{cases}$$

Peut se transformer à un (PLC) de la forme :

$$\begin{cases} \text{Trouver } z \in \mathbb{R}^{m+n} \text{ tel que :} \\ \omega = Mz + q \geq 0, z \geq 0 \\ \omega^t z = 0 \end{cases}$$

Où :

$$\omega = \begin{pmatrix} \lambda \\ \mu \end{pmatrix}, z = \begin{pmatrix} x \\ y \end{pmatrix}, q = \begin{pmatrix} c \\ b \end{pmatrix} \text{ et}$$

$$M = \begin{pmatrix} 0 & A^t \\ -A & 0 \end{pmatrix}.$$

Exemple : soit le problème linéaire suivant :

$$\begin{cases} Min\ (2,3) \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \\ sc \\ 5x_1 + 3x_2 <= 4 \\ x_1 - x_2 <= 5 \\ (x_1, x_2) \geq 0 \end{cases}$$

Son problème de complémentarité linéaire suivant :

$$\begin{cases} \text{Trouver } z \in \mathbb{R}^{m+n} \text{ tel que :} \\ \omega = Mz + q \geq 0, z \geq 0 \\ \omega^t z = 0 \end{cases}$$

Avec :

$$\omega = \begin{pmatrix} \lambda \\ \mu \end{pmatrix}, z = \begin{pmatrix} x \\ y \end{pmatrix} \text{ et } q = \begin{pmatrix} 2 \\ 3 \\ 4 \\ 5 \end{pmatrix} \text{ et } M = \begin{pmatrix} 0 & 0 & 5 & 1 \\ 0 & 0 & 3 & -1 \\ -5 & -3 & 0 & 0 \\ -1 & 1 & 0 & 0 \end{pmatrix}.$$

4.5. Transformation d'un programme complémentaire linéaire monotone en un programme quadratique convexe :

En général, on ne peut pas transformer un (PCL) quelconque en (PQC) sauf si la matrice M est semi-définie positive auquel cas on parle de (PCL) monotone.

Soit le problème complémentaire linéaire monotone suivant :

$$\begin{cases} \text{Trouver } z \in \mathbb{R}^{m+n} \text{ tel que :} \\ \omega = Mz + q \geq 0, z \geq 0 \\ \omega^t z = 0 \end{cases}$$

4.5.1. Théorème [3]:

Le (PCL) est équivalent au programme quadratique convexe suivant :

$$(PL) \qquad \begin{cases} Min\ z^t(Mz+q) \\ sc \\ Mz+q \geq 0 \\ z \geq 0 \end{cases}$$

Au sens que (z^*, Mz^*+q) est une solution de (PCL) si et seulement si z^* est une solution optimale du programme quadratique avec une valeur optimale nulle de l'objectif. En effet un programme quadratique convexe est un cas particulier du problème de complémentarité.

4.6. La résolution d'un problème de complémentarité linéaire par la méthode de trajectoire centrale :

Présentation de la méthode :

Cette approche est proposée par Kojima,Misuno et Yoshise, c'est une extension de la méthode de trajectoire centrale appliquée à la programmation linéaire.

Soit le problème de complémentarité linéaire monotone suivant :

$$(PCL) \qquad \begin{cases} Trouver\ x \in \mathbb{R}^n tel\ que : \\ s.c \\ y = Mx + q \\ x^t y = 0 \\ (x,y) \geq 0 \end{cases}$$

Notons :

$D_S = \{(x,y) \in D : y = Mx + q\}$ l'ensemble des solutions strictement réalisables de (PCL).

$D_{PCL} = \{(x,y) \in D_S : x^t y = 0\}$ L'ensemble des solutions de(PCL).

$e = (1, \dots\dots\dots 1)^t \in \mathbb{R}^n$.

$X = diag(x) : x > 0, Xe = x$.

$Y = diag(y) : y > 0, Ye = y$.

Supposons les hypothèses suivantes :

$(Hypothèse\ 01): D_S \neq \emptyset$ C'est-à-dire, un point strictement réalisable existe.

$(Hypothèse\ 02): M$ est semi -définie positive.

Le (PCL)monotone est équivalent au programme quadratique convexe suivant :

$$\text{(PQC)} \qquad \begin{cases} min\ x^t y \\ s.c \\ y = Mx + q \\ (x, y) \geq 0 \end{cases}$$

On associe à ce dernier, le problème pénalisé suivant :

$$(PCL)_\mu \begin{cases} min f_\mu(x, y) \\ s.c \\ y = Mx + q \\ (x, y) > 0 \end{cases}$$

Où $f_\mu(x, y)$ est la fonction pénalisée définie par :

$$f_\mu(x, y) = x^t y - \mu \sum_{i=1}^{n} \ln (x_i y_i)$$

Et μ un paramètre barrière strictement positif.

L'idée générale des méthodes de trajectoire centrale consiste à suivre un chemin particulier (dit des centres ou chemin central), en prenant comme direction de déplacement celle de Newton. Autrement dit : l'algorithme génère une suite (x^k, y^k) strictement réalisable et une suite μ^k qui exprime la condition de complémentarité. Cette dernière est vérifiée lorsque μ^k tend vers zéro.

Proposition 4.6.1.1 :

La fonction $f_\mu(x, y)$ est strictement convexe.

Preuve :

La fonction $f_\mu(x, y)$ peut être écrite sous la forme :

$$f_\mu(x) = x^t(Mx + q) - \mu \left[\sum_{i=1}^{n} \ln (x_i) + \sum_{i=1}^{n} \ln(Mx + q)_i \right]$$

En effet $f_\mu(x) \in C^\infty$ et nous avons en particulier :

$\nabla f_\mu(x) = (M + M^t)x + q - \mu X^{-1} e - \mu M^t diag((Mx + q)_1, \dots\dots, (Mx + q)_n)^{-1} e$

$\nabla^2 f_\mu(x) = M + M^t + \mu[X^{-2} + M^t diag((Mx+q)_1, \dots\dots, (Mx+q)_n)^{-2} M]$

Comme M est une matrice semi-définie positive,X^{-2}est définie positive pour tout $x > Oet\ \mu > o$ alors $\nabla^2 f_\mu(x)$ est une matrice définie positive, donc $f_\mu(x)$ est convexe.

Théorème 4.6.1.1: [5]

M Étant semi-définie positive et $D_S \neq \emptyset$ non vide, le problème $(PCL)_\mu$ admet une solution optimale unique pour tout $\mu > o$.

Propriétés de $f_\mu(x)$:1- Si $D_S \neq \emptyset$ est non vide, alors pour tout $\mu > o$ le problème $(PCL)_\mu$admet une solution optimale unique qui est notée (x, y) appelé point central.

2- Quand $\mu \to 0$, $f_\mu(x^*, y^*)$ valeur optimale de (PCL).

Théorème 4.6.1.2:[3]

Soit $\mu > 0$ et M une matrice semi définie positive.

(x, y) est une solution optimale de $(PCL)_\mu$ si et seulement si (x, y) satisfait le système suivant :

$$\begin{cases} XYe - \mu e = 0 \\ y = Mx + q \\ (x, y) > 0 \end{cases} \quad \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots(1)$$

Donc, la solution de $(PCL)_\mu$ est équivalente à résoudre le système(1).

Preuve :

$(PCL)_\mu$ est convexe et différentiable, les contraintes sont qualifiées car affines, alors les conditions de KKT sont nécessaires et suffisantes et s'écrivent comme suit :

$$\begin{cases} y - \mu X^{-1} e + M^t z = 0 \\ x - \mu Y^{-1} e - z = 0 \\ y = Mx + q \\ (x, y) > 0 \end{cases}$$

Où $z \in \mathbb{R}^n$ est le multiplicateur de Lagrange associé à la contrainte $Mx + q - y = 0$ du problème $(PCL)_\mu$.

Ou d'une manière équivalente :

$$\begin{cases} (XM^t + Y)z = 0 \\ y = Mx + q \\ (x, y) > 0 \end{cases}$$

$(XM^t + Y)$ est définie positive alors inversible, donc $z = 0$.On substitut dans le système de KKT, on obtient :

$$\begin{cases} y - \mu X^{-1}e = 0 \\ x - \mu Y^{-1}e = 0 \\ y = Mx + q \\ (x, y) > 0 \end{cases}$$

Alors :

$$\begin{cases} XYe - \mu e = 0 \\ y = Mx + q \\ (x, y) > 0 \end{cases}$$

Définition 4.6.1.1:

En désignant par (x_μ, y_μ) la solution du système non linéaire (1) pour $\mu > 0$ donné.L'ensemble de toutes les solutions du système (1): $T = \{(x_\mu, y_\mu) : \mu > 0\}$ est appelé trajectoire centrale.

Résolution du système (1) :

Sur la fonction issue du système (1) :

$$F_\mu(x, y) = 0 \; et \; (x, y, \mu) \in \mathbb{R}^n_+ * \mathbb{R}^n_+ * \mathbb{R}_+ \dots\dots\dots\dots\dots (2)$$

Où $F_\mu : \mathbb{R}^n_+ * \mathbb{R}^n_+ \to \mathbb{R}^n_+ * \mathbb{R}^n_+$ est une fonction définie par :

$$F_\mu(x, y) = \begin{bmatrix} XYe - \mu e \\ y - Mx - q \end{bmatrix}$$

L'itération de newton est définie par :

$$(x^+, y^+) = (x, y) \; + \; (\Delta x, \Delta y).$$

Où $(\Delta x, \Delta y)$ est solution du système linéaire :

$$\nabla F_\mu(x, y) \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} = -F_\mu(x, y) \dots\dots\dots\dots\dots \text{(sl)}$$

C'est-à-dire le système d'équations linéaires suivant :

$$\begin{cases} Y\Delta x + X\Delta y = -\mu e - XYe \\ \Delta y = M\Delta x \end{cases}$$

(sl) s'écrit sous la forme matricielle comme suit :

$$\begin{pmatrix} Y & X \\ M & -I \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} = \begin{pmatrix} -\mu e - XYe \\ 0 \end{pmatrix}$$

Par un simple calcul, on trouve la solution $(\Delta x, \Delta y)$ donnée par :

$$\begin{cases} \Delta x = (M + X^{-1}Y)(\mu X^{-1}e - Ye) \\ \Delta y = M\Delta x \end{cases}$$

4.6.1.3. Facteur de centralité :

La qualité de chaque solution trouvée est mesurée par un facteur dit de centralité :

$$\delta(x, y, \mu) = min\|XYe - \mu e\| = \left\|XYe - \left(\frac{x^t y}{n}\right)e\right\|$$

$(x, y) \in D_s$ est un point de la trajectoire centrale si : $\left\|XYe - \left(\frac{x^t y}{n}\right)e\right\| = 0.$

4.6.1.4. Concept de proximité:

Le point (x, y) est à proximité de la trajectoire centrale $T(\theta)$ s'il appartient à l'ensemble suivant :

$$T(\theta) = \left\{(x, y) \in D_S : \left\|XYe - \left(\frac{x^t y}{n}\right)e\right\| \leq \left(\frac{x^t y}{n}\right)\theta : \theta > 0\right\}$$

L'algorithme correspondant se présente comme suit :

4.6.1.5.Algorithme de la trajectoire centrale pour la programmation quadratique :[3]

Début algorithmique :

Données :$\varepsilon > 0$ un parametre de precision et $0 < \theta \leq 0.1$ une constante donnée.

1-Initialisation : $(x^0, y^0) \in T(\theta)$

2-Itération : pour $k = 0,1,2\ldots\ldots..$

Calculer :

-les paramètres : $\delta = \frac{\theta}{1-\theta}$ et $\mu^k = (1 - \frac{\delta}{\sqrt{n}})\frac{(x^k)^t y^k}{n}$

-la direction de Newton $(\Delta x^k, \Delta y^k)$ solution du système linéaire :

$$\begin{cases} Y^k\Delta x^k + X^k\Delta y^k = \mu^k e - X^k Y^k e \\ \Delta y^k = M\Delta x^k \end{cases}$$

-le nouvel itéré : $(x^{k+1}, y^{k+1}) = (x^k, y^k) + (\Delta x^k, \Delta y^k)\,, k = k + 1$

Condition d'arrêt: Si $(x^k)^t y^k \leq \varepsilon$ alors la solution trouvée x^k est optimale

Sinon aller en 02 ;

Fin de l'algorithme.

L'algorithme part d'une solution initiale $(x^0, y^0) \in D_s$ dans le voisinage de la trajectoire centrale avec $\mu = \mu^0$, génère une suite de point (x^k, y^k) solution de système (2) par la méthode de Newton. Et on s'arrête lorsqu'on obtient une valeur suffisamment petite de paramètre μ. Pour des choix convenables des constantes positives δ et θ.du plus, l'algorithme génère une suite $(x^k, y^k) \in T(\theta)$ (le nouveau point reste dans le voisinage de la trajectoire centrale) en partant d'un point initial $(x^0, y^0) \in T(\theta)$, et à chaque itération le terme $(x^k)^t y^k$ se reduit au moins linéairement par un montant fixe $\left(1 - \frac{\delta}{6\sqrt{n}}\right)$.

4.6.1.6. La convergence de la méthode: [3] a été établie par Z.KEBICH

Si le point courant (x^k, y^k) est voisin de la trajectoire centrale et si on choisit une valeur convenable du paramètre $\mu > 0$; alors le nouveau point(x^+, y^+) reste voisin de la trajectoire centrale, plus précisément, on a le théorème suivant :

Théorème 4.6.1.6.1 : $0 < \theta \leq 0.1$ $\delta = \frac{\theta}{1-\theta}$ et supposons que :

$(x^k, y^k) \in T(\theta)$ et $\mu^k = \left(1 - \frac{\delta}{\sqrt{n}}\right)\frac{(x^k)^t y^k}{n}$, alors le point (x^{k+1}, y^{k+1}) défini par :

$$(x^{k+1}, y^{k+1}) = (x^k, y^k) + (\Delta x^k, \Delta y^k) \quad \text{Satisfait :}$$

1- $(x^{k+1}, y^{k+1}) \in T(\theta)$

2- $(x^{k+1})^t y^{k+1} \leq \left(1 - \frac{\delta}{6\sqrt{n}}\right)\frac{(x^k)^t y^k}{n}$.

Remarque : Théoriquement, on suppose que la solution initiale est strictement réalisable qui soit proche de la trajectoire centrale, mais l'obtention de ce point est une difficulté majeure. Nous présentons une autre alternative à la méthode de trajectoire centrale avec poids , c'est dessus.

4.6.2. Méthode de trajectoire centrale avec poids :

Présentation de la méthode :

On associe au problème (PCL) le problème perturbé défini comme suit :

$$(PCL)_{\mu r} \begin{cases} minf_\mu(x, y) = [x^t y - \mu \sum_{i=1}^n r_i \ln(x_i y_i)], \mu > 0 \\ y = Mx + q \\ (x, y) > 0 \end{cases}$$

Où $r = (r_1, r_2, \dots \dots, r_n) \in \mathbb{R}^n_{++}$ sont les poids associés à la fonction barrière logarithmique.

Notons :$R = diag(r)$. la matrice diagonale dont ces éléments diagonaux sont les $r_i, i = 1 \dots n$.

Remarques : 1) pour $r = (1,1, \dots \dots 1) \in \mathbb{R}^n$, on trouve le système classique, c'est-à-dire la méthode trajectoire centrale (sans poids).

2) la solution optimale finie de $(PCL)_{\mu r}$converge vers la solution de (PCL) lorsque μ tend vers 0.

- $f_\mu(x, y)$ est strictement convexe et les contraintes sont linéaire (affines), alors les conditions de KKT correspondantes sont linéaires et suffisantes et s'écrivent comme suit :

$$\begin{cases} y - \mu X^{-1} r + M^t z = 0 \\ x - \mu Y^{-1} r - z = 0 \\ y = Mx + q \\ (x, y) > 0 \end{cases}$$

Où $z \in \mathbb{R}^n$ est le multiplicateur de Lagrange associé à la contrainte ($Mx + q - y = 0$) du problème $(PCL)_{\mu r}$ .Ou d'une manière équivalente :

$$\begin{cases} (XM^t + Y)z = 0 \\ y = Mx + q \\ (x, y) > 0 \end{cases}$$

$(XM^t + Y)$ est définie positive alors inversible, donc $z = 0$ .on substitut dans le système de KKT , on obtient :

$$\begin{cases} y - \mu X^{-1} r = 0 \\ x - \mu Y^{-1} r = 0 \\ y = Mx + q \\ (x, y) > 0 \end{cases}$$

Qui donne, alors :

$$(**) \begin{cases} XYe - \mu r = 0 \\ y = Mx + q \\ (x, y) > 0 \end{cases}$$

Donc, la solution de $(PCL)_{\mu r}$ est équivalente à résoudre le système$(**)$.

Définition 4.6.2.1.: L'ensemble de toutes les solutions du système $(**)$:

$T_r = \{(x_\mu, y_\mu): \mu > 0\}$ est appelé trajectoire centrale.

4.6.2.2. Facteur de centralité :[17]

La qualité de chaque solution trouvée est mesurée par un facteur dit de centralité :

$$\delta(x, y, \mu) = min\|XYe - \mu r\| = \left\|XYe - \left(\frac{x^t Ry}{n}\right) r\right\|$$

$(x, y) \in D_s$ est un point de la trajectoire centrale si : $\left\|XYe - \left(\frac{x^t Ry}{n}\right) r\right\| = 0.$

4.6.2.3. Concept de proximité [17]:

Le point (x, y) est à proximité de la trajectoire centrale $T(\theta)$ s'il appartient à l'ensemble suivant :

$$T(\theta) = \left\{(x, y) \in D_S: \left\|XYe - \left(\frac{x^t Ry}{n}\right) r\right\| \leq \left(\frac{x^t Ry}{n}\right) \theta: \theta > 0\right\}$$

- Pour résoudre le système (**), on définit la fonction :

$F_{\mu r}: \mathbb{R}_+^n * \mathbb{R}_+^n \to \mathbb{R}_+^n * \mathbb{R}_+^n$ est une fonction définie par :

$$F_{\mu r}(x, y) = \begin{bmatrix} XYe - \mu r \\ y - Mx - q \end{bmatrix}$$

La résolution du $F_{\mu r}(x, y) = 0$ revient à résoudre le système linéaire :

$$\begin{cases} Y\Delta x + X\Delta y = -\mu r - XYe \\ \Delta y = M\Delta x \end{cases} \qquad \text{(S1)}$$

(Sl) s'écrit sous la forme matricielle comme suit :

$$\begin{pmatrix} Y & X \\ M & -I \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} = \begin{pmatrix} -\mu r - XYe \\ 0 \end{pmatrix}$$

Par un simple calcul, on trouve la solution $(\Delta x, \Delta y)$ donnée par :

$$\begin{cases} \Delta x = (M + X^{-1}Y)(\mu X^{-1} r - Ye) \\ \Delta y = M\Delta x \end{cases}$$

Le nouvel itéré est calculé comme suit :

$$(x^+, y^+) = (x, y) + (\Delta x, \Delta y).$$

4.6.2.4. Calcul du pas de déplacement :

Pour réduire le nombre d'itération et le temps de calcul, on a introduit une procédure pratique moins couteuse pour la recherche et le calcul du pas de déplacement.

Si (x, y, s) est le point courant réalisable, alors le nouvel itéré :

$(x^+, y^+, s^+) = (x, y, s) + \alpha(\Delta x, \Delta y, \Delta s)$, doit être réalisable. Aussi qu'ils vérifient : $\begin{cases} x + \alpha_x \Delta x > 0 \\ y + \alpha_y \Delta y > 0 \end{cases}$

On prend: $\alpha = \beta min(\alpha_x^+, \alpha_y^+)$ tel que $0 < \beta < 1$.

Où : $$\alpha_x = \begin{cases} min - \frac{x_i}{\Delta x_i} \; avec \; i \in I = \{i : \Delta x_i < 0\} \\ 1 \qquad\qquad si \;\; \Delta x_i \geq 0 \end{cases}$$

$$\alpha_y = \begin{cases} min - \frac{y_j}{\Delta y_j} \; avec \; j \in J = \{j : \Delta y_j < 0\} \\ 1 \qquad\qquad si \;\; \Delta y_j \geq 0 \end{cases}$$

4.6.2.5. Algorithme de trajectoire centrale avec poids:

Début de l'algorithme :

Données :$\varepsilon > 0$ un paramètre de précision.

Initialisation : $(x^0, y^0) \in D_s$, $\mu = \frac{\|Y^0 X^0 e\|}{\sqrt{n}}, r = \frac{Y^0 X^0 e}{u}$, $\delta = \left(1 - \frac{\sqrt{2}}{2}\right)\eta$.

$X^0 = diag(x^0), Y^0 = diag(y^0), \eta = \min r_i$ et $R = diag(r)$ pour i :1,…,n.

Itération : pour $k = 0,1,2\ldots$.

Calculer :

-les paramètres : $\mu^k = (1 - \frac{\delta}{\sqrt{n}}) \frac{(x^k)^t R y^k}{n}$

-la direction de Newton $(\Delta x^k, \Delta y^k)$ solution du système linéaire :

$$\begin{cases} Y^k \Delta x^k + X^k \Delta y^k = \mu^k r - X^k Y^k e \\ \Delta y^k = M \Delta x^k \end{cases}$$

-calcul du pas α_k,tel que $\alpha_k = (\alpha_x, \alpha_y)$

-le nouvel itéré : $(x^{k+1}, y^{k+1}) = (x^k, y^k) + \alpha_k(\Delta x^k, \Delta y^k)$, $k = k + 1$

Test d'arrêt: $(x^k)^t R y^k \leq \varepsilon$.

Fin algorithme.

- Donc l'algorithme Part d'un point initial$(x^0, y^0) \in T_r(\theta)$.

Pour des choix convenables des constantes positives δ et η, l'algorithme génère une suite $(x^k, y^k) \in T_r(\theta)$ (le nouveau point reste toujours dans le voisinage de la trajectoire centrale) en partant d'un point initial $(x^0, y^0) \in T_r(\theta)$,de

plus ,à chaque itération le terme $(x^k)^t Ry^k$ se réduit au moins linéairement par un montant fixe $\left(1-\frac{\eta}{6\sqrt{n}}\right)$.

