REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
وزارة التعليم العالي والبحث العلمي
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Djilali Bounaama, Khemis Miliana
Faculté des Sciences de la Matière et d'informatique
Conseil Scientifique de la Faculté

جامعة جيلالي بونعامة خميس مليانة
كليّة علوم المادة والإعلام الآلي
المجلس العلمي للكلية

Ref : 01 /09/CSF/ 2026

EXTRAIT DU PV
DE LA REUNION ORDINAIRE DU CONSEIL SCIENTIFIQUE
Du 02/02/2026

Objet : : Expertise de polycopié pédagogique

En l'an deux mille vingt-six (2026), le lundi 02 février à 13 h 30, une réunion ordinaire du Conseil Scientifique de la Faculté des Sciences de la Matière et de l'Informatique s'est tenue dans la salle de réunion de la faculté (Bloc B).

Suite aux rapports favorables reçus de la part des experts cités ci-après concernant l'expertise du polycopié pédagogique, le CSF a prononcé favorablement pour la conformité du polycopié pédagogique en vue de préparer son habilitation.

- **Auteur du polycopié** : Dr. MEGHATRIA Farida (MCB)
- **Intitulé du polycopié** : Optimisation continue.
- **Destiné aux étudiants de** : 1ére année Master Analyse Mathématique et Applications.
- **Experts du polycopié** :

• HOUAS Mohamed	Pr	UDB-Khemis Miliana
• MEZOUAGHI Abdelheq	MCA	UHB-Chlef.

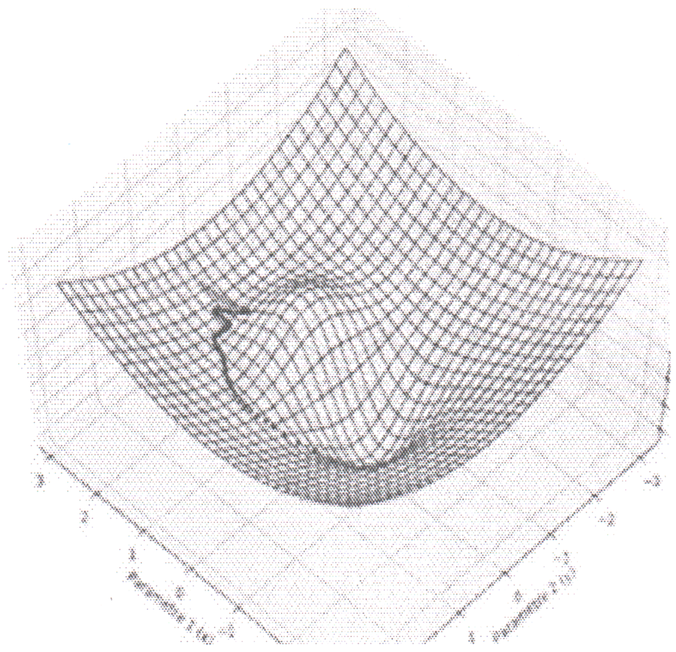
Président du
Conseil Scientifique de la Faculté SMI
Dr. BOUDERBALA Mihoub

رئيس المجلس العلمي
الدكتور: بودربالة ميهوب

المجلس العلمي
كلية علوم المادة و الإعلام الآلي

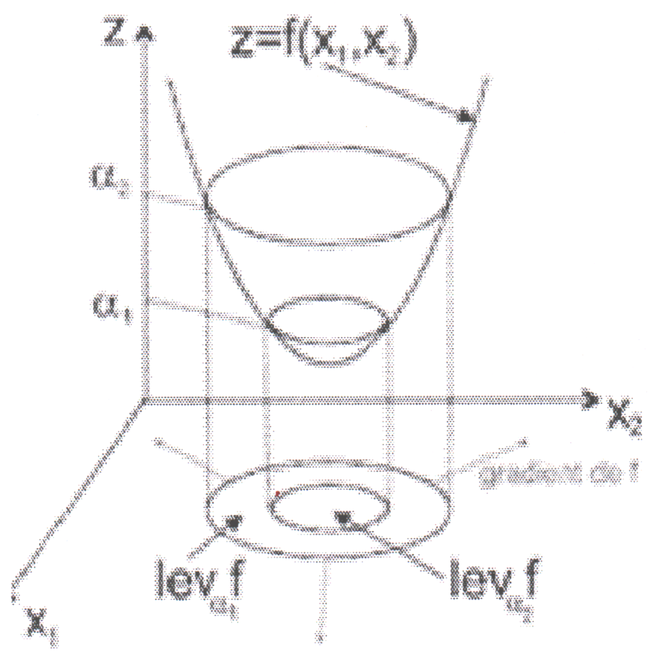
République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Faculté des Sciences de la Matière et Informatique
Département de Mathématiques



OPTIMISATION CONTINUE

Cours, exercices et applications numériques



Destiner aux étudiants de Master 1,
Analyse Mathématique et Applications

Rédiger par,:
Dr Farida Meghatria
f.meghatria@univ-dbkm.dz

Année universitaire: 2025-2026

Table des matières

جامعة الجيلالي بونعامة خميس مليانة
المجلس العلمي
كلية علوم المادة و الإعلام الآلي

1 Introduction aux problèmes d'optimisation dans $\mathbb{R}^n$ 12

1.1 Introduction 12

1.2 Cadre général d'un problème d'optimisation 12

1.2.1 Formulation générale 13

1.2.2 Types de contraintes 13

1.2.3 Objectifs de cette section 13

1.3 Classification des problèmes d'optimisation 13

1.3.1 Optimisation sans contraintes 14

1.3.2 Optimisation sous contraintes 14

1.3.3 Optimisation convexe 14

1.3.4 Optimisation non convexe 14

1.3.5 Autres classifications selon la nature de $f(x)$ 15

1.4 Définitions fondamentales 15

1.4.1 Point admissible 15

1.4.2 Point optimal (ou solution optimale) 15

1.4.3 Minimum local 16

1.4.4 Ensemble des solutions 16

1.4.5 Problème réalisable 16

1.4.6 Problème bien posé 16

1.5 Exemples concrets 28

1.5.1 Exemple 1 : Optimisation sans contraintes 28

1.5.2 Exemple 2 : Optimisation avec contraintes linéaires 29

2 Rappels de calcul différentiel 30

2.1 Fonctions différentiables : rappels essentiels 30

2.1.1 Fonctions de $\mathbb{R}^n$ dans $\mathbb{R}$ 30

2.1.2 Dérivée directionnelle 30

2.1.3 Gradient 30

2.1.4 Différentielle 31

2.1.5 Interprétation géométrique 31
2.1.6 Conditions de différentiabilité 31
2.2 Différentielle de Fréchet 33
2.2.1 Définition 33
2.2.2 Remarques 33
2.2.3 Lien avec la dérivabilité classique 33
2.3 La matrice hessienne 34
2.3.1 Définition 34
2.3.2 Symétrie de la hessienne 35
2.3.3 Interprétation quadratique 35
2.3.4 Rôle en optimisation : test de la hessienne 35
2.4 Théorème des fonctions implicites 36
2.4.1 Motivation 36
2.4.2 Conséquences en optimisation 36
2.4.3 Formule pour la dérivée de la fonction implicite 37

3 Convexité **38**
3.1 Convexité et optimisation 38
3.1.1 Ensembles convexes 38
3.1.2 Fonctions convexes 39
3.1.3 Caractérisation par le gradient (fonction différentiable) 39
3.1.4 Caractérisation par la hessienne (fonction deux fois dérivable) . . . 40
3.1.5 Conséquences en optimisation 40
3.1.6 Combinaisons convexes et opérations 40
3.1.7 Exemples en optimisation 40
3.2 Conditions d'optimalité pour les problèmes sans contrainte 41
3.2.1 Problème général 41
3.2.2 Existence du minimum global 41
3.2.3 Unicité du minimum 41
3.2.4 Conditions nécessaires du premier ordre 42
3.2.5 Conditions du second ordre 42
3.2.6 Résumé 42
3.3 Conditions d'optimalité pour les problèmes avec contraintes 43
3.3.1 Problème d'optimisation sous contraintes 43
3.3.2 Multiplicateurs de Lagrange (cas sans inégalités) 43
3.3.3 Conditions de Karush-Kuhn-Tucker (KKT) 50
3.3.4 Interprétation géométrique 52
3.3.5 Conditions du second ordre (avec égalité) 53

جامعة الجيلالي بونعامة خميس مليانة
كلية علوم المادة والإعلام الآلي
المجلس العلمي

3.3.6 Conditions du second ordre pour un problème avec inégalités 56

4 Algorithmes de minimisation sans contrainte **64**

4.1 Algorithme de descente général 64

4.1.1 Méthode du gradient à pas fixe 66

4.1.2 Méthode du gradient à pas optimal 67

4.1.3 Méthode du Gradient Conjugué 69

4.1.4 Méthode de Newton 71

4.1.5 Méthodes quasi-Newton (ex : BFGS) 77

4.1.6 Gradient conjugué (fonctions quadratiques) 79

4.1.7 Méthodes de relaxation 81

4.1.8 Algorithmes probabilistes 81

4.2 Comparaison synthétique 82

5 Méthodes avec contraintes **83**

5.1 Multiplicateurs de Lagrange 83

5.2 Théorème de Karush-Kuhn-Tucker (KKT) 84

5.3 Méthodes primales-duales 89

5.3.1 Méthode d'Uzawa (pour contraintes d'égalité) 89

5.3.2 Lagrangien Augmenté 93

5.3.3 SQP (Sequential Quadratic Programming) 99

5.3.4 Méthode de Newton-KKT 99

5.4 Comparaison synthétique 100

5.5 Exemples et cas d'application 100

5.5.1 Programmation quadratique 102

5.5.2 Optimisation en apprentissage automatique 109

5.5.3 Exemple de problème inverse 112

6 Implémentation de divers algorithmes et applications **114**

6.1 Implémentation des algorithmes sans contraintes 114

6.1.1 Méthode du Gradient 114

6.1.2 Méthode du Gradient à pas fixe 115

6.1.3 Méthode du Gradient à pas optimale 118

6.1.4 Méthode du Gradient Conjugué 119

6.1.5 Méthode de Newton 121

6.1.6 Méthode de Relaxation 123

6.2 Implémentation des algorithmes avec contraintes 125

6.2.1 Méthode des multiplicateurs de Lagrange 125

جامعة الجيلالي بونعامة خميس مليانة
كلية علوم المادة و الإعلام الآلي
المجلس العلمي

6.2.2 Méthode d'Uzawa 127

7 Annexe **131**

7.1 Examens 132

Table des figures

3.1 Le gradient $\nabla f(x^*)$ est orthogonal à la courbe $\{x(t)\}$ au point x^* qui minimise f sur la courbe. 45
3.2 Quatre exemples où la condition de Lagrange est satisfaite : (a) maximum, (b) minimum, (c) minimum, (d) point non extremum. 46
3.3 Exemple dans le cas où la condition de Lagrange n'est pas vérifiée. 47
3.4 Solution graphique du problème de l'exemple 3.8 49
3.5 Illustration du théorème de Karush-Kuhn-Tucker (KKT). 51
3.6 Point optimal x^* au bord de la contrainte active $x + y \leq 1$ 63

5.1 Courbes de niveau de $f(x, y) = x^2 + y^2 + 3x + 4y$ avec domaine admissible 109

جامعة الجيلالي بونعامة خميس مليانة
المكتبة المركزية للعلوم والتكنولوجيا

Liste des tableaux

جامعة الجيلالي بونعامة خميس مليانة
المجلس العلمي
كلية العلوم الاقتصادية والتجارية وعلوم التسيير

4.1 Tableau récapitulatif des méthodes sans contraintes 82

5.1 Tableau récapitulatif des méthodes avec contraintes 100

Avant-Propos

Ce polycopié est issu du cours d'optimisation continue présenté à l'université de Djillali Bounaama-Khemis Miliana (Algérie) (premier semestre) ces sept dernières années.

Il s'agit des cours destinées aux étudiants de première année Master spécialité Analyse Mathématique et Applications (AMA).

Dans notre présentation, nous supposons que l'étudiant possède des connaissances de base en algèbre linéaire et en analyse à plusieurs variables. Pour faciliter la lecture et garantir une meilleure compréhension.

Ce cours d'optimisation continue a pour objectif de fournir aux étudiants les bases théoriques et pratiques nécessaires à la compréhension et à la mise en œuvre des principales méthodes d'optimisation dans un cadre différentiable. Il couvre des sujets clés tels que les conditions d'optimalité, les méthodes de descente, l'optimisation avec ou sans contraintes, ainsi que les algorithmes classiques comme la descente de gradient ou les méthodes de Newton.

L'approche adoptée dans ce cours cherche à allier rigueur mathématique et intuition géométrique, tout en proposant des applications concrètes et des exercices pratiques pour favoriser l'apprentissage actif.

Le format de ce cours est composé d'une séance de $1h$: 30 de cours associées à une séance de $1h$: 30 de travaux dirigés (une séance de travaux pratiques été toujours associé avant 2023). Chaque couple de séances (cours-TD) porte sur un thème différent associé à un chapitre de ces notes. L'utilisation de logiciel de calcul est devenu depuis les années 70 absolument indispensable dans le domaine scientifique, aussi bien pour le technicien que l'ingénieur ou le chercheur. On adopte l'outil de logiciel Matlab, qui été développé par la société Mathworks. Il s'agit d'un des langages de programmation scientifique les plus populaires.

Introduction générale

L'optimisation constitue une discipline fondamentale dans l'analyse et la résolution de problèmes impliquant la prise de décision, qu'il s'agisse de domaines tels que l'ingénierie, l'économie, les sciences des données ou encore la logistique. Dans tout processus décisionnel, il s'agit de choisir, parmi un ensemble d'alternatives possibles, celle qui conduit à la meilleure performance selon un critère donné.

Ce critère, souvent appelé fonction objectif ou indice de performance, permet d'évaluer la qualité des différentes options. L'objectif de l'optimisation est donc de déterminer la solution optimale – c'est-à-dire celle qui maximise (ou minimise) cette fonction – tout en respectant éventuellement un ensemble de contraintes liées au problème étudié.

La théorie de l'optimisation et les méthodes qui en découlent offrent un cadre mathématique rigoureux pour modéliser ces situations complexes et guider le processus de décision vers des choix efficaces, rationnels et mesurables. Ces outils sont aujourd'hui essentiels dans la recherche opérationnelle, l'apprentissage automatique, la conception de systèmes techniques ou encore la gestion des ressources.

Le domaine de l'optimisation a suscité un intérêt considérable au cours des dernières années, principalement en raison des progrès rapides de la technologie informatique. Cela inclut le développement et la disponibilité de logiciels conviviaux, l'émergence de processeurs à haute vitesse et de systèmes de calcul parallèles, ainsi que l'essor des réseaux de neurones artificiels. Un exemple clair de cette évolution réside dans l'accessibilité croissante des outils logiciels d'optimisation, tels que la boîte à outils d'optimisation (Optimization Toolbox) de MATLAB, ainsi que de nombreux autres logiciels commerciaux spécialisés.