4.6.2.6. Convergence de l'algorithme :[5]

On a le théorème suivant :

Théorème 4.6.2.6.1. :

Soient $(x^0, y^0) \in D_s$, $\mu = \frac{\|Y^0 X^0 e\|}{\sqrt{n}}$, $r = \frac{Y^0 X^0 e}{\mu}$, $\delta = \theta = \left(1-\frac{\sqrt{2}}{2}\right)\eta$ et

$\eta = \min r_i$ pour i : 1, … , n.

Supposons que : $(x^k, y^k) \in T_r(\theta)$ et $\mu^k = (1-\frac{\delta}{\sqrt{n}})\frac{(x^k)^t Ry^k}{n}$, alors :

1) $(x^{k+1}, y^{k+1}) \in T_r(\theta)$
2) $(x^{k+1})^t Ry^{k+1} \leq \left(1-\frac{\eta}{6\sqrt{n}}\right)(x^k)^t Ry^k$

Où : $T_r(\theta) = \{(x,y) \in D_S : \|XYe - \mu r\| \leq \mu\theta : \theta > 0\}$.

Remarque : L'inconvénient majeur de ce type des méthodes réside dans l'initialisation, c'est-à-dire la détermination d'un point initial qui se trouve à l'intérieur du domaine et proche de la trajectoire centrale. Théoriquement, on suppose que point est connu, pour cela, on intègre la méthode de Karmarkar pour le calcul de ce dernier comme suit :

4.6.2.7. Le calcul d'une solution initiale strictement réalisable :

Un point est strictement réalisable de (PCL) est la solution du problème :

$$\{y = Mx + q, (x,y) > 0\} \quad \text{.....(**)}$$

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

$$\begin{cases} Min\ \lambda \\ s.c \\ Az + \lambda(q - Az^0) = q \\ z \geq 0, \lambda \geq 0 \end{cases} \text{.....(P1)}$$

Tel que : $z^0 \in \mathbb{R}^n_{++}$ est choisi arbitrairement dans l'orthant positif et λ une variable artificielle. Posons $\omega = (y, x, \lambda)$, le problème (P1) est équivalent au problème linéaire suivant :

$$\begin{cases} Min\ c^t\omega = z^* \\ s.c \\ \bar{A}\omega = b \\ \omega > 0 \end{cases} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{(P2)}$$

Où : $c^t = (0, \ldots, 1) \in \mathbb{R}^{n+m+1}$, $\bar{A} = [I \quad -M \quad b - Aa] \in \mathbb{R}^{m*(m+n+1)}$.

Le problème (P2) vérifie les hypothèses de Karmarkar :

1) $z^* = 0$.
2) $\omega = (a, 1)$ est une solution strictement réalisable de (P2).
3) $\bar{A}$ est une matrice de plein rang (rang $\bar{A} = m \leq m + n + 1$).

Plus précisément Karmarkar démontre le théorème suivant :

Théorème 4.6.2.7.1.:

$\exists\ \varepsilon > 0$, telle que les deux propositions suivantes sont équivalentes :

1) (x, y) une solution de (P1).
2) (P2) admet une solution optimale $\omega = (y, x, \lambda)/\lambda \leq \varepsilon$.

La résolution de (P1) se ramène donc à celle du problème (P2), auquel on appliquera l'algorithme de Karmarkar, puisque connaissant la valeur optimale et une solution réalisable de ce dernier.

4.7. Tests numériques :

Nos tests d'applications sot réalisés su un PC portable Acer sous Windows 7 en utilisant le langage MATLAB.

Les exemples traités sont de la forme canonique suivante :

$$\text{(PL)} \qquad \begin{cases} \text{Min f(x)} \\ sc \\ \text{A x} = \text{ b} \\ \text{x} \geq 0 \end{cases}$$

La précision est comprise entre $10^{-3}\ et\ 10^{-6}$.

Pour la recherche des solutions initiales, on présente un algorithme un algorithme réduit de Karmarkar **(Annexe1)**, qui va nous aider à résoudre ce dernier avec la trajectoire centrale avec et sans poids.

1) L'application avec la méthode de TC sans poids :

a)cas linéaire :

Exemple 01 : Soit le problème linéaire suivant :

$$\begin{cases} Min\ 3x_1 + 2x_2 + x_3 + 3x_4 \\ x_1 - x_2 + x_3 + x_4 = 3 \\ 2x_1 + x_2 - x_3 + 2x_4 = 4 \\ x_1 + x_3 + 2x_4 = 5 \\ x_1, x_2, x_3, x_4 \geq 0 \end{cases}$$

Prenons les solutions trouvées initiales suivantes:

$$x^{\circ} = (0.341957\ , 0.004313\ , 0.670979\ , 1.991376\ )^t$$

$$y^{\circ} = (-0.005320\ , 0.274080, 1.222858)^t$$

Après une 63 itérations, On obtient les solutions optimales suivantes :

$$x^* = (0.2972 \quad 0.0181 \quad 0.6486 \quad 2.0414)^t$$

$$y^* = (\ 1.4046 \quad 0.6134 \quad 0.1619)^t$$

Et la valeur optimale: $z^* = 7.8$.

Exemple 02 Soit le problème linéaire suivant :

$$\begin{cases} Min\ 4x_1 + x_2 + 2x_3 \\ sc \\ 2x_1 + 3x_2 + x_3 + 2x_4 = 2 \\ 3x_1 - 2x_3 + x_4 = 0 \\ x_1, x_2, x_3, x_4 \geq 0 \end{cases}$$

Soient les solutions initiales du problème :

$$x^{\circ} = (0.001069\ , 0.663691\ , 0.002641\ , 0.002074\ )^t$$

$$y^{\circ} = (0.270655\ , -0.546445)^t$$

Les solutions optimales trouvées sont :

$$x^* = (0.0000 \quad 0.6666 \quad 0.0000 \quad 0.0000)^t$$

$$y^* = (\ 0.2707 \quad -0.5464)^t$$

Après 54 itérations avec la valeur optimale $z^* = 0.6666$.

b) cas quadratique :

Exemple 03 : Soit le problème quadratique suivant :

$$\begin{cases} Min\ f(x) = \frac{1}{2}x^t Q x + c'x \\ sc \\ Ax = b \\ x \geq 0 \end{cases}$$

Tel que :A=$\begin{pmatrix} -1 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix}$, Q= $\begin{pmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{pmatrix}$ et C=$(-2, -4\ , 0)^t$, b=$\begin{pmatrix} 1 \\ 2 \end{pmatrix}$.

Pour epsilon=0.00001 et les solutions initiales égales à :

$$x^{\circ} = (0.012567\ ,\ 1.012567\ ,\ 0.974867\ )^t$$

$$y^{\circ} = (-0.000000\ ,\ 0.001387\ ,\ 1.976253)^t$$

Après 23 itérations, on obtient, les solutions optimales suivantes :

$$x^* = (0.5000 \quad 1.5000 \quad 0\ )^t$$

$$y^* = (\ \ 0.0014 \quad 2.9525)^t$$

Qui donnent la valeur optimale $z^* = -4.5$;

Exemple 04: Soit le problème quadratique suivant :

$$\begin{cases} Min\ (x_1 - 1)^2 + (x_2 - 2.5)^2 \\ sc \\ x_1 - 2x_2 + x_3 = -2 \\ -x_1 - 2x_2 + x_4 = 2 \\ -x_1 + 2x_2 + x_5 = -2 \\ x_1, x_2, x_3, x_4, x_5 \geq 0 \end{cases}$$

Les solutions initiales trouvées égales à :

$x^{\circ} = (2.000000\ ,\ 0.000001\ ,\ 4.000000\ ,\ 0.00001\ ,\ 0.000001)^t$

$y^{\circ} = (0.000013\ , -2.271984\ ,\ 0.000788)^t$

On obtient, les solutions optimales suivantes après 123 itérations :

$x^* = (2.0000 \quad 0.0000 \quad 4.0000 \quad 0.0000 \quad 0.0000\ )^t$

$y^* = (\ 0.0000 \quad -2.2720 \quad 0.0008)^t$, la valeur optimale $z^* = -4.25$.

2) L'application de méthode de TC avec poids :

1) cas linéaire :

Exemple 05 : soit le problème linéaire suivant :

$$\begin{cases} Min\ 4x_1 + x_2 + 2x_3 \\ sc \\ 2x_1 + 3x_2 + x_3 + 2x_4 = 2 \\ 3x_1 - 2x_3 + x_4 = 0 \\ x_1, x_2, x_3, x_4 \geq 0 \end{cases}$$

Soient les solutions initiales du problème :

$$x^{\circ} = (0.001069\ , 0.663691\ , 0.002641\ , 0.002074\ )^t$$

$$y^{\circ} = (0.270655\ , -0.546445)^t$$

Les solutions optimales trouvées après 49 itérations, sont :

$$x^* = (0.0000 \quad 0.6666 \quad 0.0000 \quad 0.0000)^t$$

$$y^* = (\,0.2707 \quad -0.5464)^t$$

Qui donnent $z^* = 0.6666$.

Exemple 06 : Soit le problème linéaire suivant :

$$\begin{cases} Min \;\; 3x_1 + 2x_2 + x_3 + 3x_4 \\ sc \\ x_1 - x_2 + x_3 + x_4 = 3 \\ 2x_1 + x_2 - x_3 + 2x_4 = 4 \\ x_1 + x_3 + 2x_4 = 5 \\ x_1, x_2, x_3, x_4, x_5 \geq 0 \end{cases}$$

Les solutions initiales trouvées sont:

$$x^{\circ} = (0.341957 \;, 0.004313 \;, 0.670979 \;, 1.991376 \;)^t$$

$$y^{\circ} = (-0.005320 \;, 0.274080, 1.222858)^t$$

Les solutions optimales trouvées après 46 itérations, sont :

$$x^* = (0.3333 \quad 0.0000 \quad 0.6667 \quad 2.0000)^t$$

$$y^* = (\,1.6646 \quad 0.6662 \quad 0.0015)^t$$

Avec la valeur optimale du problème : $z^* = 7.6867$.

Exemple 07 : Soit le programme linéaire :

$$\begin{cases} Min -4x_1 - 5x_2 \\ sc \\ 2x_1 + x_2 + x_3 = 8 \\ x_1 + 2x_2 + x_4 = 7 \\ x_1 \,, x_2, x_3, x_4 \geq 0 \end{cases}$$

Soient les solutions initiales trouvées suivantes :

$$x^{\circ} = (2.934176 \,, 2.023044 \;, 0.108604 \,, 0.019737 \;)^t$$

$$y^{\circ} = (-1.024282 \,, -1.971814)^t$$

Qui donnent les solutions optimales :

$$x^* = (3.0000 \quad 2.0000 \quad 0.0000 \quad 0.0000)^t$$

$$y^* = (\,-1.0703 \quad -1.8594)^t.$$

Et la valeur optimale $z^* = -22$ après 45 itérations.

3) Cas quadratique :

Exemple 08 : Soit le problème quadratique suivant :

$$\begin{cases} Min\, f(x) = \frac{1}{2} x^t Q x + c' x \\ Ax = b \\ x \geq 0 \end{cases}$$

Tel que : A=$\begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 5 & 0 & 1 \end{pmatrix}$, Q=$\begin{pmatrix} 4 & -2 & 0 & 0 \\ -2 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$, et C = $(-4\,, -6\ ,0\ ,0\ )^t$, b=$\begin{pmatrix} 2 \\ 5 \end{pmatrix}$

Les solutions initiales trouvées sont :

$$x^{\circ} = (0.628088\ ,\ 0.872245\ ,\ 0.499667\ ,\ 0.010689\ )^t$$

$$y^{\circ} = (-3.097763\ ,\ -0.137088)^t$$

Après 48 itérations, on trouve les solutions optimales après 19 itérations :

$$x^* = (1.25\ ,\ 0.75\ ,\ 0\ ,\ 0\ )^t$$

$$y^* = (\ -3.1256\quad -0.4749)^t$$

Et la valeur optimale du problème : $z^* = -6.5340.$

Exemple 09 : pour :

A=$\begin{pmatrix} -2 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}$, Q= $\begin{pmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{pmatrix}$ et C=$(-4, -5, 0)^t$, b=$\begin{pmatrix} 1 \\ 1 \end{pmatrix}$.

Les solutions initiales trouvées égales à :

$$x^{\circ} = (0.012567\ ,\ 1.012567\ ,\ 0.974867\ )^t$$

$$y^{\circ} = (-0.000000\ ,\ 0.001387\ ,\ 1.976253)^t$$

On obtient, les solutions optimales suivantes :

$$x^* = (0.2098\quad 1.4070\quad 0.7777\ )^t$$

$$y^* = (0.7902\quad 3.9525)^t$$

Qui donnent la valeur optimale : $z^* = -6.67.$

Commentaire : A travers les tests numériques effectuées, on constate l'efficacité pratique de l' algorithmes de trajectoire centrale en terme d'infériorité du nombre pratique du nombre d'itération qui est très réduit, et de la réduction de la durée d'exécution. Aussi le paramètre poids, joue un rôle très important à la réduction de ces derniers.

5.1. Introduction :

Dans ce chapitre, on va présenter deux méthodes de points intérieurs, la premier de type projectif celle de Karmarkar, et l'autre de type barrière logarithmique (méthode de trajectoire centrale avec poids), pour la minimisation de problèmes de fonction objectif non linéaire convexe différentiables sous contraintes linéaires, c'est-à-dire minimiser cette dernière sur un polyèdre.

Considérons le problème d'optimisation non linéaire convexe suivant :

$$\begin{cases} Min f(x) \\ \quad sc \\ Ax = b \\ x \geq 0 \end{cases} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (PNLC)$$

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

$D = \{x \in \mathbb{R}^n_+ : Ax = b\}$: L'ensemble des solutions réalisables de$(PNLC)$.

5.2. Méthode de Karmarkar:

5.2.1.Présentation de la méthode :

Considérons le problème d'optimisation non linéaire sous contraintes linéaire suivant :

$$\begin{cases} Min\, f(x) \\ \quad sc \\ Ax = b \\ x \geq 0 \end{cases} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (1)$$

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

Supposant les hypothèses suivantes (hypothèses de Karmarkar) :

(Hypothèse 01): La matrice A est de plein rang (rg A = m ≤ n)

(Hypothèse 02): On dispose d'un point x^0 strictement réalisable $(Ax^0 = b, x^0 > 0)$.

(Hypothèse 03): La valeur optimale z^*de l'objectif est connue au départ.

Remarque :

A l'hypothèse 2, pour le système de contraintes :$Ax = b, b \neq 0$, on se ramène facilement à un système homogène .il suffit d'écrire :

$$Ax = be_{n+1}^t x \Rightarrow (A - be_{n+1}^t)x = 0$$

Soit la transformation projective notée $T_k(x)$ une fonction définie par :[2]

$T_k(x)$: $\mathbb{R}_+^n \to S_{n+1}$ définie par : $T_k(x) = y$

Avec :

$$\begin{cases} y_i = \frac{x_i/x_i^k}{1+\sum_{i=1}^n x_i/x_i^k} \\ y_{n+1} = 1 - \sum_{i=1}^n y_i \end{cases} \quad \text{.....(2)}$$

On a :

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

Ou encore : $y[n] = (D_k^{-1}x)y_{n+1}$ ou $y[n]$ désigne les n première composantes de y .

Remarque :

La transformation $T_k(x)$ est univoque, et donnons sa transformation inverse par : $$x = T_k^{-1}(y) = \frac{D_k y[n]}{y_{n+1}}, D_k = diag(x^k).$$

- Alors, On commence par ramener le problème à la forme simplifiée suivante :

$$\begin{cases} Min\ g(y) = 0 \\ sc \\ By = 0 \\ y \in S_{n+1} \end{cases} \quad \text{.....(2)}$$

Où g: $\mathbb{R}^{n+1} \to \mathbb{R}$, est une fonction non linéaire, convexe et différentiable. $S_{n+1} = \{y \epsilon \mathbb{R}^{n+1}: e_{n+1}^t = y, y \geq 0\}$ est le simplexe de dimension n et de centre a tel que $a_i = \frac{1}{n+1}, \forall i \in \{1, \dots n+1\}, e_{n+1} \in \mathbb{R}^{n+1}$ où : $e_i = 1, \forall i \in \{1, \dots, n+1\}$.

- En appliquant la transformation projective $T_k(x)$ sur le problème (1), on trouve le problème suivant :

$$\begin{cases} \text{Min}\left[f\left(T_k^{-1}(y)\right) - z^*\right] = f\left(\frac{D_k y[n]}{y_{n+1}}\right) - z^* \\ \frac{D_k y[n]}{y_{n+1}} = b \\ \sum_{i=1}^{n+1} y_i = 1 \\ y[n] \geq 0, y_{n+1} \end{cases} \quad \text{.....(3)}$$

Qui est équivalent à résoudre le problème suivant :

$$\begin{cases} \text{Min} y_{n+1}\left[f\left(\mathrm{T}_k^{-1}(y)\right)-z^*\right]=y_{n+1}\left[f\left(\frac{D_k y[n]}{y_{n+1}}\right)-z^*\right] \\ D_k y[n]-b y_{n+1}=0 \\ \sum_{i=1}^{n+1} y_i=1 \\ y[n]\geq 0, y_{n+1} \end{cases} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots(4)$$

Encore équivalent à :

$$\begin{cases} Min\ g(y)=y_{n+1}\left[f\left(T_k^{-1}(y)\right)-z^*\right] \\ sc \\ A_k y=0 \\ y\in S_{n+1} \end{cases} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (5)$$

Avec :$A_k=[AD_k-b], y=\begin{bmatrix} y[n] \\ y_{n+1} \end{bmatrix}$.

Notons que toute solution réalisable de (1) est transformée par $\mathrm{T}_k(x)$ en une solution réalisable de (5) et réciproquement, toute solution réalisable y de (5) avec $y_{n+1}>0$ est transformée par T_k^{-1} en une solution réalisable x de (1).et que la valeur optimale de $g(y)$ est 0 et que le centre du simplexe est réalisable pour (5).

Lemme 5.2.1.1. : La fonction f étant convexe sur l'ensemble $D=\{x\in\mathbb{R}^n: Ax=b, x\geq 0\}$, il en est de même pour g sur l'ensemble: $Ds=y\in\mathbb{R}n+1:Aky=0, y\in Sn+1$.

Preuve : On a g est une fonction continue sur D_s, il suffit de montrer que :

$\forall\lambda\in[0,1], g(\lambda y+(1-\lambda)\hat{y})\leq\lambda g(y)+(1-\lambda)g(\hat{y}), \forall y,\hat{y}\in D_s$

Nous avons :$\forall y,\hat{y}\in D_s, \exists x,\hat{x}\in D$ tel que : $x=T_k^{-1}(y)$ *et* $\hat{x}=T_k^{-1}(\hat{y})$, et alors :

$$g(\lambda y+(1-\lambda)\hat{y})=(\lambda y_{n+1}+(1-\lambda)\hat{y}_{n+1})\left[f\left(\frac{D_k(\lambda y[n]+(1-\lambda)\hat{y}[n])}{(\lambda y_{n+1}+(1-\lambda)\hat{y}_{n+1})}\right)-z^*\right]=$$

$$=[\lambda y_{n+1}+(1-\lambda)\hat{y}_{n+1}]*$$

$$\left[f\left(\lambda\left(\frac{y_{n+1}}{(\lambda y_{n+1}+(1-\lambda)\hat{y}_{n+1})}x-z^*\right)+(1-\lambda)(\frac{\hat{y}_{n+1}}{(\lambda y_{n+1}+(1-\lambda)\hat{y}_{n+1})}\hat{x}\right)-z^*)\right]$$

Comme $f(x)$ est convexe, alors :

$$g(\lambda y+(1-\lambda)\hat{y})\leq(\lambda y_{n+1}+(1-\lambda)\hat{y}_{n+1})\begin{bmatrix} \lambda\left(\frac{y_{n+1}}{(\lambda y_{n+1}+(1-\lambda)\hat{y}_{n+1})}f(x)\right)+ \\ (1-\lambda)(\frac{\hat{y}_{n+1}}{(\lambda y_{n+1}+(1-\lambda)\hat{y}_{n+1})}f(\hat{x})-z^*) \end{bmatrix}$$

$$\leq\lambda y_{n+1}[f(x)-z^*]+(1-\lambda)\hat{y}_{n+1}[f(\hat{x})-z^*]=\lambda g(y)+(1-\lambda)g(\hat{y})$$

Donc: $g(\lambda y+(1-\lambda)\hat{y})\leq\lambda g(y)+(1-\lambda)g(\hat{y})\Longrightarrow g(y)$est convexe sur D_s.

- Après la formulation du problème (4), En linéarisant la fonction $g(y)$ au voisinage du centre du simplexe et en introduisant une boule de centre a (centre du simplexe) et de rayon α considérée comme voisinage de a :

$$g(y) = g(a) + \nabla g(a)^t (y - a) \text{ Pour } y \in \{y \in \mathbb{R}^{n+1} : \|y - a\| \leq \alpha\}.$$

On obtient le sous problème suivant :

$$\begin{cases} Ming(y) = g(a) + \nabla g(a)^t (y - a) \\ sc \\ A_k y = 0 \\ e_{n+1}^t y = 1, y \geq 0 \\ \|y - a\|^2 \leq \alpha^2 \end{cases} \quad \ldots\ldots\ldots\ldots\ldots\ldots (6)$$

Qui équivalent au problème suivant :

$$\begin{cases} Min \nabla g(a)^t y \\ sc \\ A_k y = 0 \\ e_{n+1}^t y = 1, y \geq 0 \\ \|y - a\|^2 \leq \alpha^2 \end{cases} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots (7)$$

On sait que (lemme3.4.3.1 chapitre 03) pour $\alpha < 1$, la contrainte $y \geq 0$ est redondante et donc :

$$\begin{cases} Min \nabla g(a)^t y \\ sc \\ A_k y = 0 \\ e_{n+1}^t y = 1 \\ \|y - a\|^2 \leq \alpha^2 \end{cases} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots (8)$$

Lemme 5.2.2:[3]

La solution optimale du problème (7) est donnée explicitement par :$y^k = a - ad^k$ où $d^k = \frac{p^k}{\|p^k\|}$ et $p^k = pB_k\big(\nabla g(a)\big), B_k = \begin{bmatrix} A_k \\ e_{n+1}^t \end{bmatrix}$.

Preuve : On pose $x = y - a$, alors on a:

$$B_k x = \begin{bmatrix} A_k \\ e_{n+1}^t \end{bmatrix} (y - a) = \begin{bmatrix} A_k \\ e_{n+1}^t \end{bmatrix} y - \begin{bmatrix} A_k \\ e_{n+1}^t \end{bmatrix} a = 0.$$

En remplaçant $(y - a)$ par x dans le problème (7), on obtient le problème suivant :

$$\begin{cases} Min \nabla g(a)^t x \\ sc \\ B_k x = 0 \\ \|x\|^2 \leq a^2 \end{cases} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots (9)$$

Le problème (8) est convexe ($\nabla g(a)^t x$ est convexe et les contraintes sont convexes), donc les conditions de KKT sont nécessaires et suffisantes, elles s'écrivent comme suit :

$$\begin{cases} \nabla g(a) + {B_k}^t \lambda + \mu x = 0 \ldots\ldots\ldots\ldots (a) \\ B_k x = 0 \qquad\qquad \ldots\ldots\ldots\ldots\ldots (b) \\ \mu(\|x\|^2 - a^2) = 0 \quad \ldots\ldots\ldots\ldots\ldots (c) \end{cases} \ldots\ldots\ldots\ldots\ldots (10)$$

Où : $\exists \lambda \in \mathbb{R}^{m+1}$ et $\exists \mu \in \mathbb{R}_+$ sont les multiplicateurs de Lagrange associés à les contraintes $B_k x = 0 \; et \; \|x\|^2 \leq a^2$ respectivement du problème (8).

Alors x^*est solution optimale si et seulement si:

$\exists \lambda \in \mathbb{R}^{m+1}$ et $\exists \mu \in \mathbb{R}_+$ tels que :

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

D'après l'équation (a) du système (9) : $\nabla g(a) + {B_k}^t \lambda + \mu x^* = 0$…….(*)

En multipliant (*) par B_k on trouve:

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

De l'équation (b) du système (9), on a: $B_k x^* = 0$, donc $B_k \nabla g(a) + B_k {B_k}^t \lambda = 0$.

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

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

$$\|x^*\| = \frac{1}{\mu}\|p_k\| \ldots\ldots\ldots\ldots\ldots\ldots (+)$$

Et on a dans l'équation (c) du système (9): pour $u \neq 0$, $\|x^*\| = \alpha$……..(++)

De (+) et (++), on trouve $\|x^*\| = \frac{1}{\mu}\|p_k\| = \alpha \Longrightarrow u = \frac{\|p_k\|}{\alpha}$ donc:

$x^* = -\frac{1}{\mu} p_k = -\alpha \frac{p_k}{\|p_k\|}$ et on a : $y^k = y^* = a + x^* = a - \alpha \frac{p_k}{\|p_k\|} = a - \alpha d_k.$

5.2.2. Algorithme:(Karmarkar pour la programmation non linéaire convexe):[3]

1) ***Initialisation*** **:** ε> 0 est une précision donnée, x^0 est un point strictement réalisable.

Tant que **:** $f(x^k) - \mathrm{z}^* \geq \varepsilon$ faire :

Pas 1 :

-construire : $D_k = diag(x^k)$,$A_k = [AD_k - b]$, $B_k = \begin{bmatrix} A_k \\ e^t_{n+1} \end{bmatrix}$.

-calculer :$p^k = [I - {B_k}^t(B_k{B_k}^t)^{-1}\, B_k]\nabla g(a)$, $d^k = \frac{p^k}{\|p^k\|}$.

-calculer : $y^k = a - \alpha d_k$.

Pas 2 :

-prendre :$x^{k+1} = T_k^{-1}(y^k) = \frac{D_k y^k[n]}{{y^k}_{n+1}}$,$k := k + 1$ et retourner au pas 1.

Fin tant que ;

Fin de l'algorithme.

- L'algorithme démarre d'une solution strictement réalisable x^0.et à chaque itération k, on applique la transformation projective T_k pour envoyer le point courant x^kau centre du simplexe, on calcule y^k solution optimale du problème linéarisé (7), et on revient à la variable initiale en appliquant la transformation inverse ${T_k}^{-1}$ et ainsi de suite jusqu'a ce que le test d'optimalité ($f(x^k) - \mathrm{z}^* \geq \varepsilon$) se réalise où ε> 0 est une précision donnée.

5.2.3. Convergence de l'algorithme :

La preuve de la convergence de notre algorithme est basée sur l'analyse de la fonction potentiel associée au problème (1), qui est définie par :

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

Sur $D_s = \{x \in \mathbb{R}^n : A_k x = 0, x \in S_n\}$.

Notre but est de montrer la réduction de $p(x)$ la fonction potentielle, pour avoir la réduction dans la fonction$(f(x) - z^*)$. Donc si la suite $(p(x^k))$ tend vers $-\infty$ alors la suite $f(x^k)$ tend vers 0.

Théorème 5.2.3.1 :[2]

A chaque itération de l'algorithme, la fonction potentiel se réduit d'une valeur constante δ telle que : $p(x^{k+1}) = p(x^k) - \delta$, où $\delta = \alpha - \frac{\alpha^2}{2(1-\alpha)^2}$.

Preuve : $p(x^{k+1}) - p(x^k) = (n+1)\ln\frac{f(x^{k+1})-z^*}{f(x^k)-z^*} - \sum_{i=1}^{n}\ln\frac{x_i^{k+1}}{x_i^k}$

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

En utilisant les résultats démontrés par Karmarkar :

Pour y^k la solution optimale du problème (7), on a :

1) $-\sum_{i=1}^{n+1}\ln y_i^k = \frac{\alpha^2}{2(1-\alpha)^2}$.
2) $g(y^k) \leq \left(1 - \frac{\alpha}{n+1}\right) g(a)$.

Donc : $p(x^{k+1}) - p(x^k) \leq (n+1)\ln\left(1 - \frac{\alpha}{n+1}\right) - \frac{\alpha^2}{2(1-\alpha)^2}$

$\leq \alpha - \frac{\alpha^2}{2(1-\alpha)^2}$ .Ce qui montre le résultat.

5.2.4. Test numériques :

Les exemples traités sont de la forme suivante :

$$\begin{cases} Min\ f(x) \\ sc \\ Ax = b \\ x \geq 0 \end{cases} \quad \text{.....(PNLC)}$$

Où $f: \mathbb{R}^n \to \mathbb{R}$ est une fonction non linéaire, convexe et différentiable, A une matrice de type (m, n).La précision est située entre $10^{-3}\ et\ 10^{-5}$.

Exemple 1 : Soit le problème non linéaire convexe suivant :

$$\begin{cases} exp(x_1 - x_2) \\ sc \\ 2(x_1 - x_2) + x_3 = 1 \\ x_1, x_2, x_3 \geq 0 \end{cases}$$

Pour : epsilon=0 .001 et $z^* = 0$, on trouve la solution optimale après 19 itérations:

$$x^* = (0.005 \quad 11.5994 \quad 22.3795)^t.$$

Qui donne la valeur optimale : $f(x^*) = \exp(-11.5989)$.

Exemple 02 : Soit le problème non linéaire suivant :

$$\begin{cases} \min x_1^2 + 2x_2^2 - 2x_1x_2 - 2x_1 - x_2 \\ sc \\ x_1 + x_2 = 2 \\ -x_1 + 2x_2 = 2 \\ x_1, x_2 \geq 0 \end{cases}$$

Pour $z^* = -7$ et esp=0.0001 après 26 itérations, on obtient :

La solution optimale qu'on obtient : x^*=$(0.6546 \ , \ 1.3453)^t$

Exemple 03 : Soit le problème non linéaire quadratique suivant :

$$\begin{cases} \min f(x) = \frac{1}{2}x^tQx + c^tx \\ sc \\ Ax = b \\ x \geq b \end{cases}$$

Tel que : Q=$\begin{pmatrix} 4 & -2 \\ -2 & 4 \end{pmatrix}$ et A=$\begin{pmatrix} 1 & 1 \\ 1 & 5 \end{pmatrix}$

avec : $b = \begin{pmatrix} 2 \\ 3 \end{pmatrix}$ et c=(-4 -6)

pour epsilon =0.0001, $z^* = -4.5$, on obtient :

$$x^*=(0.5 \ , \ 0.5)^t.$$

Commentaire :

Les exemples testés montrent la fiabilité, et l'efficacité pratique de cet algorithme en termes d'infériorité du nombre pratique d'itération, et la réduction de temps d'exécution.

5.3. La méthode de trajectoire centrale :

5 .3.1.Présentation de la méthode :

Considérons le problème d'optimisation non linéaire convexe suivant :

$$\begin{cases} Min f(x) \\ sc \\ Ax = b \\ x \geq 0 \end{cases} \ldots\ldots\ldots\ldots\ldots\ldots\ldots (PNLC)$$

Où $f: \mathbb{R}^n \to \mathbb{R}$ est une fonction non linéaire, convexe et différentiable,A une matrice de type (m, n) de plein rang (rg A = m ≤ n), $b \in \mathbb{R}^m$.

On note par : $e = (1, \ldots \ldots \ldots \ldots 1)^t \in \mathbb{R}^n$.

$D = \{x \in \mathbb{R}^n_+ : Ax = b\}$: L'ensemble des solutions réalisables de$(PNLC)$.

$D_S = \{x \in \mathbb{R}^n_{++} : Ax = b\}$: L'ensemble des solutions strictement réalisables de $(PNLC)$ qui est supposé non vide.

On associe à ce dernier, le problème pénalisé suivant :

$$\begin{cases} Min\ f_\mu(x) \\ sc \\ Ax = b \\ x > 0 \end{cases} \ldots\ldots\ldots\ldots\ldots\ldots\ldots (PNLC)_u$$

Où : $f_\mu(x)$est la fonction pénalisée définie par :

$$f_\mu(x) = f(x) - \mu \sum_{i=1}^{n} \ln\ x_i$$

Et μ un paramètre barrière strictement positif.

Proposition 5.3.1.1 :

La fonction f_μ est strictement convexe.

Preuve : il suffit de voir que :$\forall x \in \mathbb{R}^n_+$, $f_\mu(x) \in C^\infty$ et nous avons en particulier :

$\nabla f_\mu(x) = \nabla f(x) - \mu X^{-1} e$ avec $X = diag(x)$ et $X^{-1} e = diag\left(\frac{1}{x}\right) e$.

$\nabla^2 f_\mu(x) = \nabla^2 f(x) + \mu X^{-2}$ est une matrice définie positive (car f convexe).

D'où le problème $(PNLC)_\mu$ (les contraintes sont linéaires), donc les conditions de KKT sont nécessaires et suffisantes, elles s'écrivent comme suit :

$$\begin{cases} \nabla f(x) - \mu X^{-1} e - A^t y = 0 \\ b - Ax = 0 \\ x > 0, y \in \mathbb{R}^m \end{cases}$$

Où : $x \in \mathbb{R}^n_{++}$ et $y \in \mathbb{R}^m$ est le multiplicateur de Lagrange associé à la contrainte $(b - Ax = 0)$ du problème$(PNLC)_u$.

On pose $s = \mu X^{-1} e \in \mathbb{R}^n_{++}$, le système précédent devient :

$$\begin{cases} A^t y + s = \nabla f(x) \\ Ax = b \\ Xs - \mu e = 0 \\ (x, s) > 0 \end{cases} \quad \ldots\ldots\ldots\ldots..(1)$$

Pour chaque valeur de μ, on résout le système (1) par la méthode de newton, on obtient ainsi une suite de solution paramétrée qui converge vers la solution du problème initiale lorsque μ tend vers 0.

Définitions 5.3.1.1:

On définit l'ensemble :

$$T = \{(x, y, s) \in D_S * \mathbb{R}^m * \mathbb{R}^n_{++} : A^t y + s = \nabla f(x)\}$$

La trajectoire centrale est l'ensemble de toutes les solutions $\{(x_\mu, y_\mu, s_\mu)/\mu > 0\}$ du système :

$$\begin{cases} A^t y + s = \nabla f(x) \ldots\ldots\ldots\ldots\ldots (a) \\ Ax = b \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (b) \\ x_i s_i = \mu \, , i = 1 \ldots n \ldots\ldots\ldots\ldots . (c) \\ (x, s) > 0 \end{cases} \ldots\ldots\ldots\ldots\ldots\ldots(2)$$

Elle est notée :

$$TC = \{(x, y, s) \in T : XSe - \mu e = 0\} \ldots\ldots\ldots\ldots\ldots\ldots..(3)$$

Avec $S = diag(s)$.

Puisqu'on a une équation non linéaire dans le système (1) de même pour (2), il est difficile d'obtenir une solution exacte de $(PNLC)_u$. Nous allons donc nous contenter d'une solution approchée dont la qualité est jugée satisfaisante lorsqu'elle se

trouve dans le voisinage de la trajectoire centrale (TC). Ce voisinage correspond aux conditions de KKT où l'on remplace les contraintes non linéaires par la contrainte :

$$\|XSe - \mu e\| \leq \theta\mu \ \ avec\ 0 < \theta < 1.$$

Définition 5.3.1.2 :

Un point est dit voisin de trajectoire centrale s'il appartient à l'ensemble :

$$T(\theta) = \{(x, y, s) \in T : \|XSe - \mu e\| \leq \theta\mu : 0 < \theta < 1\} \ldots\ldots\ldots\ldots\ldots\ldots (4)$$

Les méthodes primales duales déterminent les solutions du système (2) en appliquant la méthode de Newton aux équations (2.a), (2.b) et (2.c) comme suit :

On définit la fonction non linéaire :

$f_\mu : \mathbb{R}^{2n+m} \longrightarrow \mathbb{R}^{2n+m}$:

$$f_\mu(x, y, s) = \begin{pmatrix} A^t y + s - \nabla f(x) \\ Ax - b \\ XSe - \mu e \end{pmatrix}$$

Où $X = diag(x)$ et $S = diag(s)$.

La direction de newton $(\Delta x, \Delta y, \Delta s)$ est solution du système linéaire :

$$\nabla f_\mu(x, y, s) \begin{pmatrix} \Delta x \\ \Delta y \\ \Delta s \end{pmatrix} = -f_\mu(x, y, s).$$

Où :

$$\nabla f_\mu(x, y, s) = \begin{pmatrix} \nabla^2 f_\mu(x) & -A^t & -I \\ A & 0 & 0 \\ S & 0 & X \end{pmatrix} \ldots\ldots\ldots\ldots\ldots\ldots (6)$$

- Le nouvel itéré est définie par :

$$(x^+, y^+, s^+) = (x, y, s) \ + \ (\Delta x, \Delta y, \Delta s).$$

Où $(\Delta x, \Delta y, \Delta s)$est solution du système linéaire (6), C'est-à-dire le système d'équations linéaires suivant :

$$\begin{pmatrix} \nabla^2 f_\mu(x) & -A^t & -I \\ A & 0 & 0 \\ S & 0 & X \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta y \\ \Delta s \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ -XSe + \mu e \end{pmatrix}$$

5.3.2. Algorithme de trajectoire centrale pour la programmation convexe :[3]

Début de l'algorithme :

Données :$\varepsilon > 0$ un paramètre de précision.

1-Initialisation : $\theta, \beta \in [0,1]$ sont des constantes.

Soient : $(x^0, y^0, s^0) \in T(\theta), \mu = \frac{(x^0)^t s^0}{n}$ avec :

$$X^0 = diag(x^0), S^0 = diag(s^0).$$

2-Itération : pour $k = 0,1,2\ldots.$

Pas 1 :

Calculer :

-les paramètres : $\mu^{k+1} = \beta\mu^k$

-résoudre le système :

$$\begin{pmatrix} \nabla^2 f_\mu(x) & -A^t & -I \\ A & 0 & 0 \\ S & 0 & X \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta y \\ \Delta s \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ -XSe + \mu e \end{pmatrix}$$

Pas 2 :

Pour obtenir : $(\Delta x, \Delta y, \Delta s)$.

-le nouvel itéré : $(x^{k+1}, y^{k+1}, s^{k+1}) = (x^k, y^k, s^k) + (\Delta x, \Delta y, \Delta s), k = k + 1$

3-Jusqu'à: $(x^k)^t s^k \leq \varepsilon$

Fin de l'algorithme.

Remarque : Si le point initial calculé n'est pas voisin de la trajectoire centrale, rien ne garantie la convergence de cet algorithme même si ce point est strictement réalisable. Nous introduisons le paramètre poids associé à la fonction barrière afin que le point initial reste au voisinage de la trajectoire centrale.

5.3.3. Méthode de trajectoire centrale avec poids :

5.3.3.1. Présentation de la méthode :

Considérons le problème d'optimisation non linéaire convexe suivant :

$$\begin{cases} Minf(x) \\ sc \\ Ax = b \\ x \geq 0 \end{cases} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (PNLC)$$

Où $f: \mathbb{R}^n \to \mathbb{R}$ est une fonction non linéaire, convexe et différentiable,A une matrice de type (m, n) de plein rang (rg A = m ≤ n), $b \in \mathbb{R}^m$.

On associe à $(PNLC)$ le problème perturbé défini comme suit :

$$\begin{cases} Minf_{\mu r}(x) \\ Ax = b \\ x > 0 \end{cases} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (PNLC)_{\mu r}$$

Où :

$$f_{\mu r}(x) = f(x) - \mu \sum_{i=1}^{n} r_i \ln\ x_i$$

Avec : $\mu > 0, r = (r_1, r_2, \ldots\ldots, r_n)^t \in \mathbb{R}_+^n$ sont les poids associés à la fonction barrière logarithmique.

Notons :$R = diag(r)$. La matrice diagonale dont ses éléments diagonaux sont les $r_i, i = 1 \ldots n$.

On a $f_{\mu r}(x)$ est strictement convexe et les contraintes du problème $(PNLC)_{ur}$ sont linéaire (affines), alors les conditions de KKT correspondantes sont nécessaires et suffisantes et s'écrivent comme suit :

$$\begin{cases} \nabla f(x) - \mu X^{-1} r - A^t y = 0 \\ b - Ax = 0 \\ x > 0, y \in \mathbb{R}^m \end{cases} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (1)$$

Où : $x \in \mathbb{R}_+^n$ et $y \in \mathbb{R}^m$ est le multiplicateur de Lagrange associé à la contrainte $(b - Ax = 0)$ du problème $(PNLC)_{\mu r}$.

On pose $s = \mu X^{-1} r \in \mathbb{R}^n_{++}$, le système précédent devient :

$$\begin{cases} A^t y + s = \nabla f(x) \\ Ax = b \\ Xs - \mu r = 0 \\ (x, s) > 0 \end{cases} \quad \text{……..(2)}$$

Pour chaque valeur de μ, on résout le système (2) par la méthode de newton, on obtient ainsi une suite de solution paramétrisées qui converge vers la solution du problème initiale lorsque μ tend vers 0.

Remarques : 1) pour $r = (1,1, \dots \dots 1)^t \in \mathbb{R}^n$, on trouve le système classique, c'est-à-dire la méthode trajectoire centrale (sans poids).

2) la solution optimale finie de $(PNLC)_{\mu r}$converge vers la solution de $(PNLC)$ lorsque μ tend vers 0.

- On applique la méthode de newton pour $(x, y, s) \in T$, on obtient le système :

$$\begin{pmatrix} \nabla^2 f_\mu(x) & -A^t & -I \\ A & 0 & 0 \\ S & 0 & X \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta y \\ \Delta s \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ -XSe + ur \end{pmatrix} \dots \dots \dots \dots \dots \dots (3)$$

- Le nouvel itéré est définie par :

$$(x^+, y^+, s^+) = (x, y, s) + (\Delta x, \Delta y, \Delta s).$$

Où $(\Delta x, \Delta y, \Delta s)$est solution du système linéaire (6).

Le point (x^+, y^+, s^+) est dit voisin de trajectoire centrale s'il appartient à l'ensemble :

$$T_r(\theta) = \{(x, y, s) \in T : \|XSe - ur\| \leq \theta u : 0 < \theta < 1\} \text{……..(4)}$$

5.3.3.2. Le calcul de la direction de descente$(\Delta x, \Delta y, \Delta s)$:

Le calcul de la direction nécessite la résolution du système linéaire (3) qui s'écrit sous la forme :

$$\begin{cases} -\nabla^2 f_\mu(x)\Delta x + A^t \Delta y + I\Delta s = 0 \\ A\Delta x = 0 \\ S\Delta x + X\Delta s = -XSe + ur \end{cases} \quad \text{…….(5)}$$

On peut éliminer grâce à la dernière équation, qui donne :

$$\Delta s = X^{-1}[-XSe + ur - S\Delta x] = -s + uX^{-1}r - X^{-1}S\Delta x$$

Le système (5) se réduit au système :

$$\begin{cases}[-X^{-1}S - \nabla^2 f_\mu(x)]\Delta x + A^t \Delta y = s - uX^{-1}r \\ A\Delta x = 0 \end{cases} \text{……..(6)}$$

On peut poursuivre la réduction de la dimension du système linéaire (6) à résoudre en éliminant Δx grâce à la première équation, on obtient :

$$\Delta x = [X^{-1}S + \nabla^2 f_\mu(x)]^{-1}[A^t \Delta y - \ s + uX^{-1}r]$$

On remplace Δx dans la deuxième équation du système(6), on obtient alors le système :$A\Delta x = A[X^{-1}S + \nabla^2 f_\mu(x)]^{-1}[A^t \Delta y - \ s + uX^{-1}r] = 0$

$\Leftrightarrow\ A[X^{-1}S + \nabla^2 f_\mu(x)]^{-1}A^t \Delta y = A[X^{-1}S + \nabla^2 f_\mu(x)]^{-1}[s - uX^{-1}r]$……(++)

On résout le système (++), pour obtenir la direction Δy unique, puis on déduit Δx et Δs.

Remarque : Pour montrer unicité de la direction Δy, présentons le lemme suivant.

Lemme 5.3.3.1. : La matrice $A[X^{-1}S + \nabla^2 f_\mu(x)]^{-1}A^t$ du système est symétrique, définie positive.