Il existe actuellement plusieurs excellents ouvrages de niveau master consacrés à la théorie et aux méthodes d'optimisation (par exemple, [5], [6], [7], [8], [9], [10], [11], [12]), ainsi que des ouvrages de niveau licence, orientés notamment vers la conception en ingénierie (par exemple, [13] et [14]). Cependant, il demeure un besoin réel d'un manuel d'introduction à la théorie et aux méthodes de l'optimisation, destiné aux étudiants de première année Master Analyse mathématique et applications.

L'objectif de cet polycopie est de fournir aux étudiants une compréhension opéra-

tionnelle de la théorie de l'optimisation et de ses méthodes. Pour atteindre ce but, nous avons intégré de nombreux exemples illustrant les concepts théoriques et les algorithmes présentés dans le texte.

Cependant, il ne s'agit pas ici de proposer un simple recueil de recettes contenant les techniques numériques les plus récentes en optimisation. Notre ambition est plutôt de doter l'étudiant d'un socle de connaissances solide, lui permettant d'aborder des sujets plus avancés dans le domaine de l'optimisation.

Le domaine de l'optimisation demeure un champ de recherche extrêmement actif. Au cours des dernières années, de nombreuses nouvelles approches ont été proposées.

On observe également une forte augmentation des applications des méthodes d'optimisation à de nouveaux types de problèmes. Un exemple notable est l'utilisation des algorithmes de descente pour l'entraînement des réseaux de neurones à propagation avant (feedforward neural networks).

Ce cours est divisé en 6 chapitres.

- Dans le **chapitre** 1, on donnons des définitions et une bonne description des problèmes d'optimisation en $\mathbb{R}^n$, la formulation générale du problème d'optimisation, la classification des problèmes, les définitions fondamentales des points (admissible, optimale, locale, globale) et les ensembles de ces points, puis le problème bien posé et quelques exemples concrets.
- Dans le **chapitre** 2, on propose quelque rappels fondamentales de calcul différentielle.
- Dans le **chapitre** 3, on a la convexité, commençons par les définitions de base de la convexité des fonctions et des ensembles, a caractérisation de ces derniers,puis les conditions d'optimalité de 1^{er} et 2^{me} ordre et on termine par une bonne gamme d'exercices avec solutions.
- Dans le **chapitre** 4, on a les méthodes sans contraintes, où on présente le principe de la méthode, l'algorithme, implémentation en Matlab et quelques exemples et exercices.
- Dans le **chapitre** 5, on a les méthodes avec contraintes, où on a d'abord les théorème de base de l'optimisation avec contraintes, puis on présente le principe de chaque méthode, l'algorithme, quelques exemples et ces cas d'application majeurs, tel que la programmation quadratique, l'apprentissage automatique et exemple de problème inverse.
- Dans le **chapitre** 6, on a les implémentation en Matlab et quelques exemples et exercices..

On clôture ce polycopié avec quelques sujets d'examens des années précédentes, afin d'aider l'étudiant à mieux se prépare aux épreuves de la fin du semestre, et des références biographiques. Toute suggestion, avis ou remarque de la part des lecteurs (étudiants ou

enseignants) qui concerne le font ou la forme de ce polycopié est la bienvenue ; cela nous aidera à l'améliorer.

Chapitre 1

Introduction aux problèmes d'optimisation dans $\mathbb{R}^n$

1.1 Introduction

L'optimisation dans $\mathbb{R}^n$ se concentre sur les problèmes où les *variables de décision* sont des vecteurs réels de dimension n. Ces problèmes peuvent être très simples (comme trouver le minimum d'une fonction quadratique sans contraintes), ou très complexes (comme l'optimisation de réseaux neuronaux ou la planification de systèmes industriels).

Ce chapitre a pour objectif de poser les bases formelles nécessaires à l'étude des problèmes d'optimisation dans $\mathbb{R}^n$. Nous introduirons les notations, la formulation générale des problèmes, les types de contraintes, ainsi que les différentes catégories de problèmes que l'on peut rencontrer. Ces fondations permettront d'aborder, dans les chapitres suivants, les *conditions d'optimalité*, les *méthodes de résolution* analytiques ou numériques, ainsi que certains cas particuliers comme les problèmes convexes ou linéaires.

1.2 Cadre général d'un problème d'optimisation

Un *problème d'optimisation* dans $\mathbb{R}^n$ consiste à déterminer les valeurs d'un vecteur de variables $x \in \mathbb{R}^n$ qui minimisent (ou maximisent) une fonction objectif $f : \mathbb{R}^n \to \mathbb{R}$, tout en satisfaisant éventuellement certaines contraintes.

1.2.1 Formulation générale

La forme générale d'un problème d'optimisation peut s'écrire comme suit :

$$\begin{aligned} &\min_{x\in\mathbb{R}^n} \quad f(x) \\ &\text{sous les contraintes :} \quad x \in S, \end{aligned}$$

où :

- $\surd$ $x = (x_1, x_2, \ldots, x_n)^T$ est le vecteur des variables de décision ;
- $\surd$ $f(x)$ est la fonction objectif à minimiser ;
- $\surd$ $S \subseteq \mathbb{R}^n$ est l'ensemble des points admissibles (aussi appelé ensemble de *faisabilité*), défini par un ensemble de contraintes.

1.2.2 Types de contraintes

L'ensemble S peut être défini à l'aide de deux types de contraintes :

— **Contraintes d'égalité** : $h_i(x) = 0$, pour $i = 1, \ldots, m$;

— **Contraintes d'inégalité** : $g_j(x) \leq 0$, pour $j = 1, \ldots, p$.

Ainsi, la formulation complète devient :

$$\begin{aligned} &\min_{x\in\mathbb{R}^n} \quad f(x) \\ &\text{sous les contraintes :} \\ &\quad h_i(x) = 0, \quad i = 1, \ldots, m, \\ &\quad g_j(x) \leq 0, \quad j = 1, \ldots, p. \end{aligned}$$

1.2.3 Objectifs de cette section

Cette formulation générale permet de couvrir une large variété de problèmes rencontrés en pratique, allant de l'optimisation sans contraintes à l'optimisation sous contraintes non linéaires. Dans les sections suivantes, nous allons explorer différentes classes de problèmes d'optimisation selon la nature de f, g_j et h_i, et établir les définitions fondamentales nécessaires à leur analyse.

1.3 Classification des problèmes d'optimisation

Les problèmes d'optimisation peuvent être classés selon plusieurs critères, en fonction de la présence ou non de contraintes, de la nature des fonctions intervenant dans la formulation (objectif et contraintes), ou encore des propriétés géométriques de ces fonctions

et de l'ensemble admissible. Cette classification permet de mieux comprendre la difficulté du problème et les méthodes de résolution appropriées.

1.3.1 Optimisation sans contraintes

Il s'agit de la forme la plus simple d'un problème d'optimisation, où l'on cherche à minimiser une fonction f définie sur tout $\mathbb{R}^n$ sans aucune contrainte :

$$\min_{x \in \mathbb{R}^n} f(x).$$

Dans ce cas, tout point $x \in \mathbb{R}^n$ est admissible. Ce type de problème est souvent utilisé comme point de départ pour comprendre les concepts de base (gradient, hessienne, etc.).

1.3.2 Optimisation sous contraintes

Lorsque certaines restrictions s'appliquent aux variables de décision, le problème prend la forme :

$$\begin{aligned} &\min_{x \in \mathbb{R}^n} \quad f(x) \\ &\text{sous les contraintes :} \\ &\quad h_i(x) = 0, \quad i = 1, \ldots, m, \\ &\quad g_j(x) \leq 0, \quad j = 1, \ldots, p. \end{aligned}$$

La présence de contraintes peut considérablement compliquer la résolution du problème, en particulier si ces contraintes sont non linéaires.

1.3.3 Optimisation convexe

Un problème est dit *convexe* si :

— la fonction objectif f est convexe,

— l'ensemble admissible S est un ensemble convexe (défini par des fonctions convexes pour les inégalités, affines pour les égalités).

Dans ce cas, tout minimum local est également un minimum global, ce qui simplifie grandement l'analyse et les méthodes de résolution.

1.3.4 Optimisation non convexe

Si au moins l'une des conditions de convexité n'est pas remplie, le problème est dit *non convexe*. Ces problèmes sont en général plus difficiles à résoudre, car ils peuvent ad-

mettre plusieurs optima locaux. Des techniques spécifiques sont alors requises (méthodes heuristiques, globales, etc.).

1.3.5 Autres classifications selon la nature de $f(x)$

On distingue aussi les problèmes selon la forme de la fonction objectif :

— **Problème linéaire** : $f(x)$ est une fonction linéaire, et les contraintes sont linéaires.

— **Problème quadratique** : $f(x)$ est une fonction quadratique, typiquement de la forme $f(x) = \frac{1}{2}x^T Qx + c^T x + d$.

— **Problème non linéaire** : $f(x)$ ou les contraintes sont non linéaires.

Chaque type de problème nécessite des méthodes spécifiques : le simplexe ou l'analyse duale pour les problèmes linéaires, les méthodes de Karush-Kuhn-Tucker (KKT) pour les problèmes non linéaires, ou encore des algorithmes numériques sophistiqués pour les problèmes de grande dimension.

1.4 Définitions fondamentales

Avant d'aborder les méthodes de résolution des problèmes d'optimisation, il est essentiel de définir quelques notions fondamentales qui permettent de mieux comprendre la structure des solutions possibles.

1.4.1 Point admissible

Un point $x \in \mathbb{R}^n$ est dit **admissible** (ou *faisable*) s'il satisfait toutes les contraintes du problème. Autrement dit, x appartient à l'ensemble de faisabilité S :

$$x \in S \quad \Leftrightarrow \quad \begin{cases} g_j(x) \leq 0, & \text{pour } j = 1, \ldots, p, \\ h_i(x) = 0, & \text{pour } i = 1, \ldots, m. \end{cases}$$

1.4.2 Point optimal (ou solution optimale)

Un point admissible $x^* \in S$ est une **solution optimale** (ou un **point optimal**) du problème s'il minimise la fonction objectif :

$$f(x^*) \leq f(x), \quad \forall x \in S.$$

Dans ce cas, on dit que x^* est un *minimum global*.

1.4.3 Minimum local

Un point admissible $x^* \in S$ est un **minimum local** s'il existe un voisinage $V(x^*)$ tel que :

$$f(x^*) \leq f(x), \quad \forall x \in S \cap V(x^*).$$

Un minimum local n'est pas nécessairement un minimum global, sauf dans le cas d'un problème convexe, où les deux coïncident.

1.4.4 Ensemble des solutions

L'ensemble des points optimaux est noté :

$$\mathcal{X}^* = \{x \in S \mid f(x) = f^*\},$$

où f^* est la **valeur optimale** du problème, définie comme :

$$f^* = \inf_{x \in S} f(x).$$

On parle de *minimum atteint* si f^* est obtenu pour au moins un point $x \in S$.

1.4.5 Problème réalisable

Un problème est dit **réalisable** (ou *faisable*) s'il existe au moins un point admissible, c'est-à-dire si $S \neq \emptyset$.

1.4.6 Problème bien posé

Un problème est dit **bien posé** s'il est réalisable, que la fonction objectif est bornée inférieurement sur S, et que la valeur optimale est atteinte. Sinon, le problème est dit *mal posé*.

Quelques problèmes d'optimisation dans $\mathbb{R}$

structure générale d'un problème d'optimisation

$$\begin{array}{ll} \min\limits_{x \in D} & f(x) \\ \text{sous contrainte(s)} & x \in [a, b] \text{ (si besoin)} \end{array}$$

1. Minimisation d'une fonction quadratique

   Problème concret : minimisation du coût de production

Une entreprise fabrique un produit. Le coût de fabrication (en euros) en fonction du nombre d'unités produites x est donné par :

$$C(x) = x^2 - 10x + 400$$

où x représente le nombre d'unités produites par jour.

On cherche à :

$$\min_{x \in \mathbb{R}} C(x)$$

(a) Étape 1 : dérivée première

$$C'(x) = \frac{d}{dx}(x^2 - 10x + 400) = 2x - 10$$

(b) Étape 2 : condition du premier ordre

$$C'(x) = 0 \quad \Rightarrow \quad 2x - 10 = 0 \quad \Rightarrow \quad x = 5$$

(c) Étape 3 : vérification du minimum

$$C''(x) = \frac{d}{dx}(2x - 10) = 2 > 0$$

Donc, $x = 5$ est un minimum global.

(d) Étape 4 : coût minimal

$$C(5) = 5^2 - 10 \cdot 5 + 400 = 25 - 50 + 400 = 375$$

Conclusion

Pour minimiser le coût de production, l'entreprise doit produire **5 unités** par jour.

Le coût minimal est de **375 euros**.

Exercice 1.1. *Trouver le minimum de la fonction :*

$$\min_{x \in \mathbb{R}} f(x) = x^2 + 4x + 5$$

2. Maximisation d'une fonction de revenu (économie)

Problème concret : maximisation du revenu

Une entreprise vend un produit à un prix unitaire dépendant de la quantité vendue :

$$p(x) = 100 - 2x$$

Le revenu total est donné par :

$$R(x) = x \cdot p(x) = x(100 - 2x) = -2x^2 + 100x$$

Le Problème à résoudre est :

$$\max_{x \in [0,50]} R(x) = -2x^2 + 100x$$

Étape 1 : dérivée première

$$R'(x) = \frac{d}{dx}(-2x^2 + 100x) = -4x + 100$$

Étape 2 : condition du premier ordre

$$R'(x) = 0 \quad \Rightarrow \quad -4x + 100 = 0 \quad \Rightarrow \quad x = 25$$

Étape 3 : vérification du maximum

$$R''(x) = \frac{d}{dx}(-4x + 100) = -4 < 0$$

Donc, $x = 25$ est un **maximum global** (car la fonction est concave).

Étape 4 : revenu maximal

$$R(25) = -2 \cdot 25^2 + 100 \cdot 25 = -1250 + 2500 = 1250$$

Conclusion

Pour maximiser son revenu, l'entreprise doit vendre **25 unités** par jour.
Le revenu maximal est de **1250 euros**.

Résultat : Quantité optimale à vendre : 25
Revenu maximal :1250 euro.

Exercice 1.2.

$$\max_{x \geq 0} R(x) = x(100 - 2x)$$

3. Minimisation d'un coût de production

Problème concret : minimiser un coût de production

Une entreprise fabrique des pièces mécaniques. Le coût de production total en fonction du nombre de pièces produites par jour x (en centaines d'unités) est donné par :

$$C(x) = 0{,}5x^2 - 6x + 150$$

où $C(x)$ est exprimé en centaines d'euros.

(a) Quel est le nombre optimal de pièces à produire pour minimiser le coût de production journalier ?

(b) Quel est le coût minimal ?