Preuve : 1-La matrice $A[X^{-1}S + \nabla^2 f_\mu(x)]^{-1}A^t$ est symétrique :

$$\left[A[X^{-1}S + \nabla^2 f_\mu(x)\right]^{-1}A^t]^t = [A^t]^t[(X^{-1}S)^t + \nabla^2 f_\mu(x)^t]^{-1}A^t]$$

On a : $(X^{-1}S), \nabla^2 f_\mu(x)$ sont symétriques, donc la somme des deux matrice est symétrique. Ce qui implique :

$$(X^{-1}S)^t + \nabla^2 f_\mu(x)^t = (X^{-1}S) + \nabla^2 f_\mu(x).$$

Donc l'égalité est vraie.

2-la matrice $A[X^{-1}S + \nabla^2 f_\mu(x)]^{-1}A^t$ est définie positive :

Il suffit de montrer que la matrice $X^{-1}S + \nabla^2 f_\mu(x)$ est définie positive :

Comme : $S = Diag(s)$ où $s > 0$, $X = Diag(x)$ où $x > 0$ sont deux matrices diagonales à éléments positifs alors la matrice $X^{-1}S$ est une matrice symetrique diagonale à élément diagonaux ($\frac{s_i}{x_i} > 0$) est définie positive. Mais pour $\nabla^2 f_\mu(x)$ comme $f_\mu(x)$ est supposée convexe, donc sa matrice hessienne semi définie positive. Par conséquent : $X^{-1}S + \nabla^2 f_\mu(x)$ une somme de deux matrices définies positives est définie positive. D'où le système (++) admet une seule solution.

5.3.3.4. Calcul d'une solution initiale strictement réalisable :

Un point est strictement réalisable de $(PNLC)$ est la solution du problème :

$$\{Ax = b, x > 0\} \quad \ldots\ldots\ldots\ldots\ldots\ldots(**)$$

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

$$\begin{cases} Min\ \lambda \\ s.c \\ Ax + \lambda(b - Aa) = b \\ x > 0, \lambda > 0 \end{cases} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(P1)$$

Tel que : $a \in \mathbb{R}^n_{++}$ est choisi arbitrairement dans l'orthant positif et λ une variable artificielle. Posons $\omega = (x, \lambda)$, le problème (P1) est équivalent au problème linéaire suivant :

$$\begin{cases} Min\ c^t\omega = z^* \\ s.c \\ \bar{A}\omega = b \\ \omega > 0 \end{cases} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{ (P2)}$$

Où : $c^t = (0, \ldots, 1) \in \mathbb{R}^{n+1}$, $\bar{A} = [A, b - Aa] \in \mathbb{R}^{m*(n+1)}$.

Le problème (P2) vérifie les hypothèses de Karmarkar :

1) $z^* = 0$.
2) $\omega = (a, 1)$ est une solution strictement réalisable de (P2).
3) $\bar{A}$ est une matrice de plein rang (rang $\bar{A} = m \leq n + 1$).

- Plus précisément Karmarkar démontre le théorème suivant :

Théorème5.3.3.4.1. [2]: $\exists\ \varepsilon > 0$, telle que les deux propositions suivantes sont équivalentes :

1) x une solution de (P1).
2) (P2) admet une solution optimale $\omega = (x, \lambda)/\lambda \leq \varepsilon$.

La résolution de (P1) se ramène donc à celle du problème (P2), auquel on appliquera l'algorithme de Karmarkar, puisque connaissant la valeur optimale et une solution réalisable de ce dernier.

5.3.3.5. Calcul du pas de déplacement :

Pour réduire le nombre d'itération et le temps de calcul, on a introduit une procédure pratique moins couteuse que la recherche pour calculer le pas de déplacement.

Si (x, y, s) est le point courant réalisable, alors le nouvel itéré :

$(x^+, y^+, s^+) = (x, y, s) + \alpha(\Delta x, \Delta y, \Delta s)$ doit être réalisable. Aussi qu'ils vérifient :

$$\begin{cases} x + \alpha_x \Delta x > 0 \\ y + \alpha_y \Delta y > 0 \\ s + \alpha_s \Delta s > 0 \end{cases}$$

On prend : $\alpha = \beta min(\alpha_x^+, \alpha_y^+, \alpha_s^+)$ tel que $0 < \beta < 1$.

Où :

$$\alpha_x = \begin{cases} min -\frac{x_i}{\Delta x_i} \; avec \; i \in I = \{i : \Delta x_i < 0\} \\ 1 \qquad\qquad si \;\; \Delta x_i \geq 0 \end{cases}$$

$$\alpha_y = \begin{cases} min -\frac{y_i}{\Delta y_i} \; avec \; i \in I = \{i : \Delta y_i < 0\} \\ 1 \qquad\qquad si \;\; \Delta y_i \geq 0 \end{cases}$$

$$\alpha_s = \begin{cases} min -\frac{s}{\Delta s_i} \; avec \; i \in I = \{i : \Delta s_i < 0\} \\ 1 \qquad\qquad si \;\; \Delta s_i \geq 0 \end{cases}$$

5.3.3.6. Algorithme de trajectoire centrale avec poids pour la programmation convexe sur un polyèdre :[3]

Début algorithmique :

Données :$\varepsilon > 0$ un paramètre de précision.

1-Initialisation : $(x^0, y^0, s^0) \in T_r(\theta)$, $\mu = \left\| \frac{X^0 S^0 e}{\sqrt{n}} \right\|$, $\eta = \min r_i$ pour i : 1, … , n

$r = \frac{X^0 S^0 e}{u}$, $\delta = \left(1 - \frac{\sqrt{2}}{2}\right)\eta$ *et* $\beta = 1 - \frac{\delta}{\sqrt{n}}$.

$X^0 = diag(x^0), Y^0 = diag(y^0), S^0 = diag(s^0)$,

2-Itération : pour $k = 0,1,2\ldots.$

-si $(x^k)^t R s^k \leq \varepsilon$.stop ! La solution trouvée est optimale.

Sinon :

Pas 1 :

Calculer :

-les paramètres : $\mu^k = \frac{(x^k)^t R s^k}{n}$ et $\tilde{\mu} = \beta \mu^k$

-la direction de Newton $(\Delta x^k, \Delta y^k, \Delta s^k)$ solution du système linéaire :

$$\begin{pmatrix} \nabla^2 f_\mu(x) & -A^t & -I \\ A & 0 & 0 \\ S & 0 & X \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta y \\ \Delta s \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ -XSe + \tilde{\mu} r \end{pmatrix}$$

-calcul du pas de déplacement α_k.tel que : $\alpha_k = (\alpha_x, \alpha_y, \alpha_s)$

Pas 2 :

-le nouvel itéré : $(x^{k+1}, y^{k+1}, s^{k+1}) = (x^k, y^k, s^k) + \alpha_k(\Delta x, \Delta y, \Delta s)$,

Poser :$k = k + 1$

***3-Fin de l'algorithme*.**

5.3.3.7. Convergence de l'algorithme:[3]

Théorème 5.3.3.7.1 : soit $\theta = \delta = \left(1 - \frac{\sqrt{2}}{2}\right)\eta$,suposons que $(x, y, s) \in T_r(\theta)$,

$\grave{u} = \beta u$ tel que $\beta = 1 - \frac{\delta}{\sqrt{n}}$ et $\mu = \frac{x^t R s}{n}$ alors :

1) $(x^+, y^+, s^+) \in T_r(\theta)$
2) $\frac{(x^+)^t R s^+}{n} \leq (1 - \frac{\eta}{6\sqrt{n}}) \frac{x^t R s}{n}$

Preuve :1-On a :$Ax^+ = A(x + \Delta x) = Ax + A\Delta x$, dans la deuxième équation du problème(2) : $A\Delta x = 0$.Donc $Ax^+ = Ax = b$.

De la première équation du problème (2), on a:

$A^t y^+ + s^+ = A^t(y + \Delta y) + s + \Delta s$

$= A^t y + s + (A^t \Delta y + \Delta s) = \nabla f(x) + (A^t \Delta y + \Delta s)$

$= \nabla f(x) + \nabla^2 f(x)\Delta x = \nabla f(x + \Delta x) = \nabla f(x^+)$

De plus, sous les conditions précédentes, on peut montrer que l'on reste dans le voisinage de la trajectoire centrale, c'est-à-dire : $\|X^+S^+e - ur\| \leq \theta u^+$

Il reste à vérifier que :$(x^+, s^+) > 0$.

On a: $|x_i^+ s_i^+ - u^+ r_i| \leq \|X^+S^+e - ur\| \leq \theta u^+$

$-\theta u^+ \leq x_i^+ s_i^+ - u^+ r_i \leq \theta u^+ \Longrightarrow (r_i - \theta)u^+ \leq x_i^+ s_i^+ \leq (r_i + \theta)u^+$

$x_i^+ s_i^+ \geq (r_i - \theta)u^+ \geq (\eta - \theta)u^+ > 0$ car: $\eta > \theta$.

Donc $x_i^+ et\ s_i^+$ sont de même signe, on suppose que $x_i^+ < 0, s_i^+ < 0$:

$x_i^+ = x_i + \Delta x_i \Longrightarrow \Delta x_i = x_i^+ - x_i < 0$

$|\Delta x_i| = -\Delta x_i = -x_i^+ + x_i > -x_i^+ = |x_i^+| \Longrightarrow |\Delta x_i| > -x_i^+$ ... ... (1)

De même on :$s_i^+ = s_i + \Delta s_i \Longrightarrow \Delta s_i = s_i^+ - s_i < 0$

$|\Delta s_i| = -\Delta s_i = -s_i^+ + s_i > -s_i^+ = |s_i^+| \Longrightarrow |\Delta s_i| > -s_i^+$ ... ... (2)

De (1) et (2), on trouve : $\Delta x_i \Delta s_i = (-|\Delta x_i|)(-|\Delta s_i|)$

$= |\Delta x_i||\Delta s_i| > |-x_i^+||-s_i^+| = |x_i^+ s_i^+| = x_i^+ s_i^+ donc: \Delta x_i \Delta s_i > x_i^+ s_i^+$

En contradiction avec: $\Delta x_i \Delta s_i = x_i^+ s_i^+ - u^+ r_i < x_i^+ s_i^+$.

2- On a :$g(\theta^*) = -\left(\sqrt{2} - 1\right)^2 \eta < -\frac{\eta}{6}$

$c = 1 + \frac{1}{\sqrt{n}} g(\theta^*) < 1 - \frac{\eta}{6\sqrt{n}}$.

De notre cote, nous avons montré la polynomialité de la convergence de l'algorithme obtenu que nous montrions dans le théorème suivant :

Théorème: pour $\varepsilon > 0$, l'algorithme converge après $k = O\left(\frac{\sqrt{2}}{(2-\sqrt{2})\eta} |\log \varepsilon|\right)$iterations,

Avec : $\eta = \min(r_i)$.

Preuve :$\forall\ \varepsilon > 0, \exists 0 < \theta < 1, \exists\ (x^+, y^+, s^+) \in T_r(\theta)$

On a: $\mu^{k+1} \leq \beta\mu^k \leq \left(1-\frac{\theta}{\sqrt{n}}\right)\mu^k \; tel\; que: \mu^k = \frac{(x^k)^t Rs^k}{n}$

Alors: $\ln(\mu^{k+1}) \leq k\ln\left(1-\frac{\theta}{\sqrt{n}}\right) + \mu^0$.

On sait que :

$\ln(x+1) \leq x, \forall x > -1. donc:$

$\ln\left(\frac{\mu^k}{\mu^0}\right) \leq k\ln\left(1-\frac{\theta}{\sqrt{n}}\right)\left(mais: \frac{\mu^k}{\mu^0} < \varepsilon\right)$.

Ainsi :

$k\left(-\frac{\theta}{\sqrt{n}}\right) < \ln\varepsilon$

Alors : $k \geq -\frac{\sqrt{n}}{\theta}\ln\varepsilon$. est le nombre d'itérations.

5.3.4. Tests numériques :

Les exemples traités sont de la forme canonique suivante :

$$\begin{cases} Min f(x) \\ sc \\ Ax = b \\ x \geq 0 \end{cases} \ldots\ldots\ldots\ldots\ldots\ldots\ldots (PNLC)$$

Où $f: \mathbb{R}^n \to \mathbb{R}$ est une fonction non linéaire, convexe et différentiable,A une matrice de type (m,n) de plein rang $(\text{rg A} = \text{m} \leq \text{n})$, $b \in \mathbb{R}^m$.

La précision est comprise entre $10^{-3}\; et\; 10^{-6}$.

Pour la recherche des solutions initiales, on présente un algorithme un algorithme réduit de Karmarkar **(Annexe1)**, qui va nous aider à résoudre ce dernier avec la trajectoire centrale avec et sans poids.

1- L'application avec la méthode TC sans poids :

Exemple 01: Soit le programme non linéaire suivant:

$$\begin{cases} Min\; x_1^3 + x_1^3 + x_1 x_2 \\ sc \\ 2x_1 + x_2 + x_3 = 8 \\ x_1 + 2x_2 + x_4 = 7 \\ x_1, x_2, x_3 \geq 0 \end{cases}$$

Les solutions initiales trouvées :

$$x^{\circ} = (3.000000, 1.00000\ ,1.000000, 1.00000\ )^t$$

$$y^{\circ} = (-1.024282\ , -1.971814)^t$$

$$s^{\circ} = (0.020377\ , 0.178774\ , 1.024282\ , 1.\ )^t$$

Qui donnent les solutions optimales après 48 itérations :

$$x^* = (2.9465 \quad 1.1071 \quad 0 \quad 0.8394)^t$$

$$y^* = (\ 6.1626 \quad -0.8987)^t$$

$$s^* = (0.0000 \quad 0.0000 \quad -5.2555 \quad 0.3574)^t$$

Exemple 02 : Soit le programme non linéaire suivant :

$$\begin{cases} Min\ (x_1-1)^2 + (x_2-2.5)^2 \\ sc \\ x_1 - 2x_2 - x_3 = -2 \\ -x_1 - 2x_2 + x_4 = -2 \\ -x_1 + 2x_2 + x_5 = -2 \\ x_1, x_2, x_3, x_4 \geq 0 \end{cases}$$

Pour epsilon=0.00001 et les solutions initiales trouvées égales à :

$x^{\circ} = (2.000000\ , 0\ .0000001, 4.000000\ , 0.0000001\ , 0.0000001)^t$

$y^{\circ} = (0.000013\ , -2.271984\ ,\ 0.000788)^t$

$s^{\circ} = (0.000001\ ,\ 0.000001\ , 0.001443\ , 0.003172\ ,\ 0.000771)^t$

Après 39 itérations, on obtient les solutions optimales suivantes :

$x^* = (2.0000 \quad 0.0000 \quad 4.0000 \quad 0.0000 \quad 0.0000\ )^t$

$y^* = (\ 0.0000 \quad -2.2720 \quad 0.0008)^t$

$s^* = (0.0000 \quad 0.0000 \quad 0.0014 \quad 0.0032 \quad 0.0008)^t$

Qui donnent la valeur optimale $f(x^*) = 2.5$.

Exemple 03: Soit le problème non linéaire suivant :

$$\begin{cases} Min - x_1 * x_2^3 \\ sc \\ x_1 + x_2 + x_3 = 2 \\ x_1 + 5x_2 + x_4 = 5 \\ x_1\ , x_2, x_3, x_4 \geq 0 \end{cases}$$

Pour epsilon=0.00001, et prenons les solutions initiales trouvées :

$x^{\circ} = (0.628088, 0\ .872245, 0.499667, 0.010689)^t$

$y^{\circ} = (-3.097763, -0.137088)^t$

$s^{\circ} = (0.002716, 0.016006, 3.097763, 0.137088)^t$

On obtient après 49 itérations, les solutions optimales suivantes :

$x^* = (1.2500 \quad 0.7500 \quad 0.0000 \quad 0.0000)^t$

$y^* = (-2.3494 \quad -0.6410)^t$

$s^* = (0.0000 \quad 0.0000 \quad 2.3494 \quad 0.6410)^t$

Qui donnent la valeur optimale $f(x^*) = -0.5273$.

2- L'application avec la méthode TC avec poids :

*Exemple 04 :*Soit le problème non linéaire suivant :

$$\begin{cases} Min\ (x_1 - x_2 + 1)^2 + exp(x_1 - x_2) \\ sc \\ 2(x_1 - x_2) + x_3 = 1 \\ x_1, x_2, x_3 \geq 0 \end{cases}$$

Pour epsilon =0.0001, et les solutions initiales égales à :

$$x^{\circ} = (1.000000, 1.000000\ , 1.000000\ )^t$$

$$y^{\circ} = 0.134424$$

$$s^{\circ} = (2.681628, 0.012906, 0.006471)^t$$

On obtient après 33 itérations, les solutions optimales suivantes :

$$x^* = (0.0000 \quad 0.0000 \quad 1.0000)^t$$

$$y^* = 0.1409$$

$$s^* = (2.6687 \quad 0.0258 \quad 0.0000)^t$$

Pour lesquelles on trouve $f(x) = 2$.

*Exemple 05 :*Soit le problème non linéaire suivant :

$$\begin{cases} Min\ (\log\ (x_1 + x_2))^2 \\ sc \\ 2x_1 + 3x_2 + x_3 + 2x_4 = 2 \\ 3x_1 - 2x_3 + x_4 = 0 \\ x_1, x_2, x_3, x_4 \geq 0 \end{cases}$$

Pour epsilon =0.001 et les solutions initiales du problème :

$$x^{\circ} = (0.001069\ , 0.663691\ , 0.002641\ , 0.002074\ )^t$$

$$y^{\circ} = (0.270655\ , -0.546445)^t$$

$$s^{\circ} = (5.098024\ , 0.188035\ , 0.636455\ , 0.005135)^t$$

Les solutions optimales obtenues après 56 itérations, sont :

$$x^* = (0.0011 \quad 0.6654 \quad 0.0016 \quad 0.0000)^t$$

$$y^* = (\ 0.3278 \quad -0.8150)^t$$

$$s^* = (0.2052 \quad 0.0075 \quad 0.0421 \quad 0.1593)^t$$

Qui donnent : $f(x) = 0.1646$.

Exemple 06 : Soit le problème non linéaire suivant :

$$\begin{cases} Min\ f(x) = {x_1}^3 - {x_2}^3 \\ sc \\ x_1 - x_2 + x_3 + x_4 = 3 \\ 2x_1 + x_2 - x_3 + x_4 = 4 \\ x_1 + x_3 + 2x_4 = 5 \\ x_1, x_2, x_3, x_4 \geq 0 \end{cases}$$

Pour epsilon=0.000001, et prenons les solutions de départ trouvées égales à :

$$x^{\circ} = (0.341957\ , 0.004313\ , 0.670979\ , 1.991376\ )^t$$

$$y^{\circ} = (-0.005320\ , 0.274080, 1.222858)^t$$

$$s^{\circ} = (1.234301\ ,\ 0.497742\ , 0.056542\ , 0.011443)^t$$

Les solutions optimales sont :

$$x^* = (0.3333 \quad 0.0000 \quad 0.6667 \quad 2.0052)^t$$

$$y^* = (\ 1.6434 \quad 0.6608 \quad 0.0175)^t$$

$$s^* = (0.0000 \quad 2.9650 \quad 0.0000 \quad 0.0000)^t$$

Avec $f(x) = 0.0370$. La valeur optimale, obtenues après 36 itérations.

*Exemple 07 :*Soit le problème non linéaire suivant :

$$\begin{cases} Min\ 3x_1{x_2}^3 - x_2{x_1}^3 \\ sc \\ 2x_1 + 3x_2 + x_3 + 2x_4 = 2 \\ 3x_1 - 2x_3 + x_4 = 0 \\ x_1, x_2, x_3, x_4 \geq 0 \end{cases}$$

Pour epsilon =0.001. Et soient les solutions initiales trouvées du problème :

$$x^{\circ} = (0.001069\,, 0.663691\ \,, 0.002641\,, 0.002074\ )^{t}$$

$$y^{\circ} = (0.270655\,, -0.546445)^{t}$$

$$s^{\circ} = (5.098024\,, 0.188035\,, 0.636455\,, 0.005135)^{t}$$

Les solutions optimales obtenues sont :

$$x^{*} = (0.0000 \quad 0.66667 \quad 0.0000 \quad 0.0000)^{t}$$

$$y^{*} = \ (0.3324 \quad -0.7737)^{t}$$

$$s^{*} = (5.6642 \quad 0.0000 \quad 0.1202 \quad 0.1089)^{t}$$

Avec la valeur optimale du problème $f(x) = 0$. après 19 itérations.

Complémentaire :

A travers les tests numériques effectuées, on a remarqué l'efficacité pratique de l' algorithmes de trajectoire centrale en terme d'infériorité du nombre pratique du nombre d'itération qui est très réduit, et la réduction de la durée d'exécution (le temps d'exécution constaté est très peu).

6.1. Définition :

Dans ce chapitre, nous allons exposer la partie programmation des méthodes de résolution présentées dans les chapitres précédents, et l'implémentation du logiciel contenant des interfaces claires et accessibles. Nous expliquerons aussi le fonctionnement de ce logiciel afin de faciliter son utilisation.

Le logiciel MATLAB (MATrix LABoratory) est spécialisé dans le domaine du calcul matriciel numérique. Tous les objets définis dans le MATLAB les sont donc au moyen des vecteurs et des matrices/tableaux de nombres. Un ensemble important d'opérateurs et de fonctions de MATLAB de base facilitent leur manipulation et des opérations comme par exemple le produit et l'inversion matricielles (inv), la transposition (') ou encore le calcul des valeurs propres (eig) font partie de la bibliothèque standard. D'autres fonctions servant à la création et à la manipulation de matrices et de tableaux (diag, rand, ones, zeros,linspace) sont également disponibles en nombre.

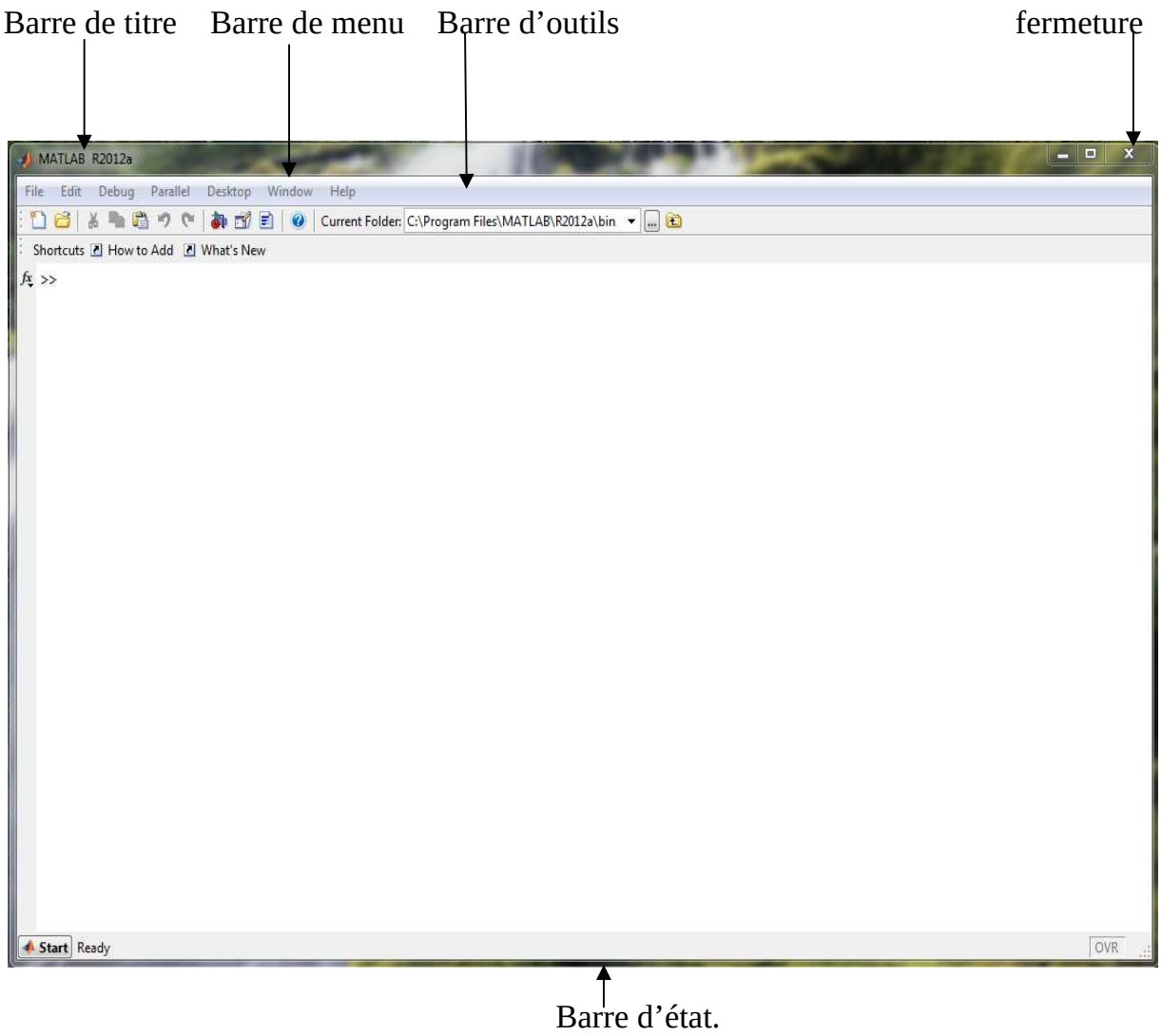
L'environnement MATLAB se présente sous la forme d'un espace de travail (Workspace), ou un interpréteur de commandes exécute des opérations et des fonctions de MATLAB. Les sources de celles-ci sont disponibles, écrites en " langage " MATLAB, voir en C ou en Fortran. L'utilisateur peut les modifier, mais en s'en inspirant, il peut surtout créer et rajouter ses propres fonctions.

Le "langage" MATLAB contient un minimum de structures de programmation (structure itérative, structure conditionnelle, sous-routine) mais reste très rudimentaire. L'avantage est qu'il est très simple et très rapide à programmer, offrant une grande tolérance (syntaxe simple, pas de définition des types, etc.), ce qui permet un gain appréciable en temps de mise au point. L'ingénieur peut par ce moyen être plus efficace dans l'analyse d'un problème, en concentrant ses efforts sur celui-ci et non pas sur l'outil servant à le résoudre.

6.2. Description de la fenêtre MATLAB :

6.2.1. La barre de titre :

La fenêtre MATLAB est surmontée par une barre de titre, contenant à sa gauche une icône et à sa droit les trois boutons <<mise en icône>>, <<minimisation/maximisation>> et <<fermeture>>.



La fenêtre principale de logiciel MATLAB.

6.2.2. La barre de menu :

La barre de menu en contient 5 fenêtres (en général) :

- File (fichier) permet d'obtenir l'éditeur de programme ;
- Edit (Edition) permet de couper / coller dans la ligne de commande et autre ;
- Debug permet l'exécution d'un programme et autres ;
- Window (fenêtre) permet le passage aux différentes du logiciel ;
- Help (aide) accède au menu d'aide.

6.2.3. La barre d'outils :

La barre d'outils en 9, qui sont souvent des raccourcis de fonctions contenues dans les menus. De gauche à droit (entre autre) :

- Ouvrir un nouveau fichier dans l'éditeur ;
- Rappeler un ancien fichier dans l'éditeur ;
- Couper ;
- Copier ;
- Coller ;
- Annuler ;
- Appeler l'aide.

6.2.4. La fenêtre de commande :

Elles se divisent en deux zones :

- La zone historique, qui ne peut être modifiée, mais dont on peut copier des pattiers ;
- La zone de commande éditable.

La zone de commande permet (comme le nom l'indique) de taper une commande qui sera accepté à l'aide de touche <return> ou <entrée>.

6.3. Méthode de travail :

6.3.1. Edition et sauvegarde des fichiers MATLAB :

Dans un premier temps, on peut se contenter d'introduire ses commandes une à une au niveau de l'espace de travail ou elles sont interprétées directement.

Cependant, par la suite, il est beaucoup plus pratique d´écrire sa séquence de commandes complété au moyen d'un éditeur, puis de sauver le tout dans un fichier avec l'extension « .m ». Cette séquence pourra alors être exécutée dans MATLAB par simple introduction du nom du fichier.

6.3.2. Aide en ligne :

En plus de l'aide de Window, une aide en ligne est disponible pour chaque

commande de MATLAB. Il suffit d'introduire : « help nom de commande ».

6.3.4. Création de fichiers de commandes et de fonctions utilisateur :

6.3.4.1. Fichiers de commande (”script files”) :

Un fichier de commande (script file) est un fichier ASCII d’extension « .m » contenant une suite de commandes MATLAB. Il être exécuté directement en tapant simplement son nom dans l’espace de travail MATLAB.

6.3.4.2. Fonctions :

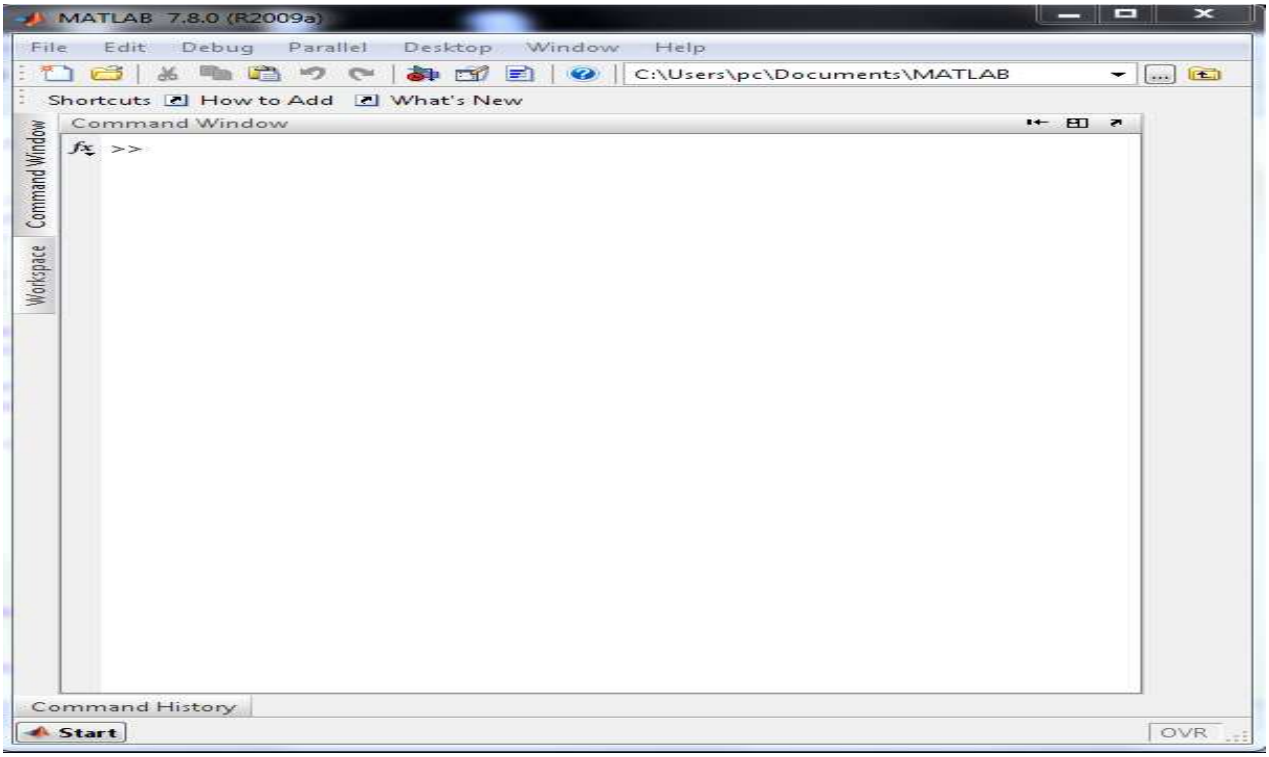
De nouvelles fonctions peuvent être ajoutées à MATLAB par l’utilisateur. Il suffit de créer un fichier de nom

nom_ de_ function.m

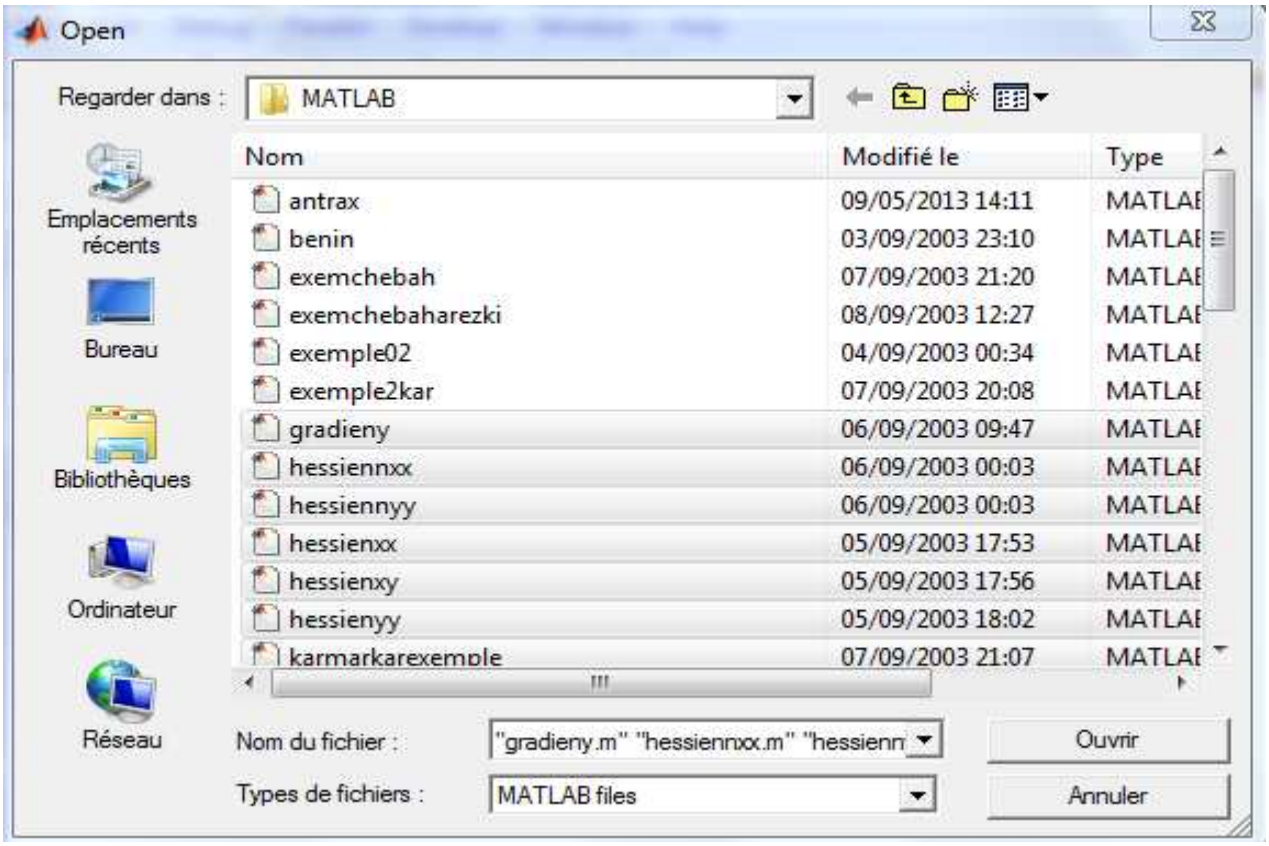
contenant les commandes à exécuter et dont l’entête a le format :

function [liste des arguments de sortie] = nom de fonction (liste des arguments d’entrée). Contrairement aux fichiers de commande, les variables intervenant dans les fonctions sont locales.

Les commentaires documentant les fonctions peuvent être insérés en les faisant précéder du symbole %.*



La fenêtre d’édition de fichier.



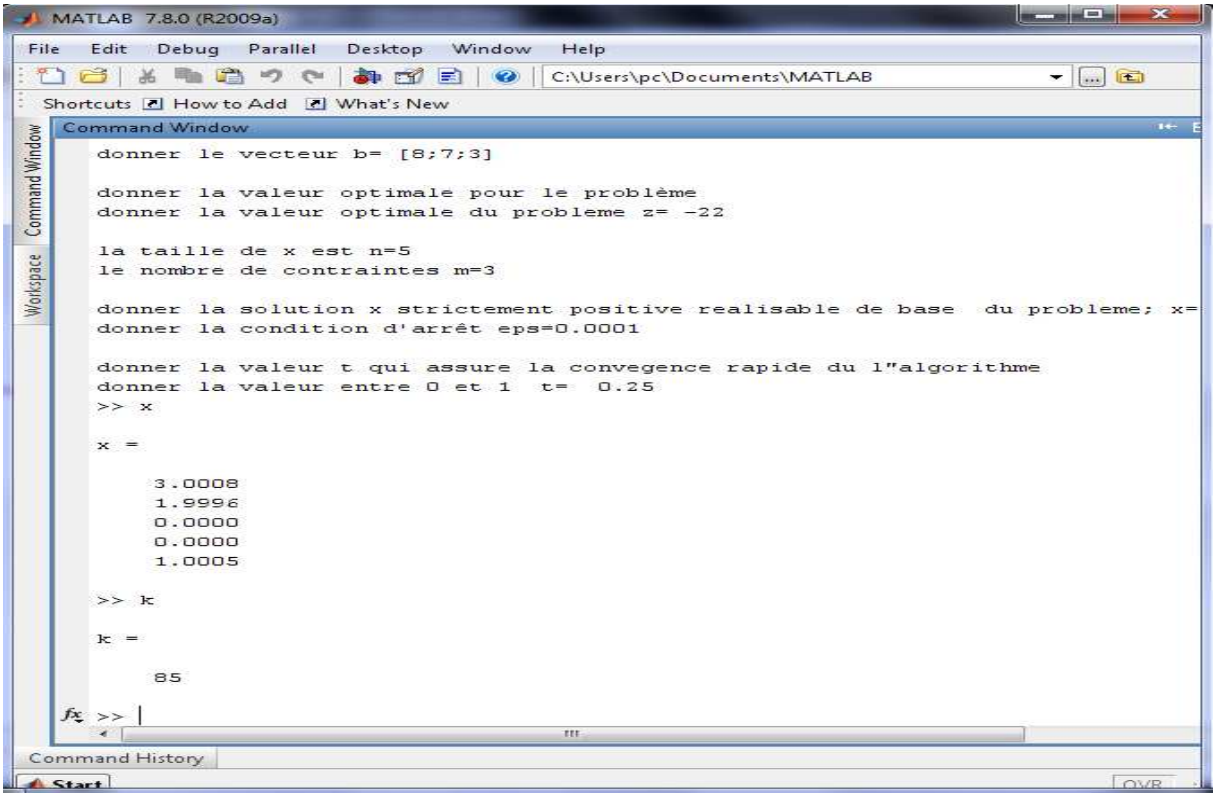
La boite de dialogue d'ouverture de fichiers.

6.3. Exemples d'applications :

6.3.1. la programmation linéaire

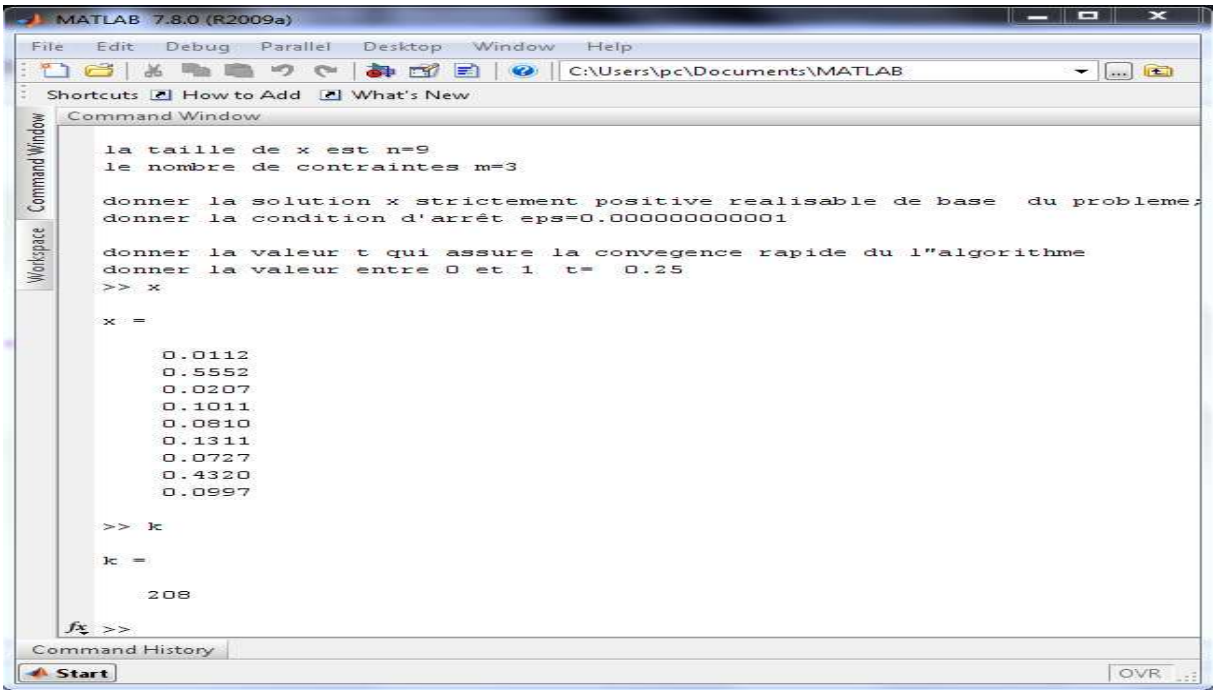
Apres avoir exécuter le programme de karmarkar dans la fenêtre de commande « command window » avec la commande « run », le programme demande des entrées telles que la matrice **A,b, et c** et la solution initiale du problème, on aura les solutions optimales des problèmes illustrées dans les fenêtres suivantes :

Exemple 01 :



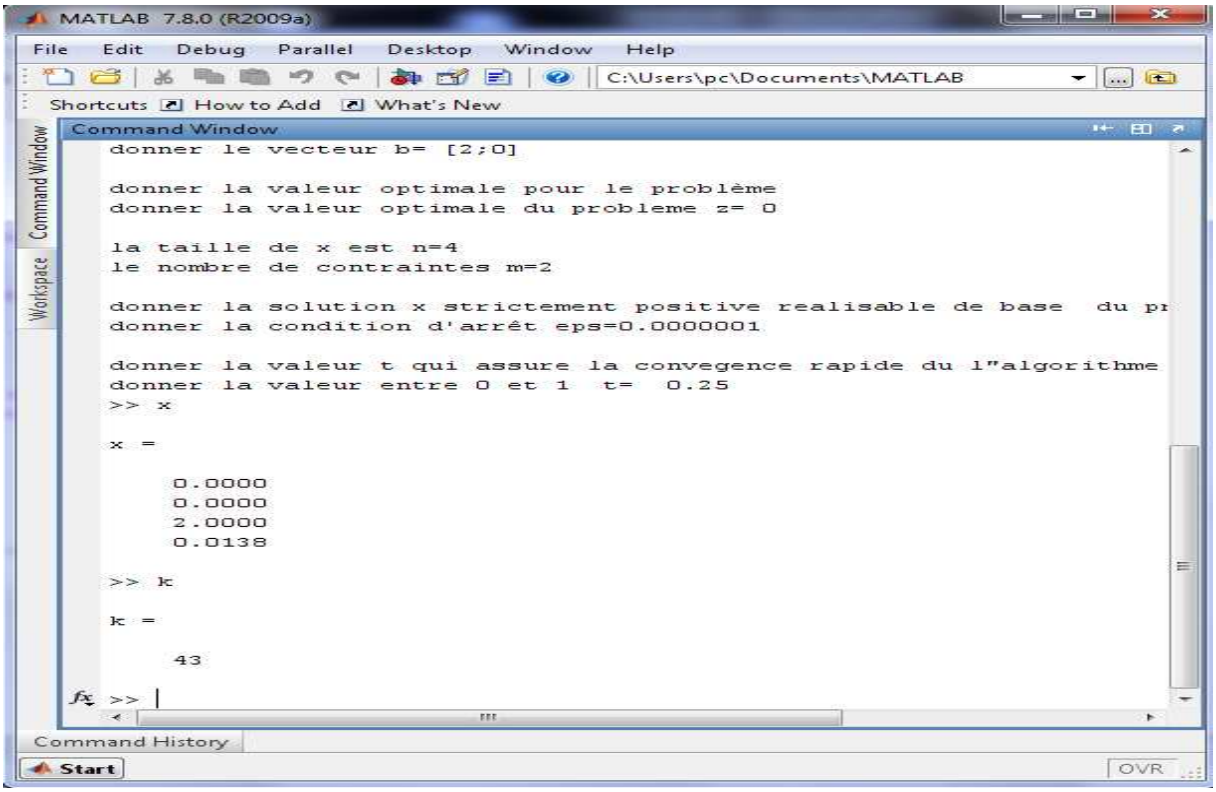
La fenêtre de commande MATLAB.