Nous cherchons à minimiser ce coût :

$$\min_{x \in \mathbb{R}_+} C(x)$$

(a) *Étape 1 : dérivée première*

$$C'(x) = \frac{d}{dx}(0{,}5x^2 - 6x + 150) = x - 6$$

(b) Étape 2 : condition du premier ordre

$$C'(x) = 0 \quad \Rightarrow \quad x - 6 = 0 \quad \Rightarrow \quad x = 6$$

(c) Étape 3 : vérification du minimum

$$C''(x) = \frac{d}{dx}(x - 6) = 1 > 0$$

Donc, $x = 6$ est un minimum global.

(d) Étape 4 : coût minimal

$$C(6) = 0{,}5 \cdot 6^2 - 6 \cdot 6 + 150 = 18 - 36 + 150 = 132$$

Le coût minimal est donc de 132 centaines d'euros, soit 13200 euro.

Conclusion

L'entreprise doit produire **600 pièces** par jour pour minimiser ses coûts.
Le coût minimal est de **13200 euro**.

Résultat final :

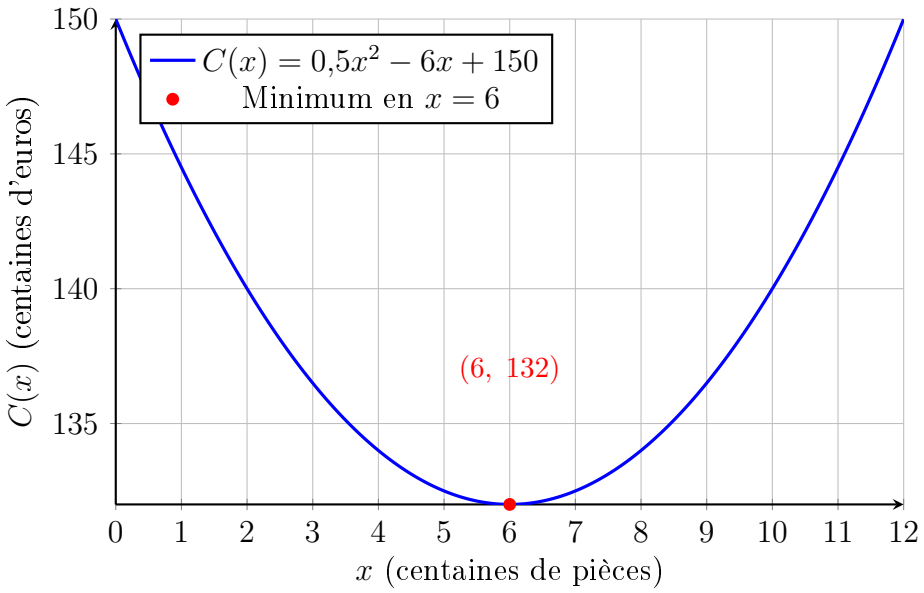
Le problème peut être améliorer par :

(a) Un graphe de $C(x)$ avec minimum visible ?

(b) Une contrainte de production (ex : capacité max ou min) ?

(c) Le coût marginal ou d'autres aspects économiques ?

Graphe du coût de production



Exercice 1.3.

$$\min_{x>0} C(x) = 0{,}1x^2 - 5x + 200$$

4. Maximisation d'une aire (géométrie)

Problème concret : Maximiser l'aire d'un rectangle avec périmètre fixé

On veut construire un enclos rectangulaire contre un mur (donc un seul des quatre côtés n'a pas besoin de clôture).

On dispose de 40 mètres de clôture pour entourer les trois autres côtés (deux largeurs et une longueur).

(a) Quelle est la dimension du rectangle qui maximise l'aire de l'enclos ?

(b) Quelle est la valeur maximale de l'aire ?

On souhaite construire un enclos rectangulaire contre un mur, avec 40 mètres de clôture pour les trois autres côtés (deux largeurs et une longueur). Soit :

$\surd$ x : la largeur (perpendiculaire au mur),

$\surd$ L : la longueur (le côté parallèle au mur, qui est clôturé).

(a) 1. Relation géométrique

Le périmètre utilisé est :

$$2x + L = 40 \quad \Rightarrow \quad L = 40 - 2x$$

(b) 2. Aire à maximiser

L'aire du rectangle est :

$$A(x) = x \cdot L = x(40 - 2x) = -2x^2 + 40x$$

On veut maximiser :

$$\max_{x \in [0,20]} A(x) = -2x^2 + 40x$$

(c) 3. Dérivée première

$$A'(x) = \frac{d}{dx}(-2x^2 + 40x) = -4x + 40$$

(d) 4. Condition du premier ordre

$$A'(x) = 0 \Rightarrow -4x + 40 = 0 \Rightarrow x = 10$$

(e) 5. Vérification

$$A''(x) = -4 < 0 \quad \Rightarrow \text{maximum atteint en } x = 10$$

(f) 6. Aire maximale

$$A(10) = -2 \cdot 10^2 + 40 \cdot 10 = -200 + 400 = 200$$

Conclusion

L'aire maximale est atteinte pour une largeur de **10 m** et une longueur de **20 m**.
L'aire maximale est de $200m^2$.

Le problème peut être améliorer par :

(a) Un graphe de l'aire $A(x) = -2x^2 + 40x$?

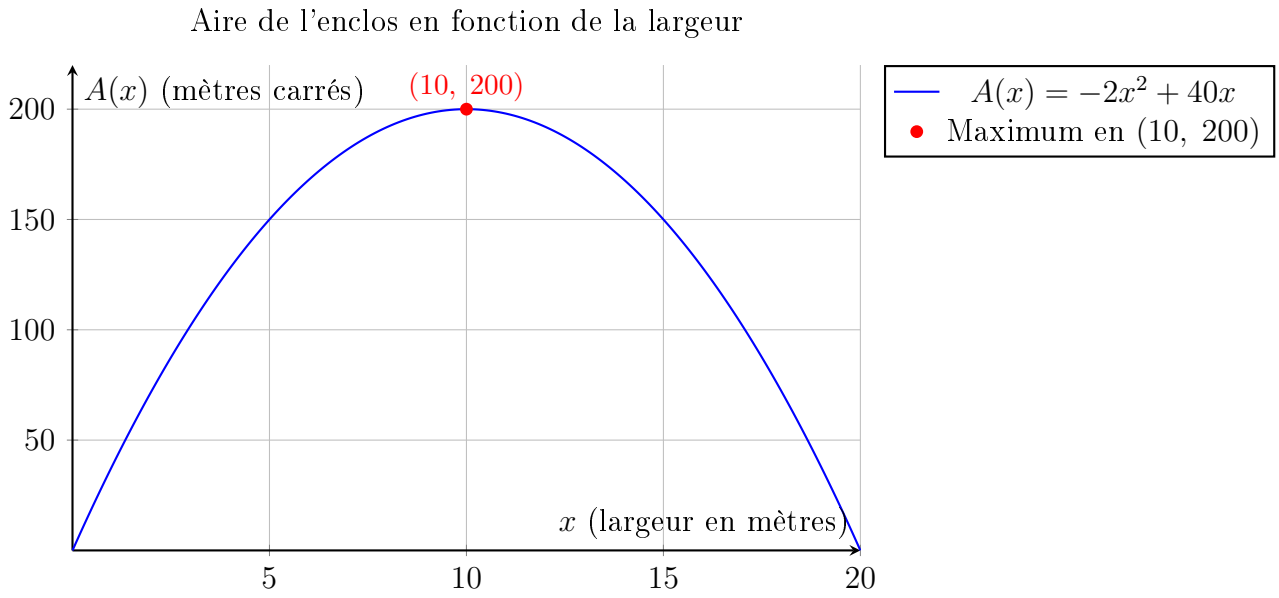
(b) Une version avec une forme différente (triangle, cercle . . .) ?

(c) Une contrainte supplémentaire (ex : terrain inaccessible) ?

Graphe de l'aire de l'enclos

La fonction de l'aire est :

$$A(x) = -2x^2 + 40x$$



Exercice 1.4. *On veut maximiser l'aire d'un rectangle de périmètre fixé.*

$$\max_{x\in[0,50]} A(x) = x(50 - x)$$

5. Minimisation d'une fonction exponentielle

Problème concret : refroidissement d'un objet (loi exponentielle)

Un objet est placé dans un environnement à température ambiante. Sa température (en C^{circ}) au bout de t minutes est modélisée par la fonction suivante :

$$T(t) = 5t \cdot e^{-0.1t}$$

On cherche le moment t où la température de l'objet est minimale.

où t est le temps (en minutes), et $T(t)$ la température de l'objet (en C^{circ}). On souhaite déterminer à quel moment la température est minimale.

(a) Étape 1 : dérivée première

On utilise la règle du produit :

$$T'(t) = \frac{d}{dt}(5t \cdot e^{-0.1t}) = 5 \cdot e^{-0.1t} + 5t \cdot (-0.1) \cdot e^{-0.1t}$$

$$T'(t) = 5e^{-0.1t} - 0.5te^{-0.1t}$$

$$T'(t) = e^{-0.1t}(5 - 0.5t)$$

(b) Étape 2 : condition du premier ordre

$$T'(t) = 0 \quad \Leftrightarrow \quad e^{-0.1t}(5 - 0.5t) = 0$$

Or, $e^{-0.1t} \neq 0$, donc :

$$5 - 0.5t = 0 \quad \Rightarrow \quad t = 10$$

(c) Étape 3 : test du minimum

On examine le signe de $T''(t)$ ou la variation de $T'(t)$. Puisque $T'(t)$ change de signe de positif à négatif à $t = 10$, il s'agit d'un **maximum**.

> Mais on cherchait un **minimum** ! Donc la **température ne diminue pas indéfiniment**, elle **atteint un maximum à $t = 10$**.

—

(d) Étape 4 : interprétation correcte

Donc si on veut minimiser $T(t) = 5te^{-0.1t}$, on cherche la **valeur minimale sur un intervalle donné**. Supposons que $t \in [0, 60]$, on compare alors les valeurs aux bornes.

$$T(0) = 0, \quad T(60) = 5 \cdot 60 \cdot e^{-6} \approx 300 \cdot 0.00248 \approx 0.744$$

Résultat

- La fonction $T(t) = 5te^{-0.1t}$ atteint un **maximum** en $t = 10$, et tend vers zéro ensuite. - La **valeur minimale** (sur $t \in [0, 60]$) est proche de **0** quand $t \to \infty$.

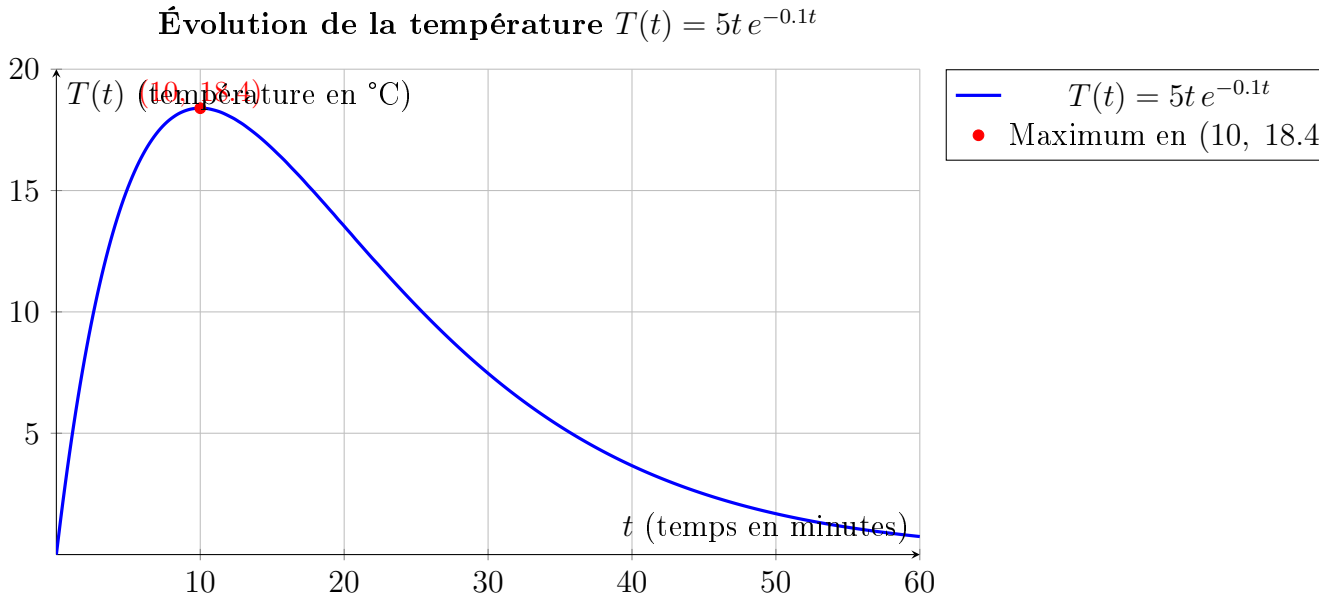
Le problème peut être améliorer par :

(a) Un graphe de cette fonction avec maximum visible ?

(b) Une variante du problème (autre coefficient, contrainte, domaine) ?

(c) Une interprétation physique plus poussée (refroidissement, réaction chimique . . .) ?

Graphe de la fonction $T(t) = 5t\, e^{-0.1t}$



Exercice 1.5.

$$\min_{x>0} f(x) = xe^{-x}$$

6. Optimisation avec contrainte (par exemple, bornes)

Problème concret : optimiser un coût de transport avec contrainte de distance

Une entreprise doit choisir l'emplacement d'un entrepôt sur une ligne droite reliant deux villes, A et B, distantes de 100 km.

Le coût de transport par jour en fonction de la position x (en km à partir de la ville A) est donné par :

$$C(x) = (x - 20)^2 + (x - 80)^2$$

où,

- Le premier terme représente le coût de transport depuis l'entrepôt vers la ville A,
- Le second vers la ville B.

On souhaite minimiser le coût total, sachant que l'entrepôt doit être situé entre les deux villes :

$$x \in [0,\ 100]$$

(a) Étape 1 : développer la fonction

$$C(x) = (x^2 - 40x + 400) + (x^2 - 160x + 6400)$$

$$C(x) = 2x^2 - 200x + 6800$$

(b) Étape 2 : dérivée première

$$C'(x) = 4x - 200$$

(c) Étape 3 : condition du premier ordre

$$C'(x) = 0 \Rightarrow 4x - 200 = 0 \Rightarrow x = 50$$

(d) Étape 4 : vérifier la contrainte

$x = 50 \in [0, 100]$, donc la solution est **admissible**.

(e) Étape 5 : valeur minimale du coût

$$C(50) = (50 - 20)^2 + (50 - 80)^2 = 30^2 + (-30)^2 = 900 + 900 = 1800$$

Conclusion

L'emplacement optimal de l'entrepôt est à **50 km de la ville A**.
Le coût de transport minimal est de **1800 unités**.

Le problème peut être améliorer par :

(a) Tracer le graphe de $C(x)$ avec les bornes visibles et le minimum marqué

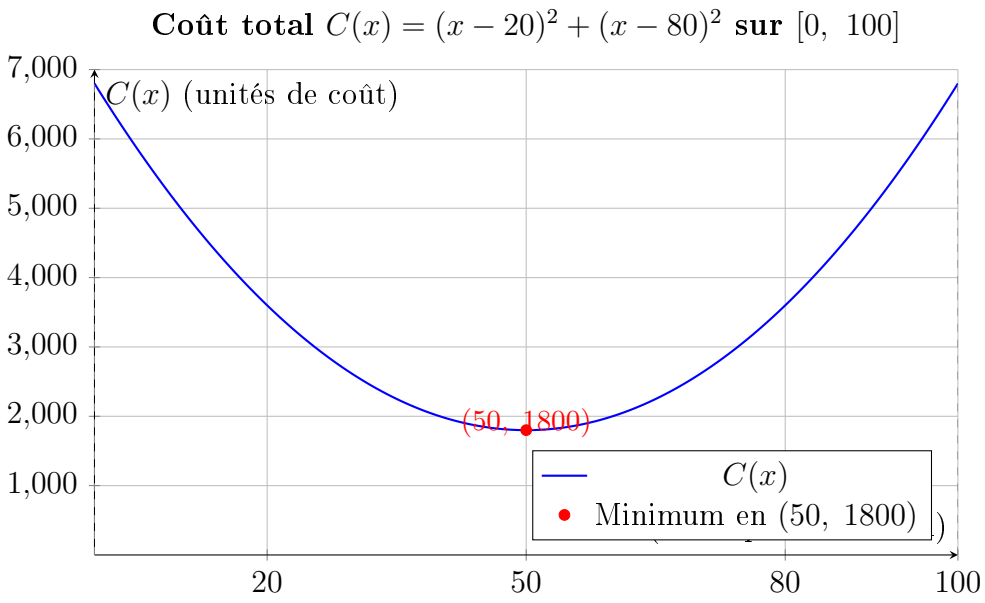
(b) Ajouter des contraintes supplémentaires (ex : zone interdite : x $\notin$

$$30, 40$$

)

(c) Résoudre un problème avec plusieurs contraintes (type programmation linéaire ou quadratique)

Graphe de la fonction $C(x) = (x - 20)^2 + (x - 80)^2$



Exercice 1.6.

$$\min_{x\in[1,10]} f(x) = \ln(x) + \frac{1}{x}$$

7. Problème de type physique : minimisation du temps

Problème de minimisation du temps : Le sauveteur et le nageur

Un sauveteur se trouve sur une plage à une distance d perpendiculaire du bord de la mer. Un nageur en difficulté est situé dans l'eau, à une distance L perpendiculaire au rivage et à une distance horizontale a du point du rivage le plus proche du sauveteur.

Le sauveteur peut :

— courir sur le sable à une vitesse v_1,

— nager dans l'eau à une vitesse plus lente v_2, avec $v_2 < v_1$.

Il doit choisir un point d'entrée dans l'eau situé à une distance x le long du rivage à partir de sa position la plus proche sur la plage. Le but est de minimiser le temps total pour atteindre le nageur.

Temps total de parcours

1. Sur la plage (course)

   Le sauveteur parcourt une distance

$$d_1 = \sqrt{d^2 + x^2}$$

   en courant, ce qui prend un temps

$$T_1(x) = \frac{\sqrt{d^2 + x^2}}{v_1}.$$

2. Dans l'eau (nage)

   Ensuite, il nage une distance

$$d_2 = \sqrt{L^2 + (a-x)^2}$$

   ce qui prend un temps

$$T_2(x) = \frac{\sqrt{L^2 + (a-x)^2}}{v_2}.$$

3. Temps total

   Le temps total est donc donné par la fonction :

$$T(x) = \frac{\sqrt{d^2 + x^2}}{v_1} + \frac{\sqrt{L^2 + (a-x)^2}}{v_2}.$$

Problème à résoudre

Trouver la valeur de $x \in [0, a]$ qui minimise le temps total $T(x)$:

$$\min_{x \in [0,a]} T(x).$$

Remarque

Ce problème est analogue à celui de la réfraction de la lumière (loi de Snell-Descartes) : la trajectoire optimale n'est pas la plus courte, mais celle qui minimise le temps de trajet, en tenant compte des vitesses différentes dans chaque milieu.

Exercice 1.7. *Un objet suit un trajet dont le temps dépend de la position x.*

$$\min_{x \in [0,a]} T(x) = \frac{\sqrt{x^2 + h^2}}{v_1} + \frac{a-x}{v_2}$$

1.5 Exemples concrets

Afin de mieux comprendre la formulation et les concepts précédemment introduits, nous présentons ici deux exemples de problèmes d'optimisation dans $\mathbb{R}^2$.

1.5.1 Exemple 1 : Optimisation sans contraintes

Considérons la fonction quadratique suivante :

$$f(x) = x_1^2 + 2x_2^2 - 4x_1 - 8x_2,$$

où $x = (x_1, x_2)^T \in \mathbb{R}^2$.
Ce problème peut être formulé ainsi :

$$\min_{x \in \mathbb{R}^2} f(x).$$

Solution : La fonction f est différentiable. Son gradient est :

$$\nabla f(x) = \begin{pmatrix} 2x_1 - 4 \\ 4x_2 - 8 \end{pmatrix}.$$

On annule le gradient pour trouver les points stationnaires :

$$\begin{cases} 2x_1 - 4 = 0 \\ 4x_2 - 8 = 0 \end{cases} \quad \Rightarrow \quad x_1 = 2, \quad x_2 = 2.$$

La hessienne est constante :

$$H = \begin{pmatrix} 2 & 0 \\ 0 & 4 \end{pmatrix},$$

qui est définie positive. Ainsi, f est convexe et $x^* = (2, 2)^T$ est un minimum global.
Valeur optimale :

$$f(2, 2) = 2^2 + 2 \cdot 2^2 - 4 \cdot 2 - 8 \cdot 2 = 4 + 8 - 8 - 16 = -12.$$

1.5.2 Exemple 2 : Optimisation avec contraintes linéaires

Considérons le problème suivant :

$$\min_{x\in\mathbb{R}^2} \quad f(x) = x_1 + x_2$$

$$\text{sous les contraintes : } \begin{cases} x_1 \geq 0 \\ x_2 \geq 0 \\ x_1 + 2x_2 \leq 4 \end{cases}$$

Interprétation : Le domaine admissible est un polygone convexe dans $\mathbb{R}^2$, délimité par les axes et la droite $x_1 + 2x_2 = 4$.

Méthode graphique (résumé) : Puisque $f(x) = x_1 + x_2$ est linéaire, le minimum est atteint à l'un des sommets du polygone de faisabilité.

Les sommets sont :

$$A = (0, 0), \quad B = (4, 0), \quad C = (0, 2).$$

On évalue f sur ces sommets :

$$f(A) = 0, \quad f(B) = 4, \quad f(C) = 2.$$

Conclusion : le minimum est atteint en $A = (0, 0)$, avec $f^* = 0$.

Chapitre 2

Rappels de calcul différentiel

2.1 Fonctions différentiables : rappels essentiels

2.1.1 Fonctions de $\mathbb{R}^n$ dans $\mathbb{R}$

Soit $f : \mathbb{R}^n \to \mathbb{R}$, une fonction réelle à plusieurs variables. Elle associe à chaque vecteur $x = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$ une valeur réelle $f(x)$.

Exemple :

$$f(x_1, x_2) = x_1^2 + 3x_1x_2 + 2x_2^2$$

2.1.2 Dérivée directionnelle

La dérivée directionnelle de f en un point $x \in \mathbb{R}^n$ dans la direction d'un vecteur unitaire $v \in \mathbb{R}^n$ est définie par :

$$D_v f(x) = \lim_{t \to 0} \frac{f(x + tv) - f(x)}{t}$$

Elle mesure le taux de variation de f dans la direction de v.

2.1.3 Gradient

Le **gradient** de f, noté $\nabla f(x)$, est le vecteur des dérivées partielles premières :

$$\nabla f(x) = \begin{pmatrix} \frac{\partial f}{\partial x_1}(x) \\ \frac{\partial f}{\partial x_2}(x) \\ \vdots \\ \frac{\partial f}{\partial x_n}(x) \end{pmatrix}$$

Propriété : la dérivée directionnelle s'exprime à l'aide du gradient :

$$D_v f(x) = \nabla f(x)^T v$$

Le gradient indique donc la direction de la plus forte croissance de la fonction f.

Exemple : Pour $f(x_1, x_2) = x_1^2 + 3x_1x_2 + 2x_2^2$, on a :

$$\frac{\partial f}{\partial x_1} = 2x_1 + 3x_2, \quad \frac{\partial f}{\partial x_2} = 3x_1 + 4x_2$$

$$\Rightarrow \nabla f(x) = \begin{pmatrix} 2x_1 + 3x_2 \\ 3x_1 + 4x_2 \end{pmatrix}$$

2.1.4 Différentielle

La **différentielle** de f au point x, notée $df(x)$, est une application linéaire :

$$df(x) : \mathbb{R}^n \to \mathbb{R}, \quad df(x)(h) = \nabla f(x)^T h$$

Cela signifie que, pour une petite perturbation $h \in \mathbb{R}^n$, on a :

$$f(x + h) \approx f(x) + df(x)(h) = f(x) + \nabla f(x)^T h$$

Cette expression correspond à une approximation linéaire de la fonction autour du point x.

2.1.5 Interprétation géométrique

- Le gradient est orthogonal aux courbes (ou surfaces) de niveau de f.
- La différentielle permet d'approximer localement f par une fonction affine.
- Cela est à la base de nombreuses méthodes d'optimisation (ex. : descente de gradient).

2.1.6 Conditions de différentiabilité

Une fonction $f : \mathbb{R}^n \to \mathbb{R}$ est dite **différentiable** en un point $x \in \mathbb{R}^n$ si :

— toutes ses dérivées partielles premières existent autour de x,

— et sont continues dans un voisinage de x.

Dans ce cas, l'approximation $f(x + h) \approx f(x) + \nabla f(x)^T h$ est valide pour h petit.

Exercice 2.1. *Mini-exercice*

Soit $f(x, y) = \ln(x^2 + y^2)$, *avec* $(x, y) \neq (0, 0)$.

(a) Calculer le gradient $\nabla f(x, y)$.

(b) Donner une approximation de $f(x + h_1, y + h_2)$ *à l'aide de la différentielle.*

Correction

Soit $f(x, y) = \ln(x^2 + y^2)$, *avec* $x \neq 0$, $y \neq 0$.

1 Calcul du gradient $\nabla f(x, y)$

On calcule les dérivées partielles de f :

$$\frac{\partial f}{\partial x} = \frac{1}{x^2 + y^2} \cdot 2x = \frac{2x}{x^2 + y^2}$$

$$\frac{\partial f}{\partial y} = \frac{1}{x^2 + y^2} \cdot 2y = \frac{2y}{x^2 + y^2}$$

Le gradient est donc :

$$\nabla f(x, y) = \begin{pmatrix} \dfrac{2x}{x^2 + y^2} \\ \dfrac{2y}{x^2 + y^2} \end{pmatrix}$$

2 Approximation de $f(x + h_1, y + h_2)$ *à l'aide de la différentielle*

La différentielle de f *en* (x, y) *appliquée à* $h = (h_1, h_2)$ *est :*

$$df(x, y)(h) = \nabla f(x, y)^T \cdot h = \frac{2x}{x^2 + y^2} h_1 + \frac{2y}{x^2 + y^2} h_2$$

L'approximation affine de f *est donc :*

$$f(x + h_1, y + h_2) \approx f(x, y) + df(x, y)(h)$$

$$\Rightarrow f(x + h_1, y + h_2) \approx \ln(x^2 + y^2) + \frac{2x}{x^2 + y^2} h_1 + \frac{2y}{x^2 + y^2} h_2$$

2.2 Différentielle de Fréchet

La notion de différentielle de Fréchet généralise la dérivée classique à des fonctions définies entre espaces vectoriels normés. Elle permet de formaliser la linéarisation optimale d'une application au voisinage d'un point donné. Cette notion est essentielle dans le cadre de l'analyse fonctionnelle et de l'optimisation en dimension finie ou infinie.

2.2.1 Définition

Soient E et F deux espaces vectoriels normés, et soit $f : E \to F$ une application. On dit que f est **différentiable au sens de Fréchet** en un point $x_0 \in E$ s'il existe une application linéaire continue $L : E \to F$ telle que :

$$\lim_{h \to 0} \frac{\|f(x_0 + h) - f(x_0) - L(h)\|_F}{\|h\|_E} = 0.$$

Dans ce cas, on note $Df(x_0) = L$ la **différentielle de Fréchet** de f en x_0.
L'application L est l'approximation linéaire optimale de f au voisinage de x_0.

2.2.2 Remarques

- La différentielle de Fréchet, si elle existe, est unique.
- Si f est différentiable au sens de Fréchet en x_0, alors f est continue en x_0.
- En dimension finie, la différentielle de Fréchet coïncide avec la matrice jacobienne (vue comme application linéaire).

2.2.3 Lien avec la dérivabilité classique

Soit $f : \mathbb{R}^n \to \mathbb{R}^m$ une fonction. Alors, f est différentiable au sens de Fréchet en x_0 si et seulement si elle est différentiable au sens classique en x_0, et la différentielle de Fréchet est alors donnée par la matrice jacobienne de f en x_0 :

$$Df(x_0)(h) = J_f(x_0) \cdot h,$$

où $J_f(x_0)$ est la matrice jacobienne de f en x_0, et $h \in \mathbb{R}^n$.

Exemple 2.1. *Soit la fonction $f : \mathbb{R}^2 \to \mathbb{R}$ définie par :*

$$f(x, y) = x^2 + xy + y^2.$$

Calculons la différentielle de Fréchet en un point (x_0, y_0).

La différentielle est définie par :

$$Df(x_0, y_0)(h_1, h_2) = \frac{d}{dt} f(x_0 + th_1, y_0 + th_2)\Big|_{t=0}.$$

On dérive :

$$f(x_0 + th_1, y_0 + th_2) = (x_0 + th_1)^2 + (x_0 + th_1)(y_0 + th_2) + (y_0 + th_2)^2.$$

On obtient :

$$Df(x_0, y_0)(h_1, h_2) = 2x_0h_1 + y_0h_1 + x_0h_2 + 2y_0h_2 = (2x_0 + y_0)h_1 + (x_0 + 2y_0)h_2.$$

Donc, la différentielle de Fréchet en (x_0, y_0) *est l'application linéaire :*

$$Df(x_0, y_0) : (h_1, h_2) \mapsto (2x_0 + y_0)h_1 + (x_0 + 2y_0)h_2.$$

Ce qui correspond au produit scalaire :

$$Df(x_0, y_0)(h) = \nabla f(x_0, y_0)^T \cdot h,$$

avec

$$\nabla f(x_0, y_0) = \begin{pmatrix} 2x_0 + y_0 \\ x_0 + 2y_0 \end{pmatrix}.$$

2.3 La matrice hessienne

2.3.1 Définition

Soit $f : \mathbb{R}^n \to \mathbb{R}$ une fonction deux fois dérivable. La **matrice hessienne** de f en un point $x \in \mathbb{R}^n$ est la matrice carrée $H_f(x) \in \mathbb{R}^{n \times n}$ des dérivées partielles secondes, définie par :

$$H_f(x) = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2}(x) & \frac{\partial^2 f}{\partial x_1 \partial x_2}(x) & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n}(x) \\ \frac{\partial^2 f}{\partial x_2 \partial x_1}(x) & \frac{\partial^2 f}{\partial x_2^2}(x) & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n}(x) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1}(x) & \frac{\partial^2 f}{\partial x_n \partial x_2}(x) & \cdots & \frac{\partial^2 f}{\partial x_n^2}(x) \end{pmatrix}$$

2.3.2 Symétrie de la hessienne

Si f est de classe $\mathcal{C}^2$, c'est-à-dire que toutes ses dérivées partielles secondes sont continues, alors la matrice hessienne est **symétrique** :

$$\frac{\partial^2 f}{\partial x_i \partial x_j} = \frac{\partial^2 f}{\partial x_j \partial x_i} \quad \text{(théorème de Schwarz)}$$

2.3.3 Interprétation quadratique

La hessienne intervient dans le développement de Taylor à l'ordre 2 :

$$f(x+h) \approx f(x) + \nabla f(x)^T h + \frac{1}{2} h^T H_f(x) h$$

Ce terme quadratique est essentiel pour analyser la nature des points critiques.