Exemple 02 :



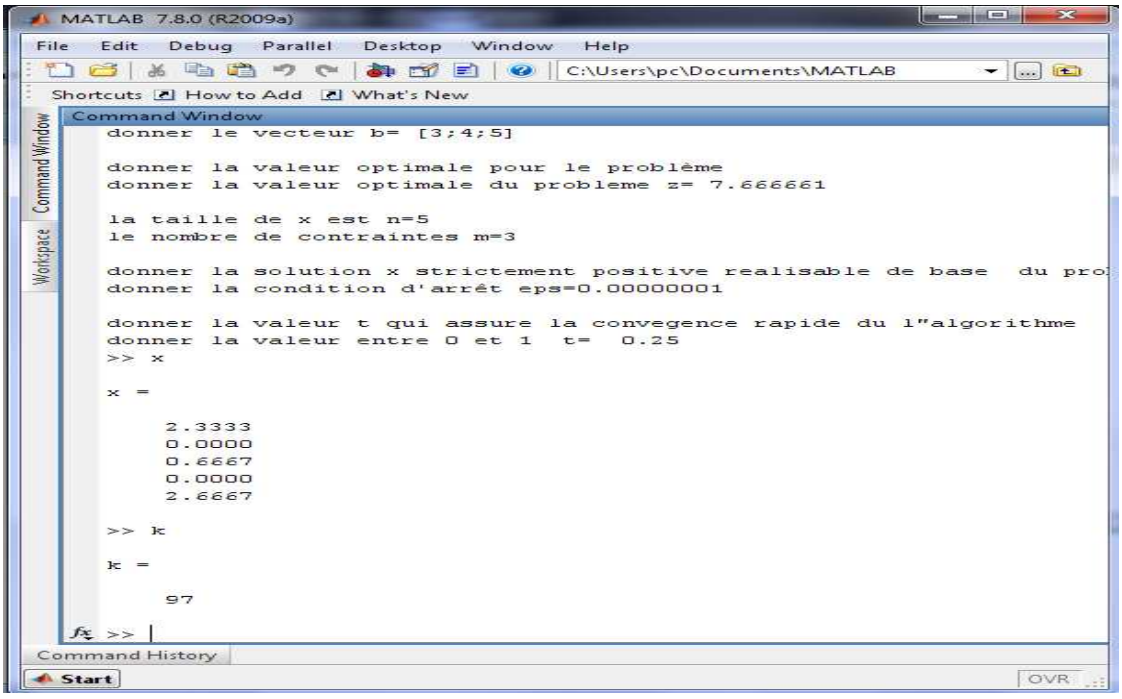
La fenêtre de commande MATLAB.

Exemple 03 :



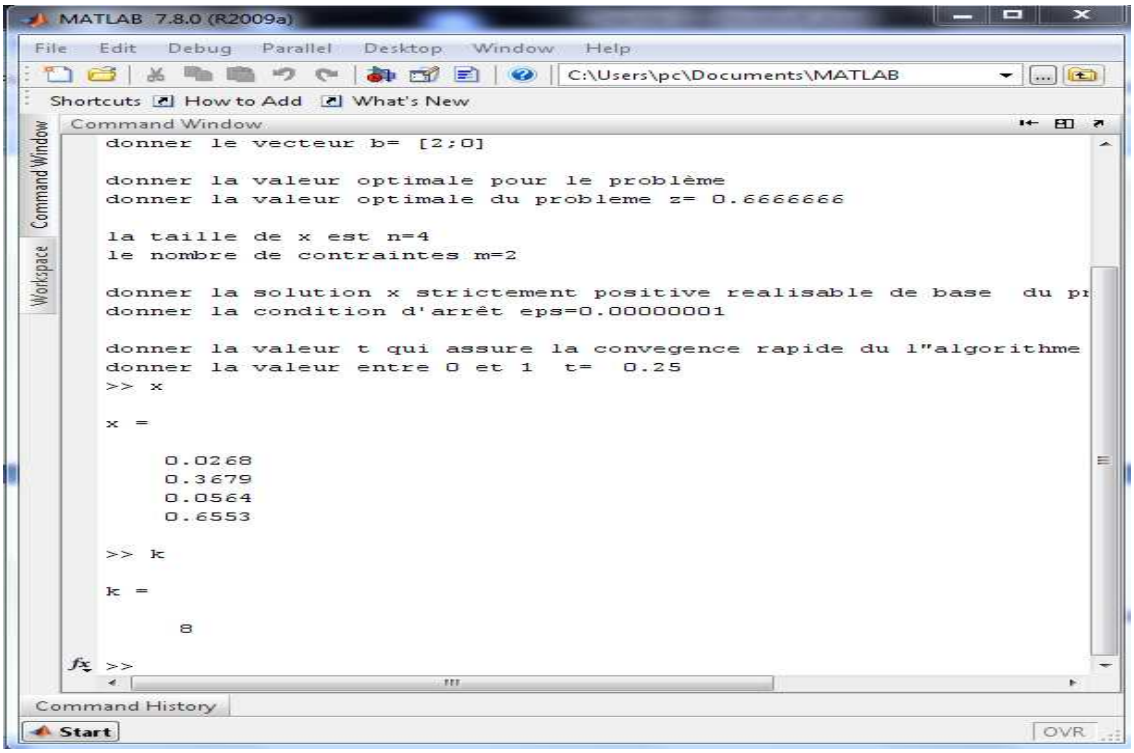
La fenêtre de commande MATLAB.

Exemple 04 :



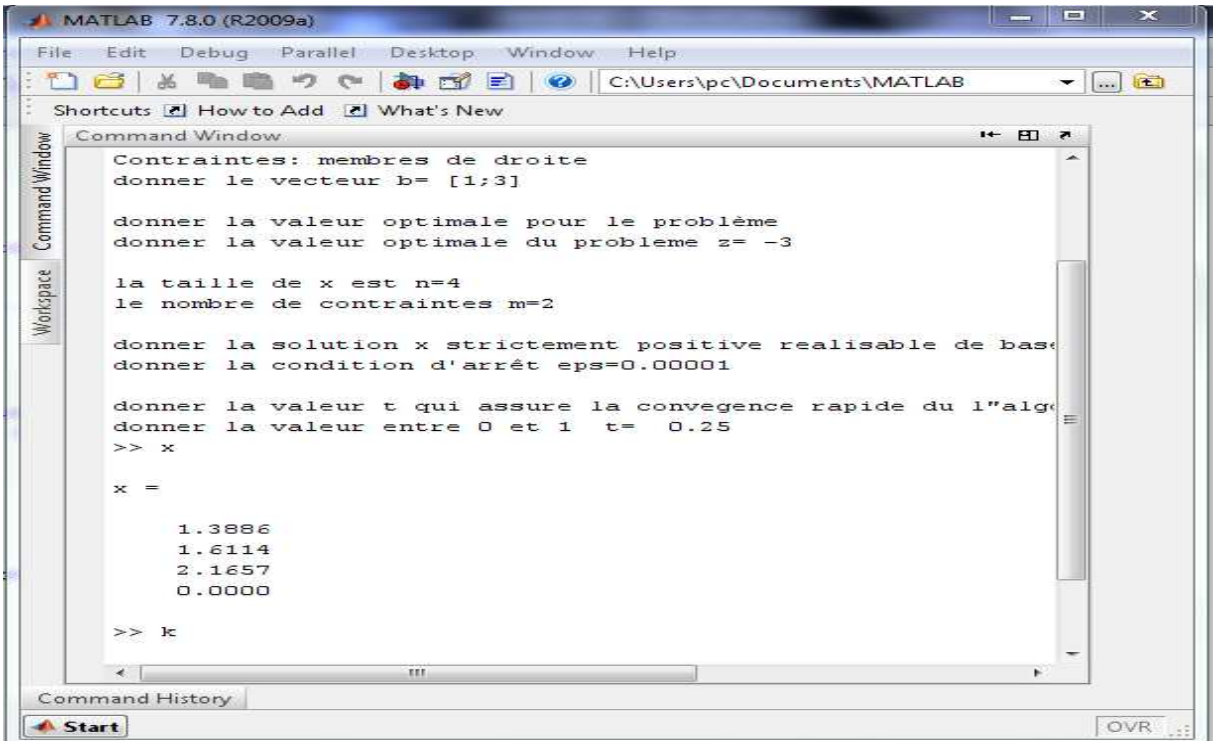
La fenêtre de commande MATLAB.

Exemple 05 :



La fenêtre de commande MATLAB.

Exemple 06 :



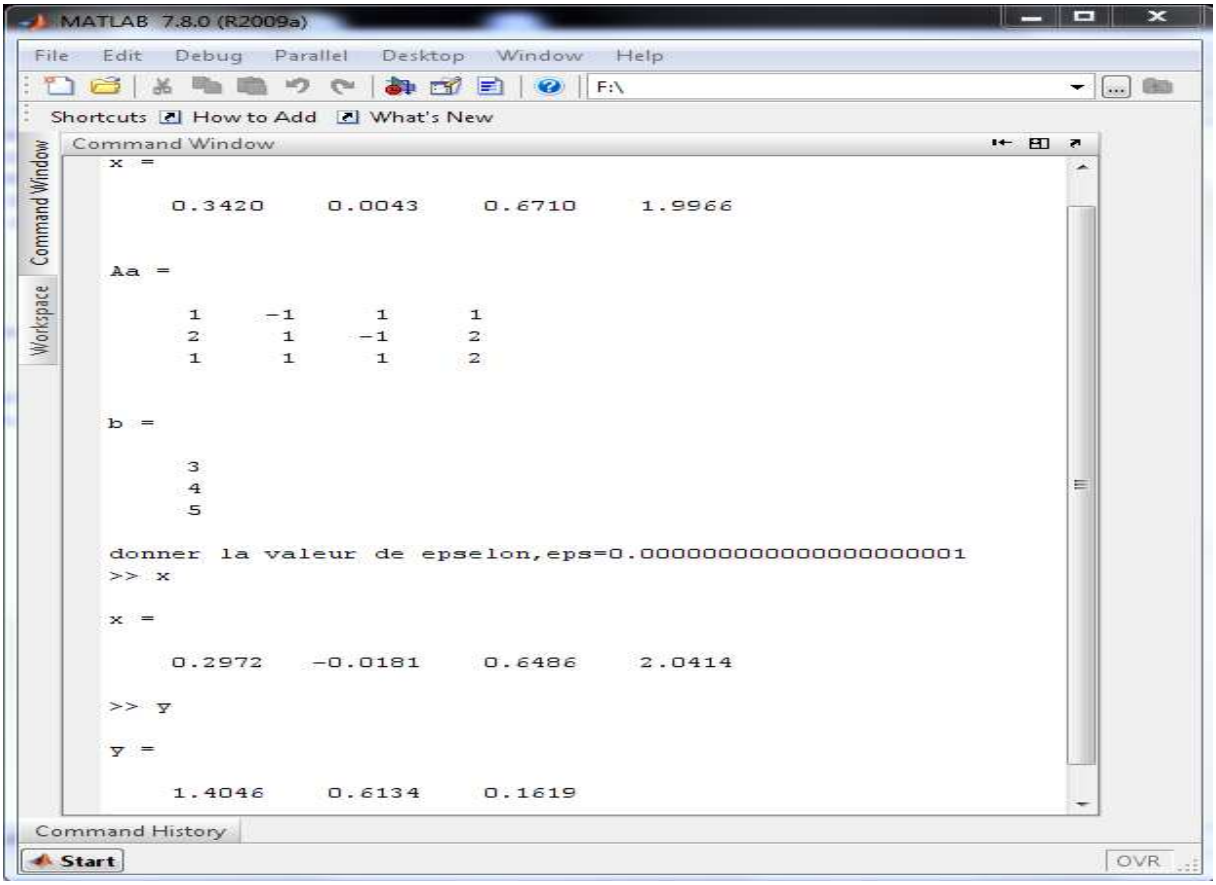
La fenêtre de commande MATLAB.

6.3.2. la programmation quadratique

6.3.2.1. L'application avec la méthode de trajectoire centrale sans poids :

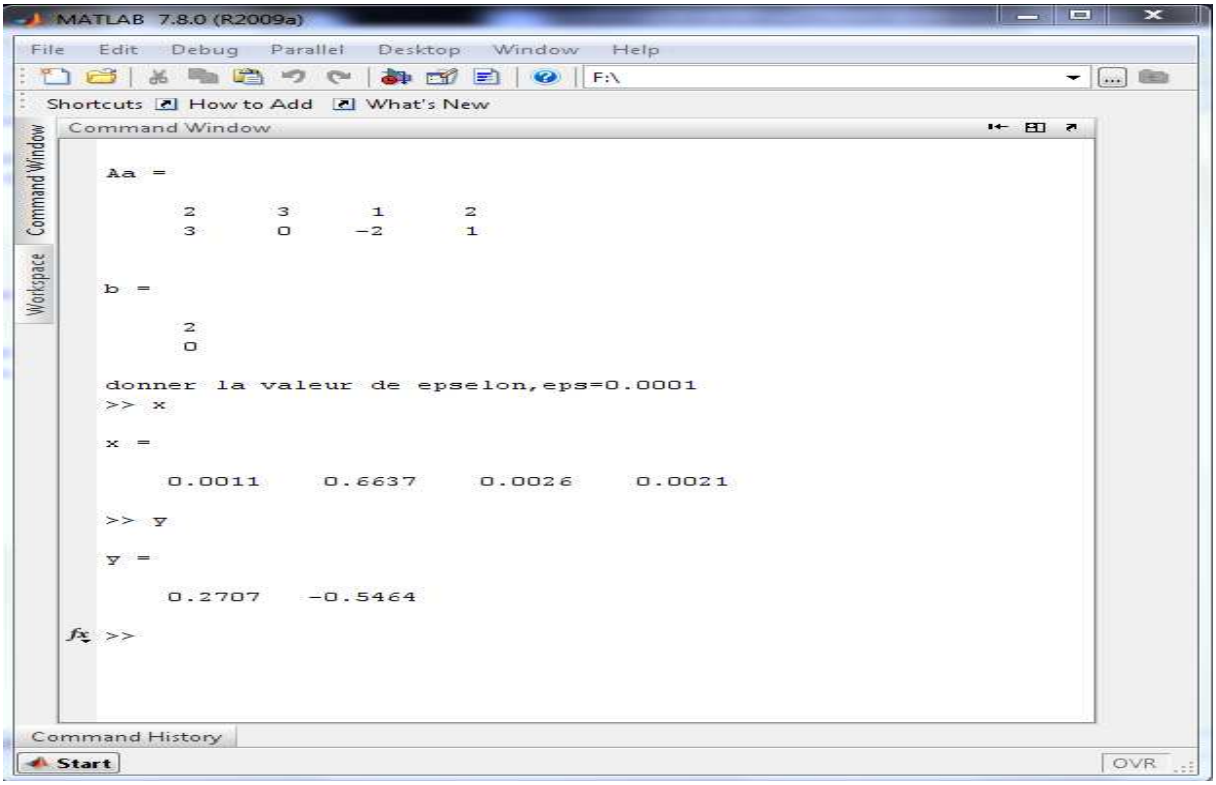
Après exécution de programme de la trajectoire centrale dans la fenêtre de commande « command window » avec la commande « run », le programme demande des entrées telles que les solutions initiales du problème obtenues par Karmarkar, on obtient les solutions optimales des problèmes illustrées dans les fenêtres suivante :

Exemple 01 :



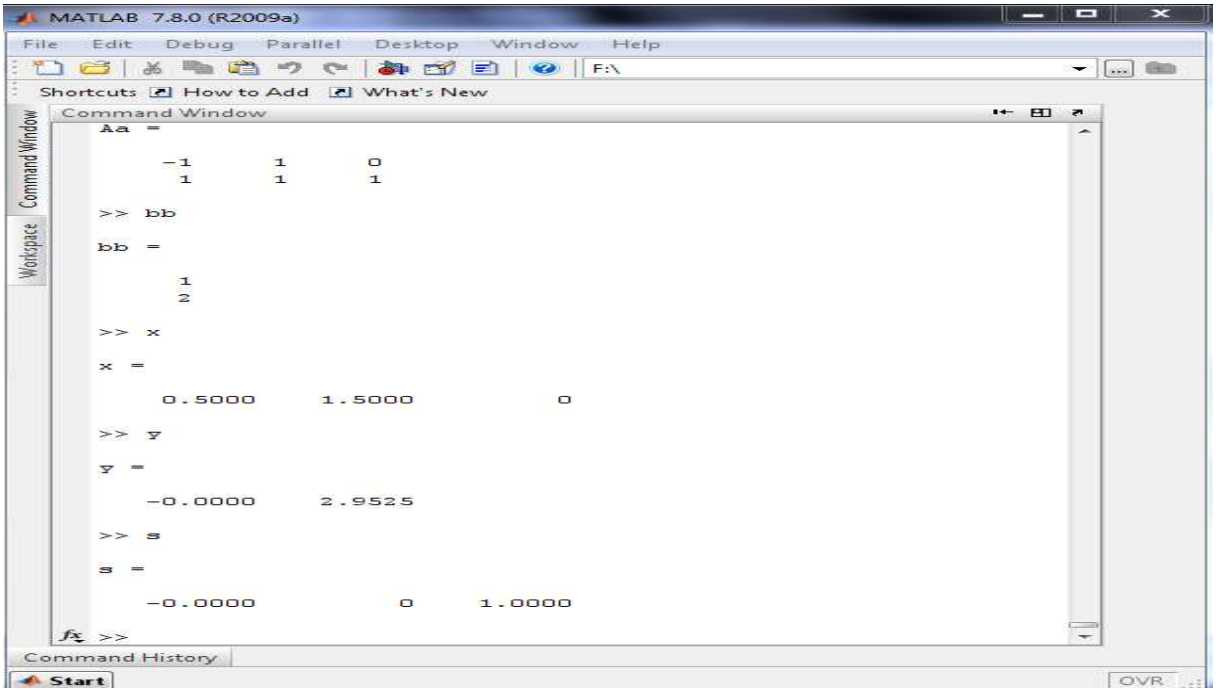
La fenêtre de commande MATLAB.

Exemple 02 :



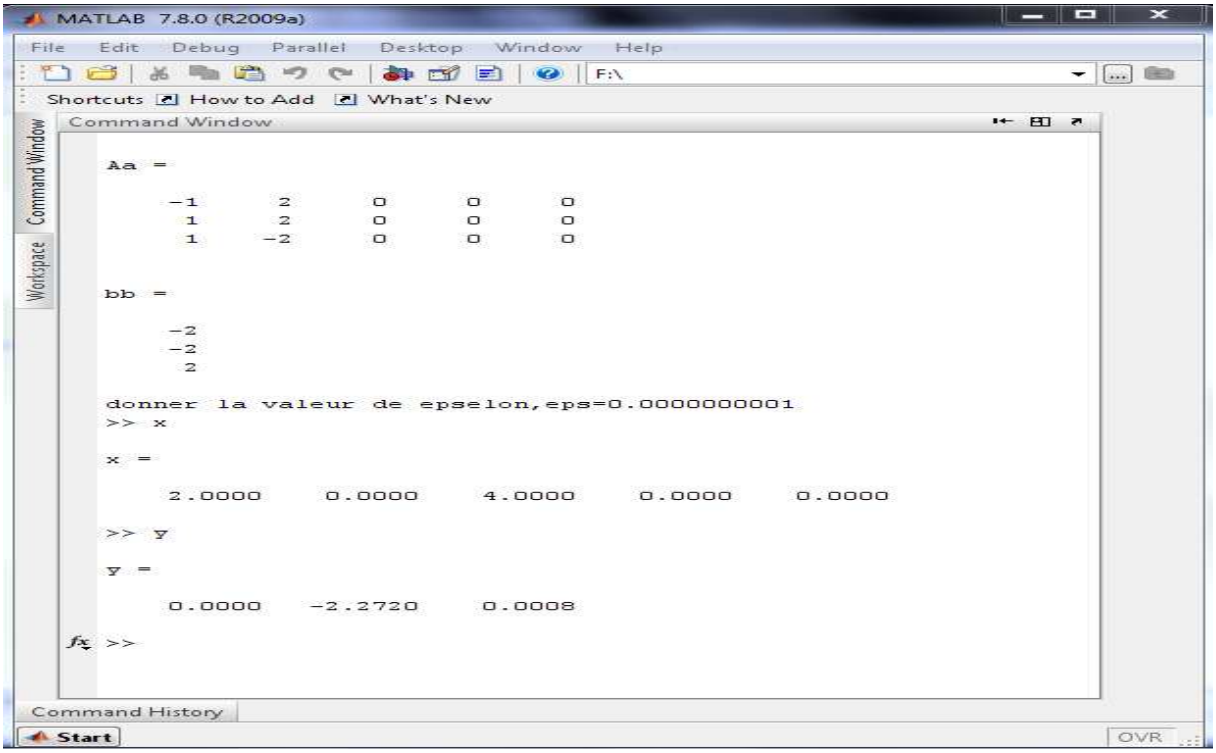
La fenêtre de commande MATLAB.

Exemple 03 :



La fenêtre de commande MATLAB.