2.3.4 Rôle en optimisation : test de la hessienne

Soit $x^* \in \mathbb{R}^n$ tel que $\nabla f(x^*) = 0$ (point critique). Alors :

- Si $H_f(x^*)$ est définie **positive** (toutes ses valeurs propres sont > 0), alors x^* est un **minimum local**.
- Si $H_f(x^*)$ est définie **négative**, alors x^* est un **maximum local**.
- Si $H_f(x^*)$ a des valeurs propres de signes opposés, alors x^* est un **point selle**.
- Si $H_f(x^*)$ est semi-définie, il faut une analyse plus fine.

Exemple 2.2. *Soit $f(x,y) = x^2 + 4xy + y^2$. On calcule la hessienne :*

$$\frac{\partial f}{\partial x} = 2x + 4y, \quad \frac{\partial f}{\partial y} = 4x + 2y$$

$$H_f(x,y) = \begin{pmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial y \partial x} & \frac{\partial^2 f}{\partial y^2} \end{pmatrix} = \begin{pmatrix} 2 & 4 \\ 4 & 2 \end{pmatrix}$$

Calculons les valeurs propres :

$$\det \begin{pmatrix} 2-\lambda & 4 \\ 4 & 2-\lambda \end{pmatrix} = (2-\lambda)^2 - 16 = \lambda^2 - 4\lambda - 12 \Rightarrow \lambda_1 = 6,\ \lambda_2 = -2$$

*Les valeurs propres sont de signes opposés implique (f) admet un **point selle** en tout point critique.*

Remarque 2.1. *La matrice hessienne est centrale dans les méthodes d'optimisation numérique de type Newton, où elle sert à corriger l'approximation de la direction de descente.*

2.4 Théorème des fonctions implicites

2.4.1 Motivation

Dans de nombreux problèmes d'optimisation avec contraintes, on cherche à exprimer certaines variables comme fonctions d'autres à partir d'une équation implicite :

$$F(x, y) = 0$$

où $F : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^m$ est une fonction différentiable. Le théorème des fonctions implicites garantit, sous certaines conditions, qu'il est possible localement d'écrire y comme une fonction de x.

Theorem 2.4.1:

Soit $F : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^m$ une fonction de classe $\mathcal{C}^1$ (continûment différentiable). Soit $(x_0, y_0) \in \mathbb{R}^n \times \mathbb{R}^m$ tel que :

$$F(x_0, y_0) = 0 \quad \text{et} \quad \det\left(\frac{\partial F}{\partial y}(x_0, y_0)\right) \neq 0$$

Alors, il existe un voisinage $U \subset \mathbb{R}^n$ de x_0, un voisinage $V \subset \mathbb{R}^m$ de y_0, et une fonction $g : U \to V$, de classe $\mathcal{C}^1$, telle que :

$$\forall x \in U, \quad F(x, g(x)) = 0 \quad \text{et} \quad g(x_0) = y_0$$

Autrement dit, on peut résoudre localement $F(x, y) = 0$ sous la forme $y = g(x)$.

2.4.2 Conséquences en optimisation

Le théorème permet :

— d'éliminer des variables dans les contraintes,

— de reformuler un problème sous contraintes comme un problème sans contrainte,

— de justifier l'utilisation de multiplicateurs de Lagrange (voir conditions de KKT).

Exemple 2.3. *Soit la contrainte* $F(x, y) = x^2 + y^2 - 1 = 0$. *On souhaite exprimer* y *en fonction de* x, *près de* $x_0 = 0$, $y_0 = 1$.

On a :

$$\frac{\partial F}{\partial y}(x, y) = 2y, \quad \text{et} \quad \frac{\partial F}{\partial y}(0, 1) = 2 \neq 0$$

Donc, d'après le théorème, il existe une fonction $y = g(x)$ *définie dans un voisinage de 0 telle que* $x^2 + g(x)^2 = 1$, *avec* $g(0) = 1$. *On retrouve ici* $g(x) = \sqrt{1 - x^2}$ *(localement).*

2.4.3 Formule pour la dérivée de la fonction implicite

Si $y = g(x)$ satisfait $F(x, g(x)) = 0$, alors la dérivée de g est donnée par :

$$Dg(x) = -\left(\frac{\partial F}{\partial y}(x, g(x))\right)^{-1} \cdot \frac{\partial F}{\partial x}(x, g(x))$$

Cette formule est utile pour calculer les gradients de fonctions contraintes.

Chapitre 3

Convexité

La notion de convexité occupe une place centrale en analyse mathématique et en optimisation. Elle permet de caractériser la forme globale d'une fonction ou d'un ensemble et de relier des propriétés géométriques à des comportements analytiques. Intuitivement, une fonction convexe est une fonction dont la courbe est située en dessous de ses cordes, traduisant une croissance régulière sans oscillations excessives. L'étude de la convexité fournit des outils puissants pour l'analyse des variations, la recherche d'extremum et la résolution de nombreux problèmes d'optimisation, aussi bien en mathématiques théoriques que dans des domaines appliqués tels que l'économie, la physique ou les sciences de l'ingénieur. Ce chapitre présente les définitions fondamentales de la convexité, ses principales propriétés et les critères analytiques permettant de l'identifier, notamment à l'aide des dérivées.

3.1 Convexité et optimisation

3.1.1 Ensembles convexes

Définition 3.1. *Un ensemble $C \subset \mathbb{R}^n$ est dit* ***convexe*** *si, pour tous $x, y \in C$ et pour tout $\lambda \in [0,1]$, le point $z = \lambda x + (1-\lambda)y$ appartient également à C.*

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

Exemple 3.1. *Les exemples suivants sont convexes :*

- *L'espace $\mathbb{R}^n$,*
- *Un sous-espace vectoriel,*
- *Une boule fermée $\{x \in \mathbb{R}^n : \|x - x_0\| \leq r\}$,*
- *Un polyèdre défini par des inégalités linéaires $Ax \leq b$.*

3.1.2 Fonctions convexes

Définition 3.2. *Une fonction $f : \mathbb{R}^n \to \mathbb{R}$ est dite* ***convexe*** *sur un ensemble convexe $C \subset \mathbb{R}^n$ si, pour tous $x, y \in C$ et tout $\lambda \in [0,1]$, on a :*

$$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda) f(y)$$

Définition 3.3 (Fonction strictement convexe)**.** *La fonction f est* ***strictement convexe*** *si l'inégalité est stricte pour $x \neq y$:*

$$f(\lambda x + (1-\lambda)y) < \lambda f(x) + (1-\lambda) f(y)$$

Exemple 3.2. *Les fonctions suivantes sont convexes :*

- $\surd$ $f(x) = x^2$,
- $\surd$ $f(x) = e^x$,
- $\surd$ $f(x) = \|x\|$,
- $\surd$ $f(x) = -\ln(x)$ *sur* $(0, \infty)$.

3.1.3 Caractérisation par le gradient (fonction différentiable)

Theorem 3.1.1:

Si $f : \mathbb{R}^n \to \mathbb{R}$ est de classe $\mathcal{C}^1$ sur un ensemble convexe C, alors f est convexe si et seulement si :

$$\forall x, y \in C, \quad f(y) \geq f(x) + \nabla f(x)^T (y - x)$$

Cette inégalité exprime que le graphe de f est toujours au-dessus de son plan tangent.

3.1.4 Caractérisation par la hessienne (fonction deux fois dérivable)

Theorem 3.1.2:

Soit $f : \mathbb{R}^n \to \mathbb{R}$ de classe $\mathcal{C}^2$. Alors :

- $\surd$ f est convexe sur un ensemble convexe C si et seulement si sa matrice hessienne $H_f(x)$ est **semi-définie positive** sur C.
- $\surd$ f est strictement convexe si $H_f(x)$ est **définie positive** sur C.

3.1.5 Conséquences en optimisation

Theorem 3.1.3

Soit $f : \mathbb{R}^n \to \mathbb{R}$ convexe sur un ensemble convexe C.

- $\surd$ Si $x^* \in C$ vérifie $\nabla f(x^*) = 0$, alors x^* est un **minimum global**.
- $\surd$ Si f est strictement convexe, alors ce minimum est **unique**.

Exemple 3.3. *Considérons $f(x) = x^2 + 1$. On a $\nabla f(x) = 2x$, donc $x^* = 0$ est le seul point critique. Comme $f''(x) = 2 > 0$, f est strictement convexe implique $x^* = 0$ est le minimum global unique.*

3.1.6 Combinaisons convexes et opérations

Proposition 3.1. *Les propriétés suivantes sont vérifiées :*

- *La somme de fonctions convexes est convexe.*
- *La composition $g \circ f$ est convexe si f est convexe et g est convexe croissante.*
- *Le maximum de fonctions convexes est convexe :*

$$f(x) = \max\{f_1(x), f_2(x), \ldots, f_k(x)\}$$

3.1.7 Exemples en optimisation

Exemple 3.4. *Minimiser la fonction $f(x) = \|Ax - b\|_2^2$. Cette fonction est convexe car elle s'écrit :*

$$f(x) = (Ax - b)^T(Ax - b) = x^T A^T Ax - 2b^T Ax + b^T b$$

La hessienne est $H_f(x) = 2A^TA$*, semi-définie positive* $\Rightarrow$ f *est convexe.*

Exemple 3.5. *Minimiser* $f(x) = -\ln(x)$ *sur* $(0, \infty)$*. On a :*

$$f''(x) = \frac{1}{x^2} > 0 \Rightarrow f \text{ est strictement convexe.}$$

3.2 Conditions d'optimalité pour les problèmes sans contrainte

3.2.1 Problème général

Soit $f : \mathbb{R}^n \to \mathbb{R}$, une fonction de classe $\mathcal{C}^2$. On considère le problème suivant :

$$\min_{x \in \mathbb{R}^n} f(x)$$

On cherche à déterminer les conditions qui permettent de garantir :

- $\surd$ l'existence d'un minimum,
- $\surd$ l'unicité du minimum,
- $\surd$ la caractérisation des points candidats au minimum.

3.2.2 Existence du minimum global

Theorem 3.2.1: Weierstrass

Si f est continue sur un ensemble fermé et borné $K \subset \mathbb{R}^n$, alors f admet un minimum global sur K.

Remarque 3.1. *Dans* $\mathbb{R}^n$*, si* f *est continue et* $\lim_{\|x\| \to \infty} f(x) = +\infty$*, alors* f *admet un minimum global.*

3.2.3 Unicité du minimum

Proposition 3.2. *Si* f *est strictement convexe sur un ensemble convexe* $C \subset \mathbb{R}^n$*, alors tout minimum global de* f *sur* C *est unique.*

—

3.2.4 Conditions nécessaires du premier ordre

Théorème 3.1. *Soit $x^* \in \mathbb{R}^n$ un point **localement optimal** pour f, supposée différentiable. Alors :*

$$\nabla f(x^*) = 0$$

Définition 3.4. *Un point x^* tel que $\nabla f(x^*) = 0$ est appelé un **point critique** ou **stationnaire**.*

—

3.2.5 Conditions du second ordre

Soit $f : \mathbb{R}^n \to \mathbb{R}$ deux fois continûment différentiable. Soit x^* un point critique de f, c'est-à-dire $\nabla f(x^*) = 0$.

Condition nécessaire du second ordre

Theorem 3.2.2

Si x^* est un minimum local de f, alors la hessienne de f en x^* est **semi-définie positive** :

$$\forall y \in \mathbb{R}^n, \quad y^T H_f(x^*) y \geq 0$$

Condition suffisante du second ordre

Theorem 3.2.3

Si $\nabla f(x^*) = 0$ et si la hessienne $H_f(x^*)$ est **définie positive**, alors x^* est un **minimum local strict** de f.

3.2.6 Résumé

$\surd$ $\nabla f(x^*) = 0$ est une **condition nécessaire** (1^{er} ordre).

$\surd$ $H_f(x^*) \succeq 0$ est une **condition nécessaire** (2^{e} ordre).

$\surd$ $H_f(x^*) \succ 0$ est une **condition suffisante** pour un minimum local strict.

$\surd$ Si f est convexe, alors tout point critique est un minimum global.

√ Si f est strictement convexe, le minimum global est unique.

—

Exemple 3.6. *Soit* $f(x,y) = x^2 + xy + y^2$.

√ $\nabla f(x,y) = \begin{pmatrix} 2x+y \\ x+2y \end{pmatrix} \Rightarrow \nabla f = 0 \Leftrightarrow x = 0, y = 0$

√ *La hessienne est :*

$$H_f = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}$$

Elle est définie positive (valeurs propres : $1, 3 > 0$*)*

Conclusion : $(0,0)$ *est un minimum local strict (et même global car* f *est convexe).*

3.3 Conditions d'optimalité pour les problèmes avec contraintes

3.3.1 Problème d'optimisation sous contraintes

Soit $f : \mathbb{R}^n \to \mathbb{R}$ une fonction à minimiser, sous des contraintes :

$$\begin{cases} h_i(x) = 0, \quad i = 1, \ldots, m \quad \text{(contraintes d'égalité)} \\ g_j(x) \leq 0, \quad j = 1, \ldots, p \quad \text{(contraintes d'inégalité)} \end{cases}$$

On cherche à résoudre :

$$\min_{x \in \mathbb{R}^n} f(x) \quad \text{tel que} \quad h_i(x) = 0,\ g_j(x) \leq 0$$

—

3.3.2 Multiplicateurs de Lagrange (cas sans inégalités)

Dans cette section, nous présentons une condition nécessaire du premier ordre pour les problèmes d'extremum avec contraintes. Le résultat est le théorème de Lagrange, bien connu. Pour mieux comprendre l'idée sous-jacente à ce théorème, nous considérons d'abord les fonctions de deux variables et une seule contrainte d'égalité. Soit h : la fonction de contrainte. Rappelons qu'en tout point x du domaine, le vecteur gradient $\nabla h(x)$ est orthogonal à l'ensemble de niveau passant par ce point. Choisissons en effet un point tel que $h(x^*) = 0$, et supposons que $\nabla h(x^*) \neq 0$.

L'ensemble de niveau passant par le point x^* est l'ensemble $\{x : h(x) = 0\}$. Nous paramétrons alors cet ensemble de niveau au voisinage de x^* par une courbe $\{x(t)\}$, c'est-à-dire une fonction vectorielle continûment différentiable, $x : \mathbb{R} \longrightarrow \mathbb{R}^2$ telle que

$$x(t) = \begin{pmatrix} x_1(t) \\ x_2(t) \end{pmatrix}, \quad t \in (a,b), \quad x^*(t) = x(t^*), \quad x'(t^*) \neq 0, \ t^* \in (a,b)$$

Nous pouvons maintenant montrer que $\nabla h(x^*)$ est orthogonal à $x'(t^*)$. En effet, puisque h est constante sur la courbe $\{x(t) : t \in (a,b)\}$, on a que pour tout $t \in (a,b)$, $h(x) = 0$.
Donc, pour tout $t \in (a,b)$, $\dfrac{d}{dt}h(x(t)) = 0$.
En appliquant la règle de la chaine (chain rule), on obtient

$$\frac{d}{dt}h(x(t)) = \nabla h(x(t))^T x'(t) = 0$$

par conséquent, $\nabla h(x^*)$ est orthogonal à $x'(t)$.
Supposons maintenant que x^* minimise $f : \mathbb{R}^2 \longrightarrow \mathbb{R}$ sur l'ensemble $\{x : h(x) = 0\}$. Nous affirmons que $\nabla f(x^*)$ est orthogonal à $x'(t^*)$. Pour le démontrer, il suffit d'observer que la fonction composée de t donnée par $\phi(t) = f(x(t))$ atteint un minimum en t^*. Par conséquent, la condition nécessaire du premier ordre pour le problème d'extremum sans contrainte implique que $\dfrac{d\phi}{dt}(t^*) = 0$.

L'application de la règle de la chaîne donne :

$$0 = \frac{d\phi}{dt}(t^*) = \nabla f(x(t^*))^T x'(t^*) = \nabla f(x^*)^T x'(t^*).$$

Ainsi, $\nabla f(x^*)$ est orthogonal à $x'(t^*)$. Le fait que $x'(t^*)$ soit tangente à la courbe $\{x(t)\}$ en x^* signifie que $\nabla f(x^*)$ est orthogonal à la courbe en x^* (voir 3.3.2)
Rappelons que $\nabla h(x^*)$ est également orthogonal à $\{x(t)\}$. Par conséquent, les vecteurs $\nabla h(x^*)$ et $\nabla f(x^*)$ sont parallèles ; autrement dit, $\nabla f(x^*)$ est un multiple scalaire de $\nabla f(x^*)$. Ces observations nous permettent maintenant de formuler le théorème de Lagrange pour les fonctions de deux variables avec une contrainte.
On a le théorème de Lagrange pour $n = 2$, $m = 1$.

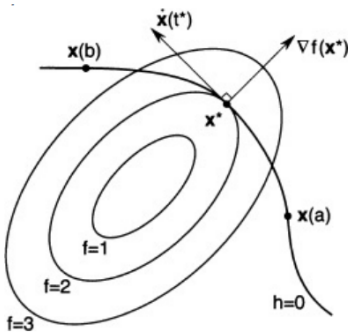


FIGURE 3.1 – Le gradient $\nabla f(x^*)$ est orthogonal à la courbe $\{x(t)\}$ au point x^* qui minimise f sur la courbe.

Theorem 3.3.1

Soit le point x^* minimisant $f : \mathbb{R}^2 \longrightarrow \mathbb{R}$, sous la contrainte $h(x) = 0$, $h : \mathbb{R}^2 \longrightarrow \mathbb{R}$. Alors,
$\nabla f(x^*)$ et $\nabla h(x^*)$ sont parallèles. Autrement dit, si $\nabla h(x^*) \neq 0$, alors il existe un scalaire λ^* tel que $\nabla f(x^*) + \lambda^* \nabla h(x^*) = 0$

Dans ce cas, nous appelons λ^* le multiplicateur de Lagrange. Notons que le théorème est également valable pour les problèmes de maximisation.

Le théorème de Lagrange fournit une condition nécessaire du premier ordre pour qu'un point soit un minimum local. Cette condition, que nous appelons condition de Lagrange, se compose de deux équations

$$\begin{cases} \nabla f(x^*) + \lambda^* \nabla h(x^*) = 0 \\ h(x^*) = 0 \end{cases}$$

On remarque que la condition de Lagrange est nécessaire mais non suffisante. Dans la figure 3.3.2, nous illustrons divers points où la condition de Lagrange est satisfaite, y compris un cas où le point n'est pas un extremum (ni un minimum, ni un maximum). Nous généralisons maintenant le théorème de Lagrange au cas où $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ et $h : \mathbb{R}^n \longrightarrow \mathbb{R}^m$, $m \leq n$

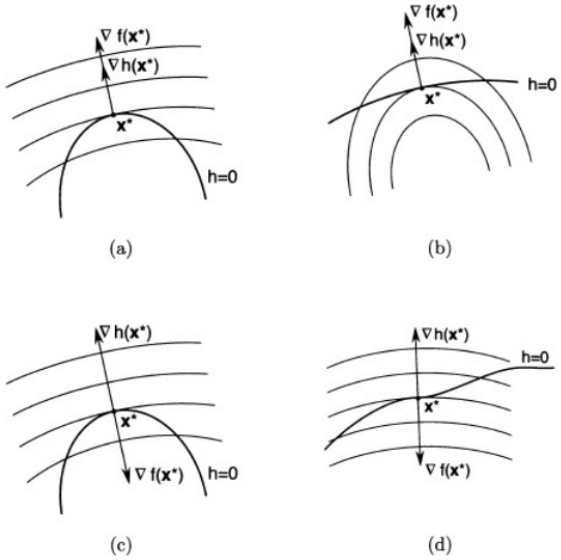


FIGURE 3.2 – Quatre exemples où la condition de Lagrange est satisfaite : (a) maximum, (b) minimum, (c) minimum, (d) point non extremum.

Theorem 3.3.2: Conditions de Lagrange — 1er ordre

Soit $x^* \in \mathbb{R}^n$ une solution locale du problème :

$$\min f(x) \quad \text{sous} \quad h_i(x) = 0,\ i = 1, \ldots, m$$

Supposons que les gradients $\nabla h_1(x^*), \ldots, \nabla h_m(x^*)$ soient linéairement indépendants. Alors il existe des multiplicateurs $\lambda_1, \ldots, \lambda_m \in \mathbb{R}$ tels que :

$$\nabla f(x^*) + \sum_{i=1}^{m} \lambda_i \nabla h_i(x^*) = 0$$

$$h_i(x^*) = 0, \quad \text{pour tout } i$$

Définition 3.5. *On appelle* ***fonction de Lagrange*** *:*

$$\mathcal{L}(x, \lambda) = f(x) + \sum_{i=1}^{m} \lambda_i h_i(x)$$

Le théorème de Lagrange stipule que si x^* est un extrémum, alors le gradient de la fonction objectif f peut être exprimé comme une combinaison linéaire des gradients des contraintes.

Dans le théorème **??**, le vecteur λ^* est appelé vecteur multiplicateur de Lagrange,

et ses composantes, multiplicateurs de Lagrange.

Si cette condition n'est pas vérifiée, alors x^* ne peut pas être un extrémum. Cette situation est illustrée dans la figure 3.3.2.

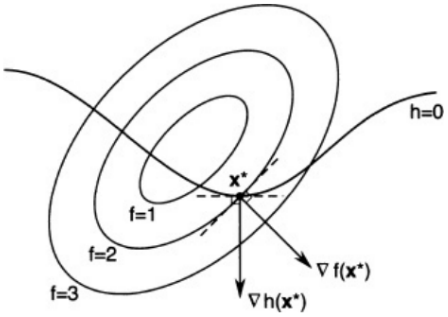


FIGURE 3.3 – Exemple dans le cas où la condition de Lagrange n'est pas vérifiée.

On remarque que la régularité est une hypothèse du théorème de Lagrange. Cette hypothèse est essentielle, comme le montre l'exemple suivant.

Exemple 3.7. *Considérons le problème suivant :*

$$\begin{cases} \min f(x) \\ S.C \quad h(x) = 0 \end{cases}$$

où $f(x) = x$ *et*

$$h(x) = \begin{cases} x^2 \quad si \quad x < 0 \\ 0 \quad si \quad 0 \leq x \leq 1 \\ (x-1)^2 \quad si \quad x > 1. \end{cases}$$

L'ensemble admissible est manifestement $[0, 1]$. Clairement, $x^* = 0$ est un minimiseur local. Cependant, $f'(x^*) = 1$ et $h'(x^*) = 0$. Par conséquent, x^* ne satisfait pas la condition nécessaire du théorème de Lagrange.

Remarquons toutefois que x^* n'est pas un point régulier, ce qui explique pourquoi le théorème de Lagrange ne s'applique pas ici.

Il est commode d'introduire la fonction lagrangienne $l : \mathbb{R}^n \times \mathbb{R}^m \longrightarrow \mathbb{R}$, définie par $l(x, \lambda) \triangleq f(x) + \lambda^T h(x)$

La condition de Lagrange pour un minimum local x^* peut être représentée à l'aide de la fonction Lagrangienne comme suit :

$$Dl(x^*, \lambda^*) = 0^T$$

pour un certain λ^*, où la dérivée D est prise par rapport à l'argument entier $[x, \lambda]$. En d'autres termes, la condition nécessaire du théorème de Lagrange est équivalente à la condition nécessaire du premier ordre pour l'optimisation sans contrainte appliquée à la fonction lagrangienne.

Pour illustrer ce qui précède, notons la dérivée de l par rapport à x par $D_x l$ et la dérivée de l par rapport à λ par $D_\lambda l$. Alors,

Remarquons que $D_x l(x, \lambda) = Df(x) + \lambda^T Dh(x)$ et $D_x l(x, \lambda) = h(x)$. Par conséquent,

$$\begin{cases} D_x l(x^*, \lambda^*) = 0^T \\ D_\lambda l(x^*, \lambda^*) = 0^T \end{cases}$$

le théorème de Lagrange pour un minimum local x^* peut s'énoncer comme suit :

pour un certain λ^*, ce qui est équivalent à

$Dl(x^*, \lambda^*) = 0^T$

Autrement dit, la condition de Lagrange peut s'exprimer par $Dl(x^*, \lambda^*) = 0$.

La condition de Lagrange est utilisée pour trouver d'éventuels extrémums.

Cela implique la résolution des équations

$$\begin{cases} D_x l(x^*, \lambda^*) = 0^T, \\ D_\lambda l(x^*, \lambda^*) = 0^T. \end{cases}$$

Le système ci-dessus représente $n+m$ équations à $n+m$ inconnues. Il est important de noter que la condition de Lagrange est nécessaire mais non suffisante ; autrement dit, un point x^* satisfaisant les équations ci-dessus n'est pas nécessairement un extremum.

Exemple 3.8. *Considérons le problème de l'extrémisation de la fonction objectif* $f(x) = x_1^2 + x_2^2$
sur l'ellipse,

$$[x_1, x_2]^T : h(x) = x_1^2 + 2x_2^2 - 1 = 0$$

On a,

$\nabla f(x) = [2x_1, 2x_2]^T$
$\nabla h(x) = [2x_1, 4x_2]^T$

Ainsi,

$$D_x l(x, \lambda) = [f(x) + \lambda h(x)] = [2x_1 + 2\lambda x_1, 2x_2 + 4\lambda x_2]$$

et

$$D_\lambda l(x,\lambda) = h(x) = x_1^2 + 2x_2^2 - 1$$

En posant $D_x l(x,\lambda) = 0$ *et* $D_\lambda l(x,\lambda) = 0$, *nous obtenons trois équations à trois inconnues,*

$2x_1 + 2\lambda x_1 = 0$
$2x_2 + 4\lambda x_2 = 0$
$x_1^2 + 2x_2^2 = 1$

Tous les points admissibles de ce problème sont réguliers. À partir de la première des équations ci-dessus, nous obtenons soit $x_1 = 0$, *soit* $\lambda = -1$. *Dans le cas où* $x_1 = 0$, *les deuxième et troisième équations impliquent que* $\lambda = \frac{1}{2}$ *et* $x_2 = \pm\frac{1}{\sqrt{2}}$. *Dans le cas où* $\lambda = 1$, *les deuxième et troisième équations impliquent que* $x_1 = \pm 1$ *et* $x_2 = 0$. *Ainsi, les points qui satisfont la condition de Lagrange pour les extrema sont*

$$x^{(1)} = \begin{bmatrix} 0 \\ \frac{1}{\sqrt{2}} \end{bmatrix}, \quad x^{(2)} = \begin{bmatrix} 0 \\ -\frac{1}{\sqrt{2}} \end{bmatrix}, \quad x^{(3)} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad x^{(4)} = \begin{bmatrix} -1 \\ 0 \end{bmatrix}$$

Puisque, $f(x^{(1)}) = f(x^{(2)} = \frac{1}{2}$ *et* $f(x^{(3)}) = f(x^{(4)} = 1$
nous en concluons que s'il existe des minimums, ils sont situés en $x^{(1)}$ *et* $x^{(2)}$ *et s'il existe des maximums, ils sont situés en* $x^{(3)}$ *et* $x^{(4)}$.
Il s'avère qu'en effet, $x^{(1)}$ *et* $x^{(2)}$ *sont des minimums et* $x^{(3)}$ *et* $x^{(4)}$ *sont des maximums. Ce problème peut être résolu graphiquement, comme illustré dans la figure 20.14.*

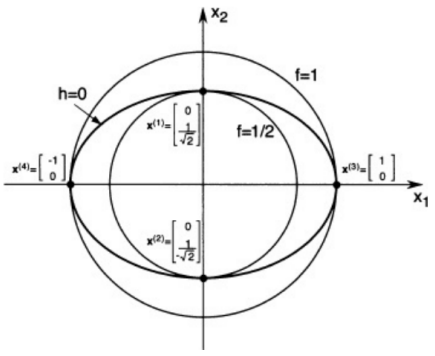


FIGURE 3.4 – Solution graphique du problème de l'exemple 3.8

3.3.3 Conditions de Karush-Kuhn-Tucker (KKT)

Nous avons analysé les problèmes d'optimisation sous contraintes comportant uniquement des contraintes d'égalité.
Dans cette partie, nous abordons les problèmes d'extremum qui comportent également des contraintes d'inégalité. Le traitement présenté ici est similaire à celui du précédente. En particulier, comme nous le verrons, les problèmes avec contraintes d'inégalité peuvent également être traités à l'aide des multiplicateurs de Lagrange.
On considère le problème suivant :

$$Minimiser f(x)\ S.C\ \ h(x) = 0 \qquad g(x) \leq 0$$

où $f : \mathbb{R}^n \longrightarrow \mathbb{R}$, $h : \mathbb{R}^n \longrightarrow \mathbb{R}^m$ tel que $m \leq n$, et $g : \mathbb{R}^n \longrightarrow \mathbb{R}^p$. Pour le problème général ci-dessus, nous adoptons les définitions suivantes.