Exemple 04 :

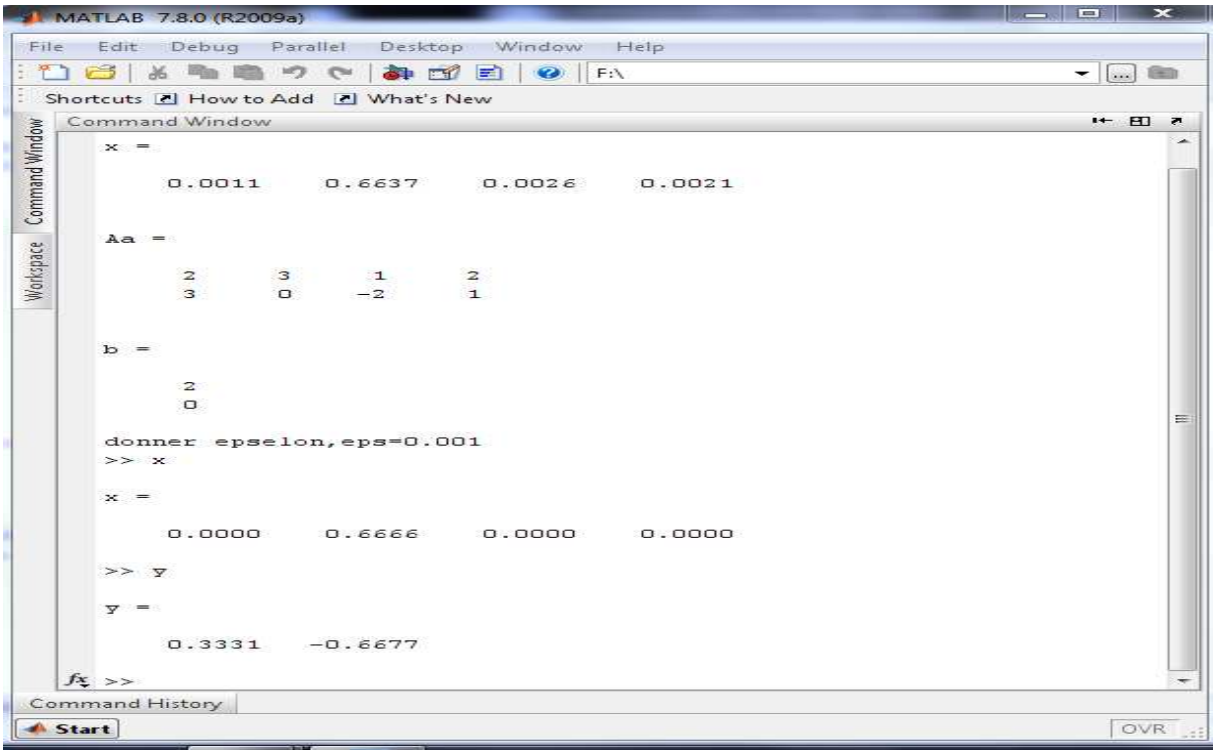


La fenêtre de commande MATLAB.

6.3.2.2. L'application avec la méthode de trajectoire centrale avec poids :

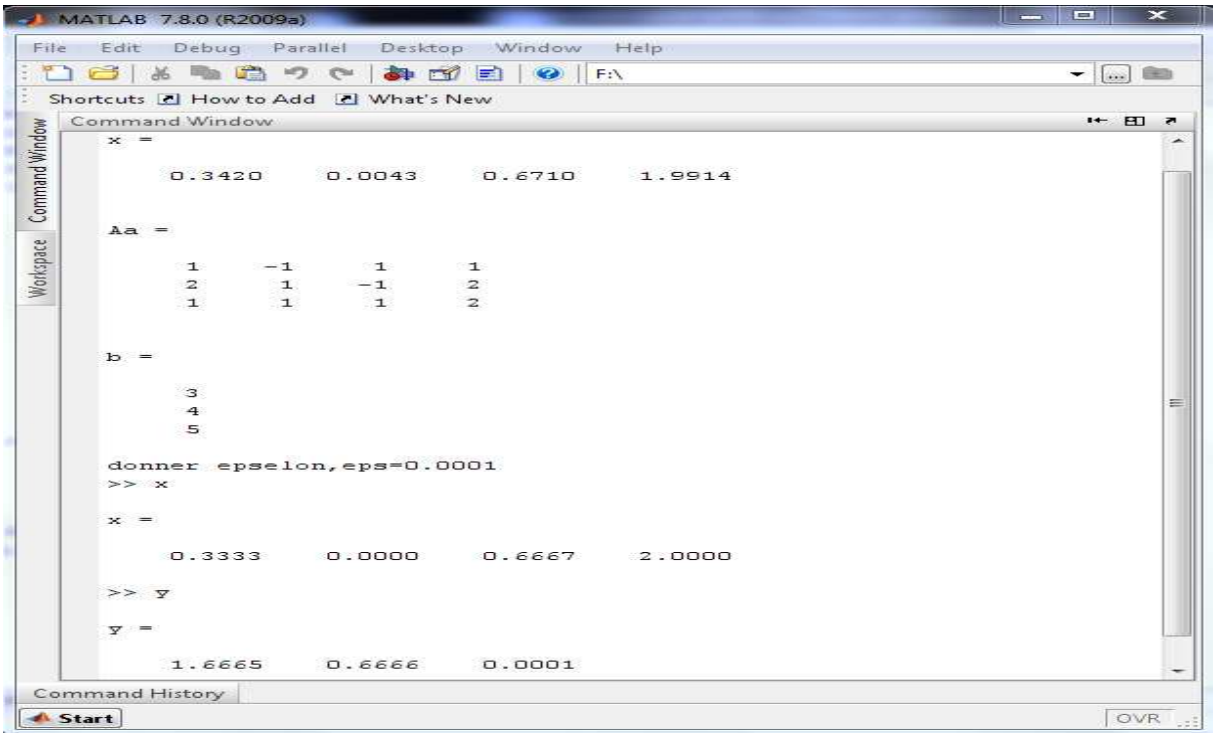
Après avoir exécuter le programme le programme de la trajectoire avec poids dans la fenêtre de commande « command window » avec la commande « run », le programme demande des entrées telles que les solutions initiales du problème obtenues par Karmarkar, on aura les solutions optimales des problèmes montrées dans les fenêtres suivantes :

Exemple 05 :



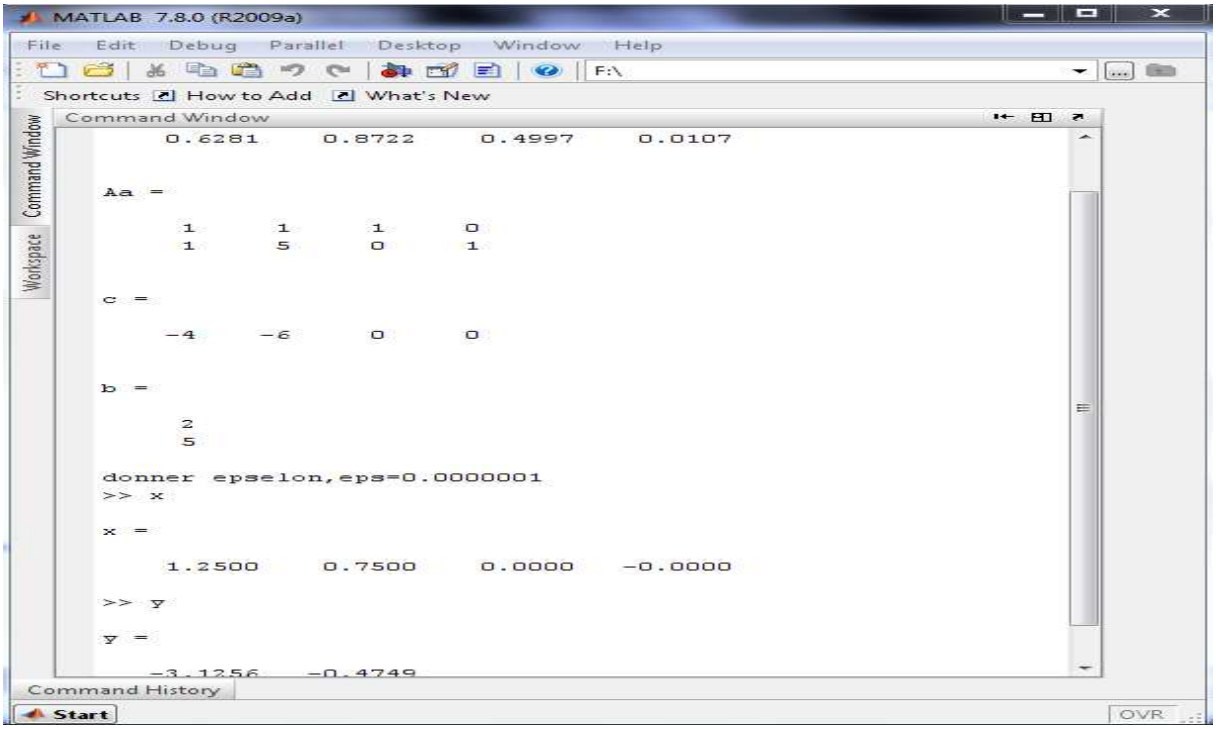
La fenêtre de commande MATLAB.

Exemple 06 :



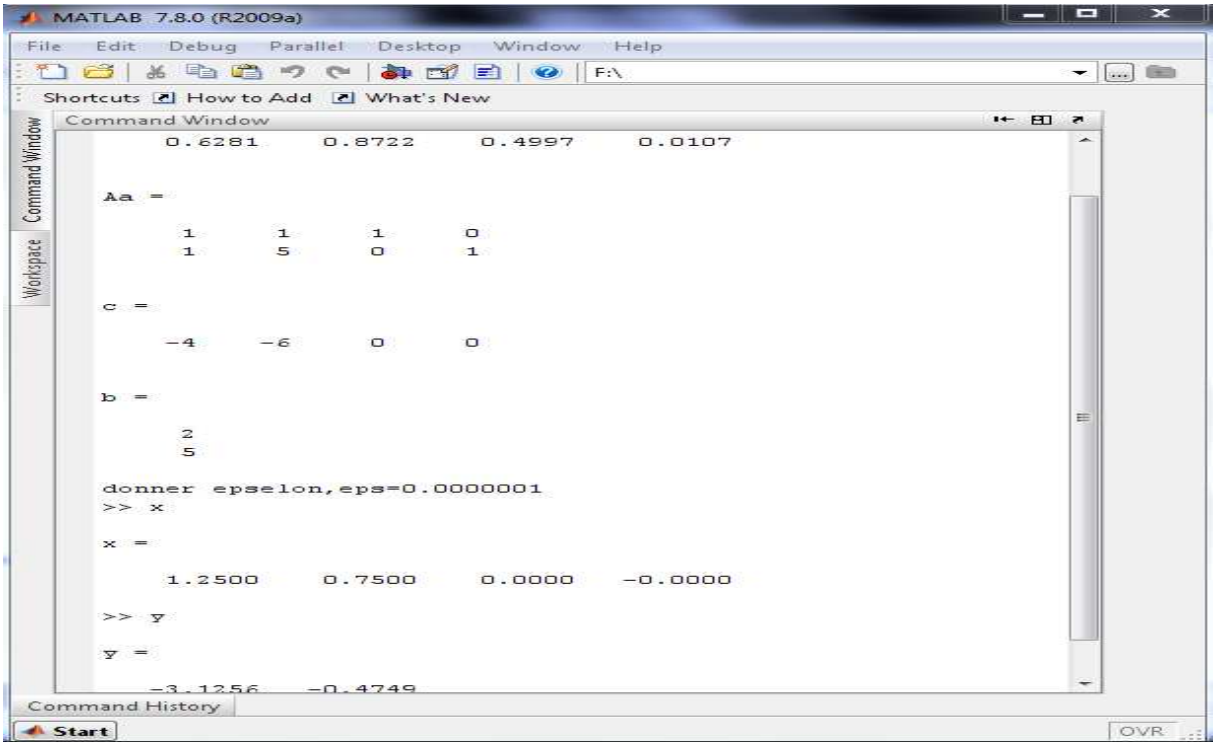
La fenêtre de commande MATLAB.

Exemple 07 :



La fenêtre de commande MATLAB.

Exemple 08 :



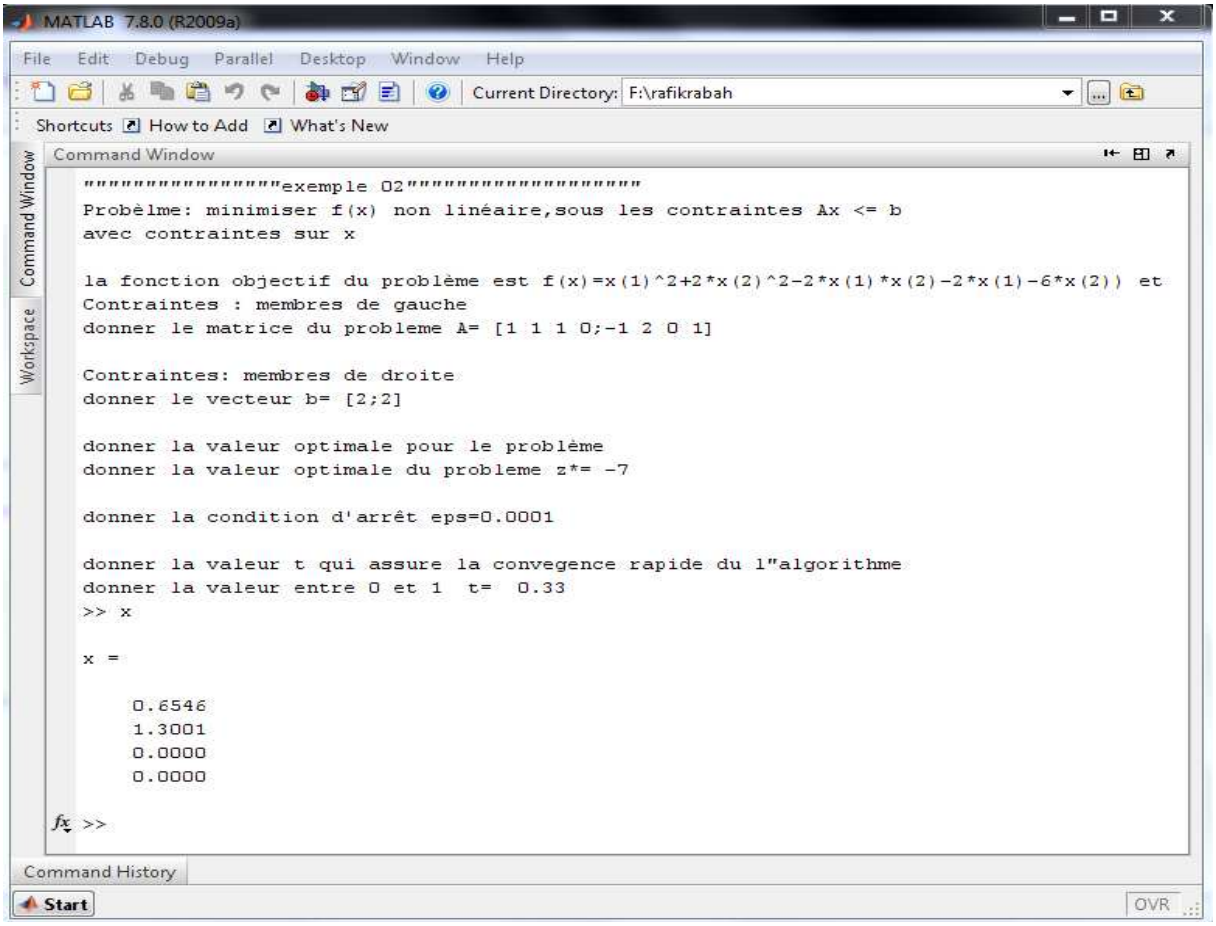
La fenêtre de commande MATLAB.

6.3.1. L'application dans la programmation convexe

6.3.1.1. L'application avec le programme de Karmarkar :

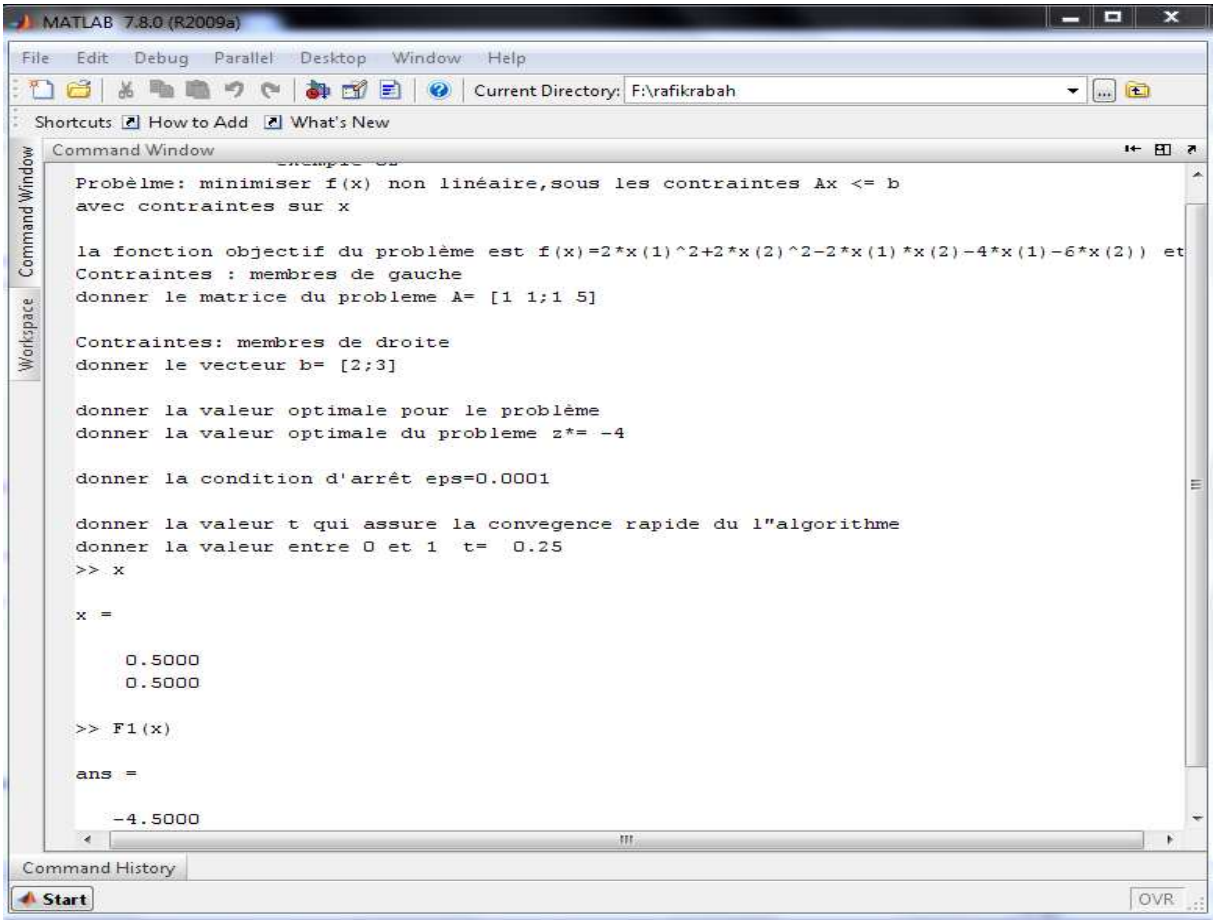
Après l'exécution du programme de Karmarkar dans la fenêtre de commande « command window » avec la commande « run », le programme demande des entrées telles que la matrice **A,** le vecteur **b, et f(x) la fonction objectif** et la solution initiale du problème strictement positive, on aura les solutions optimales des problèmes illustrées dans les fenêtres suivantes :

Exemple 02 :



La fenêtre de commande MATLAB.

Exemple 03 :



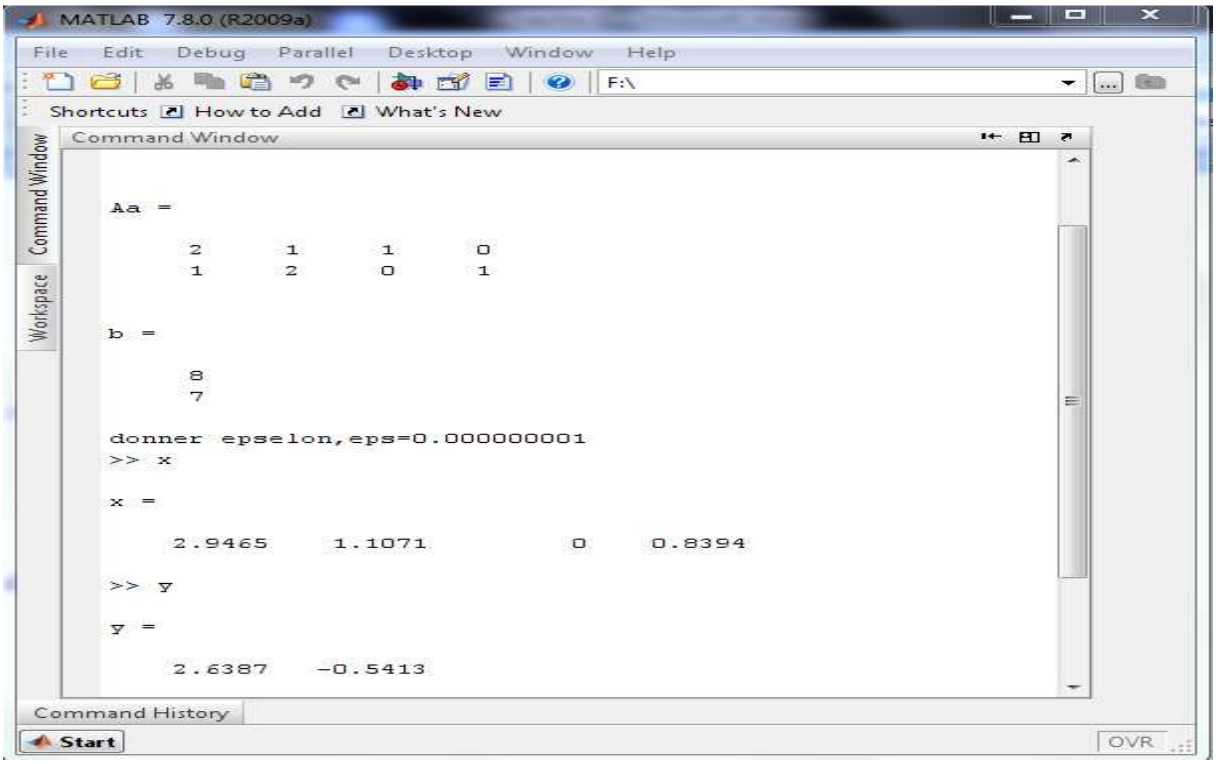
La fenêtre de commande MATLAB.