Définition 3.6. *Une contrainte d'inégalité $g_j(x)$ est dite active en x^* si $g_j(x^*) = 0$. Elle est inactive en x^* si $g_j(x^*) < 0$.*
Par convention, nous considérons qu'une contrainte d'égalité $h_i(x) = 0$ est toujours active.

Définition 3.7. *Soit x^* tel que $h(x^*) = 0$, $g(x^*) \leq 0$, et soit $J(x^*)$ l'ensemble des indices des contraintes d'inégalité actives : $J(x^*) \triangleq \{j : g_j(x^*) = 0\}$.*
Alors, nous disons que x^* est un point régulier si les vecteurs $\nabla h_i(x^*) = 0$, $\nabla g_j(x^*) = 0$, $1 \leq i \leq m$ et $j \in J(x^*)$ sont linéairement indépendants.

Nous démontrons maintenant une condition nécessaire du premier ordre pour qu'un point soit un minimum local. On appelle cette condition la condition de Karush-Kuhn-Tucker (KKT).
Dans la littérature, cette condition est parfois également appelée condition de Kuhn-Tucker.

Theorem 3.3.3: Condition de KKT

Soit $f,\ h,\ g \in \mathcal{C}$. Soit x^* un point régulier et un minimum local du problème de minimisation de f sous les contraintes : $h(x) = 0,\ g(x) \leq 0$. Alors, il existe $\lambda^* \in \mathbb{R}^m$ et $\mu^* \in \mathbb{R}^p$ tels que :

1. $\mu^* \geq 0$
2. $Df(x^*) + \lambda^{*T} Dh(x^*) + \mu^{*T} Dg(x^*) = 0^T$
3. $\mu^{*T} g(x^*) = 0$

Dans le théorème, λ^* est appelé vecteur des multiplicateurs de Lagrange et μ^* le vecteur des multiplicateurs de Karush-Kuhn-Tucker (KKT). Leurs composantes sont appelées respectivement multiplicateurs de Lagrange et multiplicateurs de Karush-Kuhn-Tucker (KKT).

Avant de démontrer ce théorème, examinons d'abord sa signification. On observe que $\mu_j^* \geq 0$ (d'après la condition 1) et $g_j(x)* \leq 0$. Par conséquent, la condition implique que si $g_j(x^*) < 0$, alors $\mu_j^* = 0$; c'est-à-dire que pour tout $j \notin J(x^*)$, on a $\mu_j^* = 0$.

Autrement dit, les multiplicateurs KKT μ_j^* correspondant aux contraintes inactives sont nuls. Les autres multiplicateurs KKT, $\mu_i^*,\ i \in J(x^*)$, sont non négatifs ; ils peuvent être ou non égaux à zéro.

Exemple 3.9. *Une illustration graphique du théorème KKT est donnée dans la figure 3.9. Dans cet exemple bidimensionnel, nous n'avons que des contraintes d'inégalité* $g_j(x) \leq 0, \quad j = 1, 2, 3.$

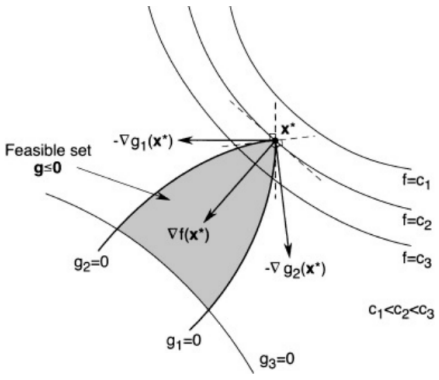


FIGURE 3.5 – Illustration du théorème de Karush-Kuhn-Tucker (KKT).

Remarquons que le point x^* sur la figure est bien un minimum. La contrainte $g_3(x) \leq 0$ est inactive : $g_3(x) < 0$; par conséquent $\mu_3^* = 0$. D'après le théorème de Karush-Kuhn-Tucker, nous avons

$$\nabla f(x^*) + \mu_1^* \nabla g_1(x^*) + \mu_2^* \nabla g_2(x^*) = 0$$

ou, de manière équivalente,

$$\nabla f(x^*) = -\mu_1^* \nabla g_1(x^*) - \mu_2^* \nabla g_2(x^*)$$

où μ_1^* et μ_2^*. Il est facile d'interpréter graphiquement la condition de KKT pour cet exemple. Plus précisément, on peut voir sur la figure ?? que $\nabla f(x^*)$ doit être une combinaison linéaire des vecteurs $-\nabla g_1(x^*)$ et $-\nabla g_2(x^*)$ à coefficients positifs.
Ceci se reflète exactement dans l'équation ci-dessus, où les coefficients μ_1^* et μ_2^* sont les multiplicateurs KKT.

On applique la condition de KKT de la même manière que toute condition nécessaire. Plus précisément, nous recherchons les points satisfaisant la condition de KKT et considérons ces points comme des minimums potentiels.

En résumé, la condition de KKT se compose de cinq parties (trois équations et deux inégalités) :

1. $\mu^* \geq 0$, (complémentarité)
2. $Df(x^*) + \lambda^{*T} Dh(x^*) + \mu^{*T} Dg(x^*) = 0^T$, (stationnarité)
3. $\mu^{*T} g(x^*) = 0$, (complémentarité)
4. $h(x^*) = 0$, (primalité)
5. $g(x^*) \leq 0$. (primalité)

—

3.3.4 Interprétation géométrique

√ La stationnarité exprime que le gradient de f est une combinaison linéaire des gradients des contraintes actives.

√ La complémentarité signifie que si une contrainte $g_j(x^*) < 0$ est inactive, alors son multiplicateur $\mu_j = 0$.

√ Les multiplicateurs λ_i, μ_j mesurent la sensibilité de la fonction objectif à la contrainte associée.

3.3.5 Conditions du second ordre (avec égalité)

Nous supposons que $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ et $h : \mathbb{R}^n \longrightarrow \mathbb{R}^m$, sont deux fois continûment différentiables : $f, h \in \mathcal{C}^2$. Soit $l(x, \lambda) = f(x) + \lambda^T h(x) = f(x) + \lambda_1 h_1(x) + ... + \lambda_m h_m(x)$ la fonction lagrangienne. Soit $L(x, \lambda)$ la matrice hessienne de $l(x, \lambda)$ par rapport à x : $L(x, \lambda) = F(x) + \lambda_1 H_1(x) + ... + H_m(x)$, où $F(x)$ est la matrice hessienne de f en x et $H_k(x)$ est la matrice hessienne de h_k en x, $k = 1, ..., m$. Donnée par

$$H_k(x) = \begin{bmatrix} \frac{\partial^2 h_k(x)}{\partial x_1^2} & \cdots & \frac{\partial^2 h_k(x)}{\partial x_n \partial x_1} \\ . & . & . \\ . & . & . \\ . & . & . \\ \frac{\partial^2 h_k(x)}{\partial x_1 \partial x_n} & \cdots & \frac{\partial^2 h_k(x)}{\partial x_n^2} \end{bmatrix}$$

Nous introduisons la notation $[\lambda H(x)]$:
$[\lambda H(x)] = \lambda_1 H_1(x), ..., \lambda_m H_m(x)$.

Theorem 3.3.4: Conditions nécessaire du second ordre

Soit x^* un minimum local de $f : \mathbb{R}^n \to \mathbb{R}$ avec la contrainte $h(x) = 0$, $h : \mathbb{R}^n \to \mathbb{R}^m$, $m \leq n$ et $f, h \in C^2$.
Supposons que x^* est un point régulier. Alors il existe un multiplicateur de Lagrange $\lambda^* \in \mathbb{R}^m$ tel que :

1. $Df(x^*) + \lambda^{*T} Dh(x^*) = 0^T$
2. pour tout $y \in T(x^*)$, on a $y^T L(x^*, \lambda^*) y \geq 0$

Observons que $H_L(x, \lambda)$ joue un rôle similaire à celui de la matrice hessienne $F(x)$ de la fonction objectif f. fait dans le cas de minimisation sans contraintes.

Cependant, nous exigeons maintenant que $L(x^*, \lambda^*) \geq 0$ uniquement sur $T(x^*)$ plutôt que sur $\mathbb{R}^n$.

Les conditions ci-dessus sont nécessaires, mais pas suffisantes, pour qu'un point soit un minimum local.
Nous présentons maintenant, sans preuve, des conditions suffisantes pour qu'un point soit un minimum local strict.

Theorem 3.3.5: Conditions suffisante du second ordre

Supposons que $f, h \in C^2$ et il existe un point $x^* \in \mathbb{R}^n$, et $\lambda^* \in \mathbb{R}^m$ tel que :

1. $Df(x^*) + \lambda^{*T} Dh(x^*) = 0^T$
2. pour tout $y \in T(x^*)$, $y \neq 0$ on a $y^T L(x^*, \lambda^*) y \geq 0$

Remarque 3.2. *L'explication la plus simple du théorème ?? est comme suit :*

Soit x^* un point stationnaire et λ les multiplicateurs associés. On définit l'espace tangent aux contraintes :

$$T = \{d \in \mathbb{R}^n \mid \nabla h_i(x^*)^T d = 0,\ i = 1, \ldots, m\}$$

Si :

$$\forall d \in T \setminus \{0\}, \quad d^T \nabla^2_{xx} \mathcal{L}(x^*, \lambda) d > 0$$

alors x^* est un minimum local strict.

Exemple 3.10. *Considérons le problème :*

$$maximiser \frac{x^T Q x}{x^T P x}$$

Où $Q = \begin{bmatrix} 4 & 0 \\ 0 & 1 \end{bmatrix}$ *et* $P = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}$.

Comme indiqué précédemment, nous pouvons ce problème sous la forme équivalente :

$$\begin{cases} Minimiser x^T Q x \\ s.c \quad x^T P x = 1 \end{cases}$$

La fonction objectif pour le problème transformé est donné par

$$L(x, \lambda) = x^T Q x + \lambda(1 - x^T P x),$$

Les conditions de Lagrange, donne :

$$(\lambda I - P^{-1} Q) x = 0$$

Où

$$P^{-1}Q = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}$$

Il n'y a que deux valeurs de λ*, satisfant* $(\lambda I - P^{-1}Q)x = 0$*, ce sont, les valeurs propres de* $P^{-1}Q^{-1}$*,* $\lambda = 2, \quad \lambda = 1$.

Nous rappelons de notre discussion précédente de ce problème que le multiplicateur de Lagrange correspondant à la solution est la valeur propre maximale de $P^{-1}Q$*, notée :* $\lambda^* = \lambda_1 = 2$.

Le vecteur propre est le maximum-la solution de ce problème. Le vecteur propre correspondant au valeur propre $\lambda^* = 2$*, satisfont la contrainte* $x^TQx = 1$ *est* $\pm x^*$ *où,*

$$x^* = [\frac{1}{\sqrt{2}}, \quad 0]$$

A ce stade, tout ce que nous avons établi est que les paires $(\pm x^*, \lambda^*)$ *satisfont la condition de Lagrange.*

Nous montrons maintenant que les points $\pm x^*$ *sont, en fait, des maximums locaux stricts.*

Notons qu'une procédure similaire s'applique à $-x^*$.

Nous calculons d'abord la matrice hessienne de la fonction Lagrangienne. On à :

$$H_L(x^*, \lambda^*) = 2Q - 2\lambda P = \begin{bmatrix} 0 & 0 \\ 0 & -2 \end{bmatrix}$$

$$T(x^*) = \{y \in \mathbb{R} : \quad x^{*T}Py = 0\} = \{y : \quad (\sqrt{2}, 0)y = 0\}$$

$= \{y = [0, a]^T, \quad a \in \mathbb{R}\}$

L'espace tangent $T(x^*)$ *a* $\{x : \quad 1 - x^TPx = 0\}$ *est*

$$y^TL(x^*, \lambda^*)y = (0, a)\begin{bmatrix} 0 & 0 \\ 0 & -2 \end{bmatrix}\begin{bmatrix} 0 \\ a \end{bmatrix} = -2a^2 < 0$$

Notons que pour chaque $y \in T(x^*), y \neq 0$

—

Exemple 3.11. *Minimiser* $f(x, y) = x^2 + y^2$ *sous la contrainte* $x + y - 1 = 0$.

1. *Formule de Lagrange :*

$$\mathcal{L}(x, y, \lambda) = x^2 + y^2 + \lambda(x + y - 1)$$

2. *Conditions de stationnarité :*

$$\begin{cases} \frac{\partial \mathcal{L}}{\partial x} = 2x + \lambda = 0 \\ \frac{\partial \mathcal{L}}{\partial y} = 2y + \lambda = 0 \\ x + y - 1 = 0 \end{cases} \Rightarrow \begin{cases} x = y \\ 2x + \lambda = 0 \Rightarrow \lambda = -2x \\ x + x = 1 \Rightarrow x = \frac{1}{2},\ y = \frac{1}{2} \end{cases}$$

Résultat : *minimum en* $(x^*, y^*) = \left(\frac{1}{2}, \frac{1}{2}\right)$

3.3.6 Conditions du second ordre pour un problème avec inégalités

Comme pour les problèmes d'extremum avec contraintes d'égalité, nous pouvons également donner des conditions nécessaires et suffisantes du second ordre pour les problèmes d'extremum avec contraintes d'inégalité. Pour cela, nous devons définir les matrices suivantes : $L(x, \lambda, \mu) = F(x) + [\lambda H(x)] + [\mu G(x)]$, où $F(x)$ est la matrice hessienne de f en x, et la notation $[\lambda H(x)]$ représente $[\lambda H(x)] = \lambda_1 H_1(x) + ... + \lambda_m H_m(x)$, comme précédemment.
De même, la notation $[\mu G(x)]$ représente $[\mu G(x)] = \mu_1 G_1(x) + ... + \mu_p G_p(x)$, où $G_k(x)$ est la matrice hessienne de g_k en x, donnée par

$$G_k(x) = \begin{bmatrix} \frac{\partial^2 g_k(x)}{\partial x_1^2} & \cdots & \frac{\partial^2 g_k(x)}{\partial x_n \partial x_1} \\ . & . & . \\ . & . & . \\ . & . & . \\ \frac{\partial^2 g_k(x)}{\partial x_1 \partial x_n} & \cdots & \frac{\partial^2 g_k(x)}{\partial x_n^2} \end{bmatrix}$$

Dans le théorème suivant, nous utilisons,

$$T(x^*) = \{y \in \mathbb{R}^n : Dh(x^*)y = 0,\ Dg_j(x^*)y = 0,\ j \in J(x^*)\}$$

,

c'est-à-dire l'espace tangent à la surface définie par les contraintes actives.