6.3.3. L'application avec la méthode de la trajectoire centrale :

6.3.3.1. La trajectoire centrale sans poids :

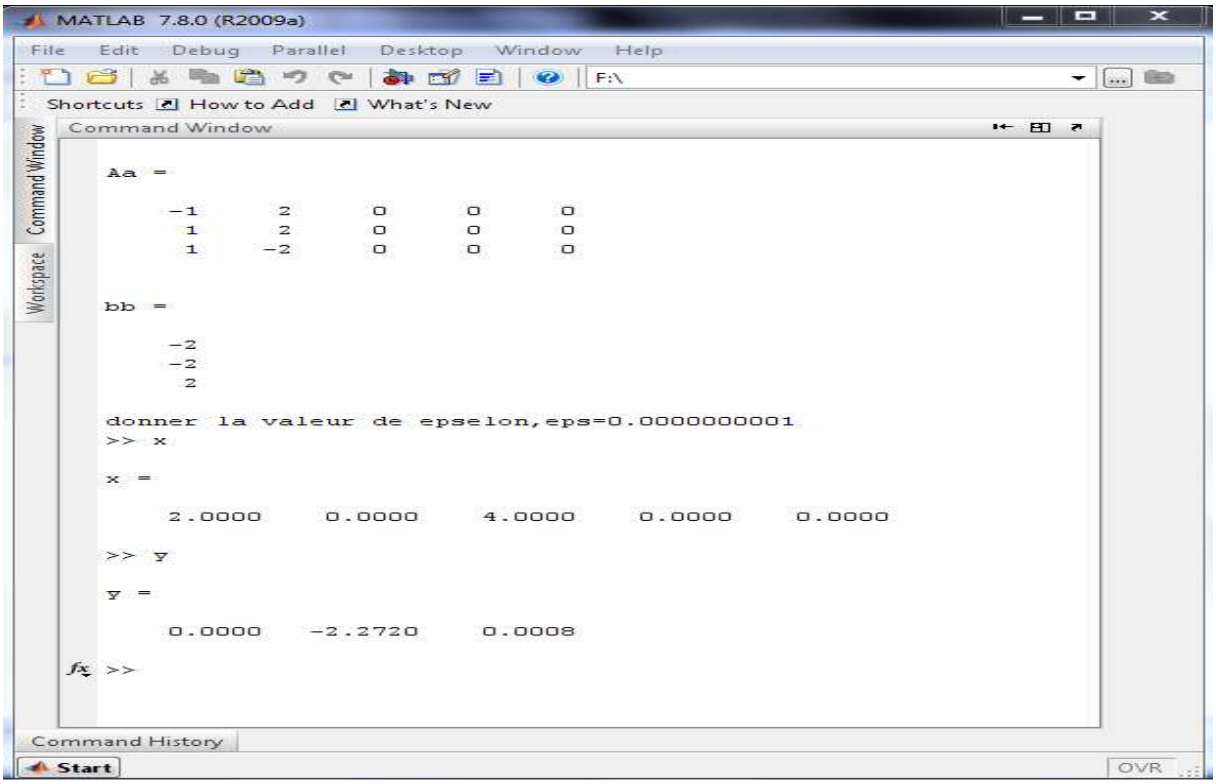
Après avoir exécuter le programme de la trajectoire centrales sans poids dans la fenêtre de commande « command window » avec la commande « run », le programme demande des entrées telles que les solutions initiales du problème obtenues par Karmarkar, on aura les solutions optimales des problèmes illustrées dans les fenêtres suivantes:

Exemple 01 :



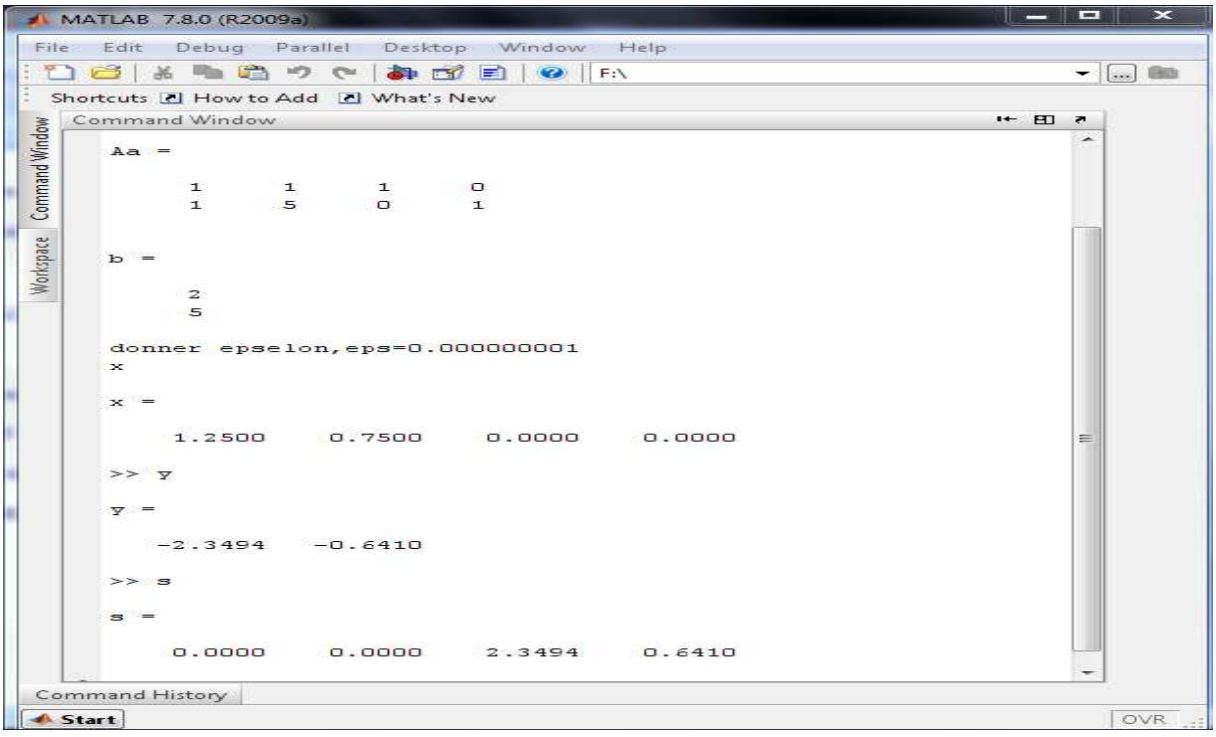
La fenêtre de commande MATLAB.

Exemple 02 :



La fenêtre de commande MATLAB.

Exemple 03 :

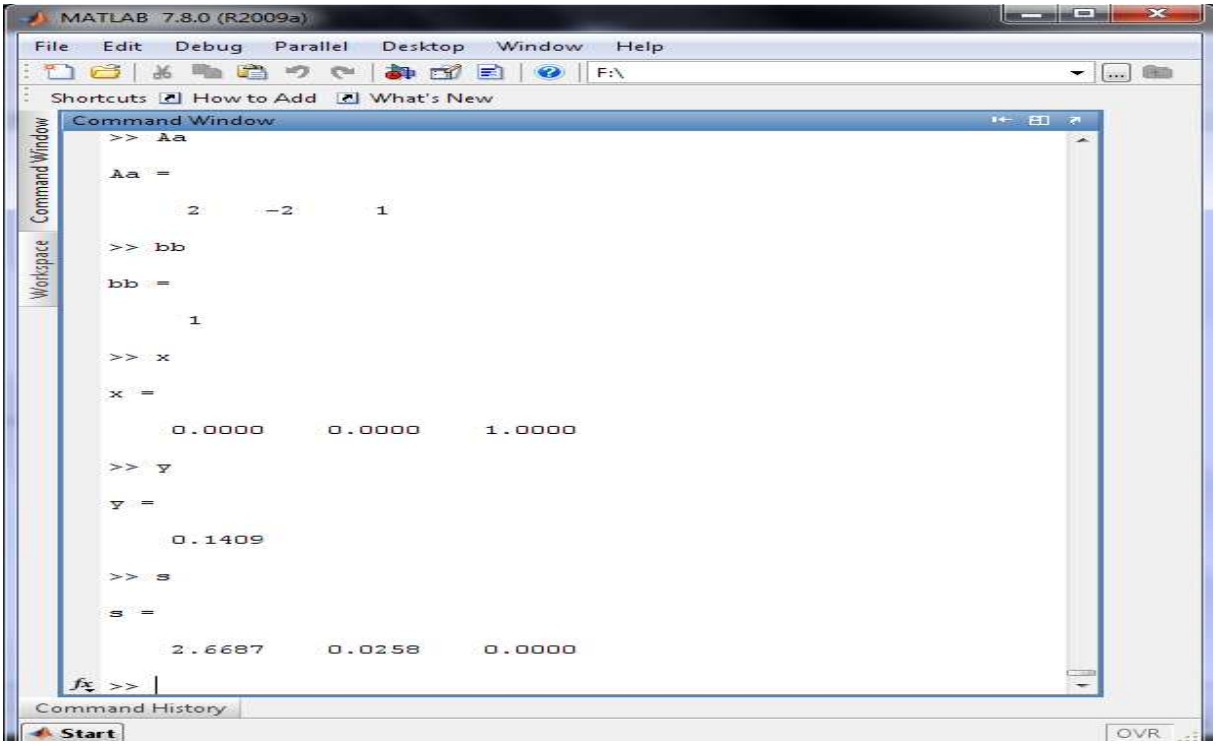


La fenêtre de commande MATLAB.

6.3.3.2. L'application avec la méthode de trajectoire centrale avec poids :

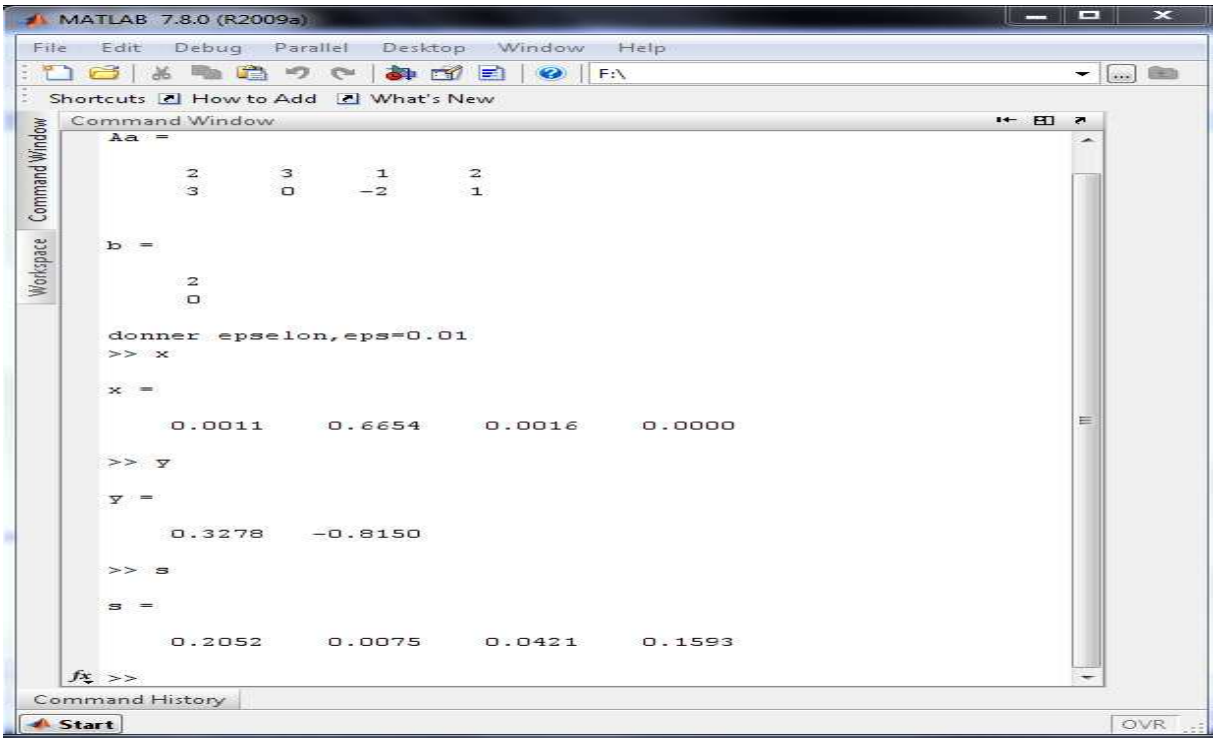
Après l'exécution du programme dans la fenêtre de commande « command window » avec la commande « run », le programme demande des entrées telles que les solutions initiales du problème obtenues par Karmarkar, on aura les solutions optimales des problèmes illustrées dans les fenêtres suivantes :

Exemple 01 :



La fenêtre de commande MATLAB.

Exemple 02 :

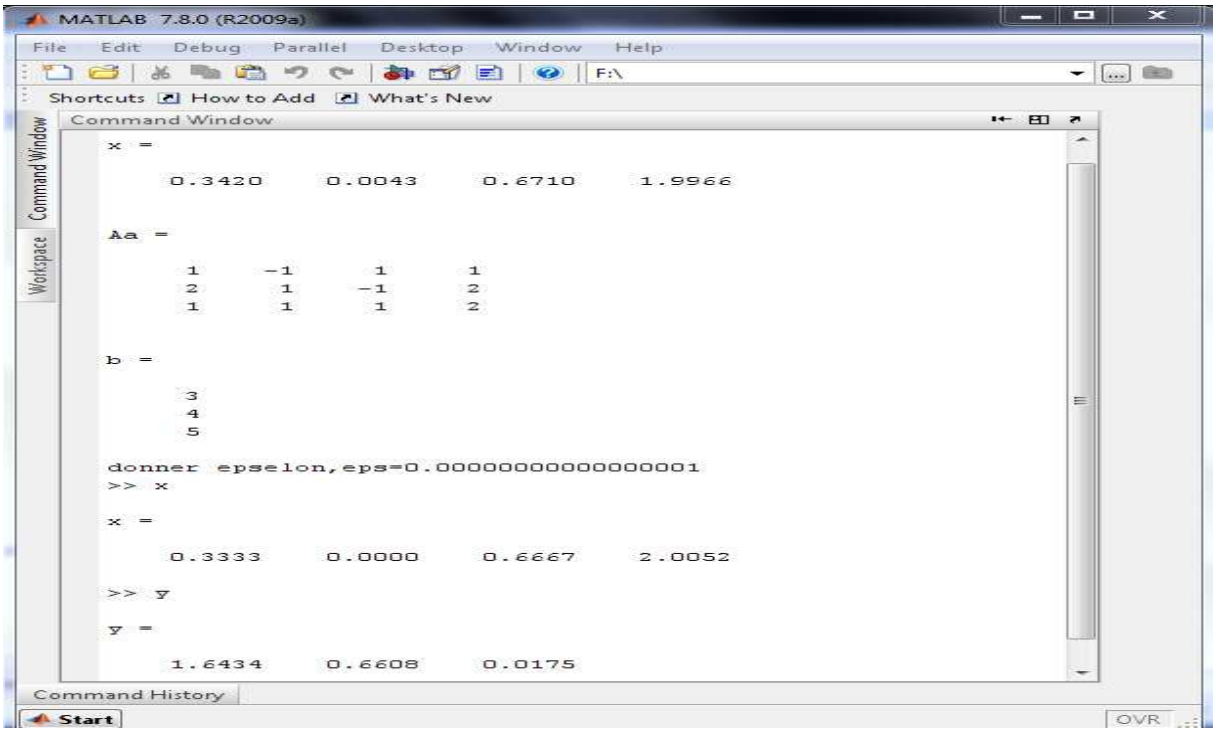


La fenêtre de commande MATLAB.

Exemple 03 :

La fenêtre de commande MATLAB.

Exemple 04 :



La fenêtre de commande MATLAB.

-Conclusion :

Notre étude a porté essentiellement sur une étude des problèmes convexes sous contraintes linéaires, puis une implémentation de logiciel menant à leurs résolutions par des techniques numériques modernes. On a exposé les résultats théoriques de l'optimisation convexe sous contraintes linéaires, notamment au sujet des conditions d'optimalité. Les méthodes des points intérieurs sont connues par leurs efficacité, rapidité de convergence pour résoudre les problèmes de grande taille, l'inconvénient majeur de ce type de méthodes réside dans l'initialisation, c'est-a-dire la détermination d'un point initial qui se trouve à l'intérieur de domaine réalisable. Théoriquement, on suppose que ce point est connu, mais numériquement l'obtention de ce point initial présente un handicap majeur pour ces méthodes importantes.

La recherche de ce point initial pour la méthode de trajectoire centrale nécessite par exemple l'intégration des méthodes de type projectif, la forme réduit de la méthode de Karmarkar[annexe], ce qui réduit l'exploitation de cette méthode grâce a l'augmentation énorme de cette méthode.

Dans l'étude numérique, On a abordé la méthode du simplexe pour la résolution des problèmes linéaires, puis la méthode de Karmarkar pour la résolution de ce dernier. Par la suite, la méthode de trajectoire centrale (sans et avec poids) qui couvre la programmation linéaire et quadratique par le problème de complémentarité linéaire.

A la fin, une généralisation de ces deux méthodes proposées des points intérieurs pour la minimisation des fonctions convexes sous contraintes linéaires. Les méthodes de résolutions issus caractérisent le mode de raisonnement sur la complexité du problème, leur variété entre exactes et approchées est dû à la variété et la complexité des déroulements des opérations, l'implémentation nous a permis d'approfondir nos connaissances dans le domaine de la programmation informatique et de découvrir la richesse démesuré de l'environnement de développement utilisé.

Annexe :

Algorithme (Ye-lustig)

Debut

Donner le paramètre de précision $\boldsymbol{\varepsilon > 0}$, $\boldsymbol{\acute{x}^0 = (x^0, \lambda^0)}$,$\boldsymbol{\acute{c} = (0, \dots, 0, 1)^t}$

$\boldsymbol{k = 0}$

a) Si ($\boldsymbol{\|Ax^0 - b\| < \varepsilon}$) **alors :**

Stop, la solution $\boldsymbol{x^0}$ est une solution approximative et fin.

b) Sinon :

Si $\boldsymbol{\lambda^k > \varepsilon}$ **alors**

Poser :

$$\boldsymbol{D_K = diag(\acute{x}^k)}$$

$$\boldsymbol{B = [A, b - Aa]}$$

$$\boldsymbol{r = \frac{1}{\sqrt{(n+1)(n+2)}}}$$

$$\boldsymbol{B_K = [BD_K, -b]}$$

$$\boldsymbol{p_k = [I - B_k^t(B_k B_K^t)^{-1} B_k][D_k \acute{c}, -\acute{c}^t \acute{x}^k]^t}$$

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

Prendre :

$$\boldsymbol{y^{k+1} = \frac{e_{n+2}}{n+2} - \alpha_k r d_k}$$

$$\boldsymbol{\acute{x}^{k+1} = \frac{1}{y^{k+1}[n+1]} D_K y^{k+1}[n+1].}$$

Poser :

$$\boldsymbol{\lambda^k = \acute{x}^{k+1}[n]}$$**,et aller à b).**

Fin si ;

Fin si ;

Fin.

[1]: *CC.Gonzaga, Path* following method for linear programming. Siam Revieu, M(2) : 1992.

[2]: N. Karmarkar, New polynomial-times algorithm for linear programming. In Proceeding of the 16th Annual A CM, Symposium on Weory of computing.

[3]:Z kebbich , études et extensions d'algorithmes de points intérieurs pour la programmation non linéaire, thèse de doctorat d'état, université FERHAT ABBAS-SETIF(2007).

[4]:M.Godran, M.Minoux, programmation mathématique, 2008.

[5]: M. Kojima S. Mizuno , and A. Yoshise, A polynomial algorithm for a class of linear complimentary problems, Mathematical Programming, 44 1989.

[6]:R.D.C. Monteiro and I.Alder, interior path following primal-dual algorithms ,part 1 : Linear programming. Mathematical Programming, 44 : 1989.

[7] :Dominique de Werra. Thomas M.Liebling.Jean-François Hêche, Recherche opérationnelle pour ingénieurs, tome 01, ISBN 2-88074-446-, (2003),page68-72.

[8]:Renato D.C MONTEIRO and ilan ADLER, interior path following primal-dual algorithme .PART 1:linear programming 44(1989) 27-41 page: 3,7.

[9] :GS,GERAD en collaboration avec le département des mathématiques et génie industriel, école polytechnique de Montréal, introduction au méthodes des points intérieurs, février 2001.

[10]: Robert M.Freund, projective transformations for interior point methods, Part 02: analysis of An Algorithm for finding the weighted center of polyhedral system. Sloan W.P NO. 2050-8,Octobre 1988.

[11] :M.OURIEMCHI , programmation de problème non linéaire par les méthodes des points intérieurs.

[12] :S.J.Wright & Y.Zhang, "A super quadratic-interior-point method for linear complementarily problems", mathematical programming 73(1996).

[13]:Michel minoux, Programmation mathématique,théorie et algorithme 2eme édition,TEC et DOC 2009.

[14]: Michel Bierlaire introduction à l'optimisation différentiable, presse polytechnique et universitaire romandes 2006.

[15]: Aderian Biran & Mosche Breiner, MATLAB pour l'ingénieur, version 06 et 07 édition Pearson(2004).

[16]: jean-Thierry LAPRESTE, introduction à MATLAB, édition Ellipses,1999.

[17]: Lakhdar Djeffal, A New Approach for Solving an Optimization Problem , Applied Mathematical Sciences, Vol. 6, 2012, no.