Theorem 3.3.6: Conditions nécessaire du second ordre

Soit x^* un minimum local de $f : \mathbb{R}^n \longrightarrow \mathbb{R}$
sous les contraintes $h(x) = 0$, $g(x) \leq 0$, $h : \mathbb{R}^n \longrightarrow \mathbb{R}^m$, $g : \mathbb{R}^n \longrightarrow \mathbb{R}^p$,
et $f,\ h,\ g\ \in \mathcal{C}^2$.
Supposons que x^* soit régulier. Alors, il existe $\lambda^* \in \mathbb{R}^m$ et $\mu^* \in \mathbb{R}^p$,
tels que :

1. $\mu^* \geq 0,\ Df(x^*) + \lambda^* Dh(x^*) + \mu^* Dg(x^*) = 0,\ \mu^* g(x^*) = 0.$
2. Pour tout $y \in T(x^*)$, on a $yL(x^*, \lambda^*, \mu^*)y \geq 0.$

Démonstration. La première partie découle simplement du théorème KKT. Pour démontrer la seconde partie, nous remarquons que, puisque le point x^* est un minimum local sur $\{x : h(x) = 0, g(x) \leq 0\}$, il est également un minimum local sur $\{x : h(x) = 0, gj(x) = 0, j \in J(x^*)\}$; autrement dit, le point x^* est un minimum local avec des contraintes actives considérées comme des contraintes d'égalité (voir l'exercice 21.16). Par conséquent, les conditions nécessaires du second ordre pour les contraintes d'égalité sont applicables ici, ce qui achève la démonstration.

Nous énonçons maintenant les conditions suffisantes du second ordre pour les problèmes d'extremum avec contraintes d'inégalité. Dans la formulation du résultat, nous utilisons l'ensemble suivant :

$$\tilde{T}(x^*, \mu^*) = \{y : Dh(x^*)y = 0,\ Dg_i(x^*)y = 0,\ i \in \tilde{J}(x^*, \mu^*)\},$$

où $\tilde{J}(x^*, \mu^*) \in \{i : g_i(x^*) = 0,\ \mu_i^* > 0\}$. Notons que $\tilde{J}(x^*, \mu^*)$ est un sous-ensemble de $J(x*)$, $\tilde{J}(x^*, \mu^*) \subseteq J(x^*)$. Ceci implique que $T(x^*)$ est un sous-ensemble de $\tilde{T}(x^*, \mu^*) : T(x^*) \subseteq \tilde{T}(x^*, \mu^*)$. □

Theorem 3.3.7: Conditions suffisante du second ordre

Supposons que $f,\ g,\ h\ \in \mathcal{C}^2$ et qu'il existe un point admissible $x^* \in \mathbb{R}^n$
et des vecteurs $\lambda^* \in \mathbb{R}^m$ et $\mu^* \in \mathbb{R}^p$ tels que :

1. $\mu^* \geq 0,\ Df(x^*) + \lambda^* Dh(x^*) + \mu^* Dg(x^*) = 0,\ \mu^* g(x^*) = 0.$
2. Pour tout $y \in \tilde{T}(x^*, \mu^*),\ y \neq 0$, on a $y^T L(x^*, \lambda^*, \mu^*)y \geq 0.$

Alors, x^* est un minimum local strict de f sous les contraintes $h(x) = 0$,
$g(x) \leq 0$.

Un résultat similaire aux conditions suffisantes de second ordre est valable pour un maximum local strict, la seule différence étant que l'on a besoin de $\mu^* \leq 0$ et que $L(x^*, \lambda^*)$ soit défini négatif sur $\tilde{T}(x^*, \mu^*)$.

Exemple 3.12. *Considérons le problème suivant :*

$$\begin{cases} Minimiser\ x_1 x_2 \\ s.c\ \ x_1 + x_2 \geqslant 2 \\ x_2 \geqslant x_1 \end{cases}$$

1. *Écrire les conditions de KKT pour ce problème.*
2. *Trouvez tous les points (et les multiplicateurs KKT) satisfaisant les conditions KKT. Dans chaque cas, déterminer si le point est régulier.*
3. *Trouvez tous les points de la partie b qui satisfont également le SONC.*
4. *Trouvez tous les points de la partie c qui satisfont également le SOSC.*
5. *Trouver tous les points de la partie c qui sont des minimums locaux.*

Solution :

1- On écrit $f(x) = x_1 x_2, \quad g_1(x) = 2 - x_1 - x_2, \ and g_2(x) = x_1 - x_2$.
Les conditions de KKT sont :

$$\begin{cases} x_2 - \mu_1 + \mu_2 = 0 \\ x_1 - \mu_1 - \mu_2 = 0 \\ \mu_1(2 - x_1 - x_2) + \mu_2(x_1 - x_2) = 0 \\ \mu_1, \quad \mu_2 \geqslant 0 \\ 2 - x_1 - x_2 \leqslant 0 \\ x_1 - x_2 \leqslant 0 \end{cases}$$

2- Il est facile de vérifier que $\mu_1 \neq 0$ *et* $\mu_2 \not> 0$.
Alors on sorte avec une seule solution au conditions de KKT : $x_1^* = x_2^* = 1$, $\mu_1^* = 1$, $\mu_2^* = 0$.
Pour ce point, on a $Dg_1(x^*) = [-1, \ -1]$ *et* $Dg_2(x^*) = [-1, \ -1]$. *Donc* x^* *est régulier.*

3- Les deux contraintes sont actives. Donc, puisque x^* *est régulier,* $T(x^*) = 0$.
Cela implique que le SONC est satisfait.

4- Maintenant,

$$H_L(x^*) = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

De plus, $(x^*, \mu^*) = \{y : [-1,\ -1]y = 0\} = \{y : y_1 = -y_2\}$. *Choisissons* $y = [1, 1]^T \in \tilde{T}(x^*, \mu^*)$. *Nous avons* $y^T L(x^{*,*})y = 2 < 0$, *ce qui signifie que le SOSC échoue.*

5. En fait, le point x^* *n'est pas un minimum local. Pour le voir, dessinons un graphique de l'ensemble des contraintes et des ensembles de niveau de la fonction objectif. En se déplaçant dans la direction admissible* $[1, 1]$, *la fonction objectif augmente ; mais en se déplaçant dans la direction admissible* $[1, 1]$, *la fonction objectif diminue.*

Exercices corrigés – Conditions d'optimalité avec contraintes

Exercice 1 – Méthode de Lagrange

Énoncé : Minimiser $f(x, y) = x^2 + y^2$ sous la contrainte $g(x, y) = x + y - 1 = 0$.

Solution :

On forme la fonction de Lagrange :

$$\mathcal{L}(x, y, \lambda) = x^2 + y^2 + \lambda(x + y - 1)$$

Conditions de stationnarité :

$$\begin{cases} \frac{\partial \mathcal{L}}{\partial x} = 2x + \lambda = 0 \\ \frac{\partial \mathcal{L}}{\partial y} = 2y + \lambda = 0 \\ x + y - 1 = 0 \end{cases}$$

Résolution :

$$2x = -\lambda, \quad 2y = -\lambda \Rightarrow x = y \Rightarrow x + x = 1 \Rightarrow x = y = \frac{1}{2}$$

Conclusion : Le minimum est atteint en $\left(\frac{1}{2}, \frac{1}{2}\right)$, avec $f = \frac{1}{2}$.

—

Exercice 2 – Conditions KKT

Énoncé : Minimiser $f(x, y) = x^2 + y^2$ sous $h(x, y) = x + y - 1 \leq 0$.

Solution :

On applique les conditions KKT avec :

$$\mathcal{L}(x, y, \mu) = x^2 + y^2 + \mu(x + y - 1)$$

Conditions :

$$\begin{cases} \nabla f(x, y) + \mu \nabla h(x, y) = 0 \\ x + y - 1 \leq 0 \\ \mu \geq 0 \\ \mu(x + y - 1) = 0 \end{cases}$$

On résout :

$$\begin{cases} 2x + \mu = 0 \Rightarrow x = -\mu/2 \\ 2y + \mu = 0 \Rightarrow y = -\mu/2 \\ x + y = -\mu \leq 1 \end{cases}$$

Mais on voit que $x + y = -\mu \leq 1 \Rightarrow$ toujours vérifié. Complémentarité : $\mu(x + y - 1) = 0 \Rightarrow \mu(-\mu - 1) = 0 \Rightarrow \mu = 0$ ou $\mu = -1$ (impossible car $\mu \geq 0$). Donc : $\mu = 0 \Rightarrow x = y = 0 \Rightarrow x + y = 0 < 1$, contrainte inactive.

Conclusion : Minimum global atteint en $(0, 0)$, $f = 0$, contrainte inactive.

Exercice 3 – Analyse du second ordre

Énoncé : On considère $f(x) = x_1^2 + x_2^2$ avec la contrainte $x_1 + x_2 = 1$. Montrer que le point obtenu par Lagrange est un minimum strict.

Solution :

On a déjà vu que $x^* = \left(\frac{1}{2}, \frac{1}{2}\right)$. La hessienne de f est $H_f = 2I$, définie positive sur $\mathbb{R}^2$. Le gradient de la contrainte est $\nabla g = (1, 1)^T$.

L'espace tangent :

$$T = \{d \in \mathbb{R}^2 : \nabla g^T d = d_1 + d_2 = 0\} \Rightarrow d_2 = -d_1$$

Soit $d = (d_1, -d_1) \Rightarrow d^T H_f d = 2(d_1^2 + d_1^2) = 4d_1^2 > 0$

Conclusion : La condition du second ordre est vérifiée $\Rightarrow$ minimum local strict.

—

Exercice 4 – Cas de contraintes inactives

Énoncé : Minimiser $f(x) = x^2$ sur $x \geq 1$.

Solution :

Minimum global de f est en $x = 0$, mais interdit par la contrainte.

Donc on cherche le minimum de f sur $[1, \infty)$.

Résultat : Minimum atteint en $x^* = 1$, $f(1) = 1$, avec contrainte active.

Conditions KKT :

$$\begin{cases} f'(x^*) + \mu \cdot h'(x^*) = 0 \Rightarrow 2x + \mu(-1) = 0 \Rightarrow \mu = 2 \\ x - 1 \geq 0,\ \mu \geq 0,\ \mu(x-1) = 0 \end{cases}$$

Tout est satisfait. On confirme que $x = 1$ est bien solution KKT.

Exercice 5 – Contrainte non linéaire

Minimiser $f(x, y) = x^2 + y^2$ sous la contrainte $g(x, y) = x^2 + y - 1 = 0$.

Solution :

On utilise Lagrange : $\mathcal{L}(x, y, \lambda) = x^2 + y^2 + \lambda(x^2 + y - 1)$

Conditions stationnaires :

$$\begin{cases} \frac{\partial \mathcal{L}}{\partial x} = 2x + 2\lambda x = 0 \\ \frac{\partial \mathcal{L}}{\partial y} = 2y + \lambda = 0 \\ x^2 + y - 1 = 0 \end{cases}$$

Cas 1 : $x = 0 \Rightarrow$ première équation satisfaite.

- Troisième équation : $y = 1$,
- Deuxième équation : $2 \cdot 1 + \lambda = 0 \Rightarrow \lambda = -2$

Donc $(x, y) = (0, 1)$, $f = 1$

Cas 2 : $x \neq 0 \Rightarrow 1 + \lambda = 0 \Rightarrow \lambda = -1$

Deuxième équation : $2y - 1 = 0 \Rightarrow y = 0.5$

Troisième : $x^2 + 0.5 = 1 \Rightarrow x^2 = 0.5 \Rightarrow x = \pm\frac{1}{\sqrt{2}}$

On obtient deux autres candidats : $\left(\pm\frac{1}{\sqrt{2}},\ 0.5\right)$, $f = \frac{1}{2} + \frac{1}{4} = \frac{3}{4}$

Conclusion : Le minimum est atteint en $x = \pm\frac{1}{\sqrt{2}}, y = 0.5$, valeur minimale $f = \frac{3}{4}$

—

Exercice 6 – Deux contraintes actives

Minimiser $f(x, y) = x^2 + y^2$, sous :

$$\begin{cases} x + y \geq 1 \\ x \geq 0 \end{cases}$$

Solution :

Écrivons les contraintes sous forme $h_1(x, y) = 1 - x - y \leq 0$, $h_2(x, y) = -x \leq 0$

Stationnarité (KKT) :

$$\nabla f + \mu_1 \nabla h_1 + \mu_2 \nabla h_2 = 0 \Rightarrow \begin{cases} 2x - \mu_1 - \mu_2 = 0 \\ 2y - \mu_1 = 0 \end{cases}$$

Complémentarité :

$$\mu_1(1 - x - y) = 0, \quad \mu_2(-x) = 0$$

Essayons $\mu_1 \neq 0$, $\mu_2 \neq 0 \Rightarrow$ contraintes actives : $x + y = 1$, $x = 0 \Rightarrow y = 1$

Vérifions les équations :

$$\begin{cases} 2x - \mu_1 - \mu_2 = 0 \Rightarrow -\mu_1 - \mu_2 = 0 \Rightarrow \mu_1 = -\mu_2 \\ 2y - \mu_1 = 0 \Rightarrow 2 - \mu_1 = 0 \Rightarrow \mu_1 = 2 \Rightarrow \mu_2 = -2 \end{cases}$$

Problème : $\mu_2 = -2 < 0$, interdit.

Essayons maintenant : **Contrainte 1 active, contrainte 2 inactive** $\rightarrow x + y = 1$, $x > 0$

Alors $\mu_2 = 0$, $\mu_1 \geq 0$

$$\begin{cases} 2x = \mu_1 \\ 2y = \mu_1 \Rightarrow x = y \Rightarrow x + x = 1 \Rightarrow x = y = \frac{1}{2} \end{cases}$$

Vérifications :

$$x + y = 1,\ x > 0,\ \mu_1 = 1 \geq 0,\ \mu_1(1 - x - y) = 0 \Rightarrow \text{Tout est OK}$$

Conclusion : minimum en $(1/2, 1/2)$, $f = 1/2$

—

Exercice 7 – Deux contraintes d'inégalités, point à la frontière

Minimiser $f(x, y) = (x-1)^2 + (y-2)^2$ sous les contraintes :

$$\begin{cases} x \geq 0 \\ y \geq 0 \end{cases}$$

Solution : minimum sans contrainte à $x = 1,\ y = 2$ — admissible $\Rightarrow$ c'est aussi le minimum sous contraintes.

Supposons maintenant qu'on ajoute une contrainte : $x + y \leq 1$

Nouveau problème :

$$\min f(x, y) = (x-1)^2 + (y-2)^2 \quad \text{sous} \quad \begin{cases} x + y \leq 1 \\ x \geq 0,\ y \geq 0 \end{cases}$$

Le minimum libre $(1, 2)$ est interdit (contrainte violée). On cherche le point le plus proche de $(1, 2)$ sur le triangle :

$$\mathcal{C} = \{(x, y) \in \mathbb{R}^2 : x + y \leq 1,\ x \geq 0,\ y \geq 0\}$$

Ce problème revient à minimiser la distance euclidienne à $(1, 2)$ sur le triangle.

La solution est le **point de $\mathcal{C}$** le plus proche de $(1, 2)$. On peut démontrer (ou dessiner) que ce point est $(0, 1)$

$$f(0, 1) = (0-1)^2 + (1-2)^2 = 1 + 1 = 2$$

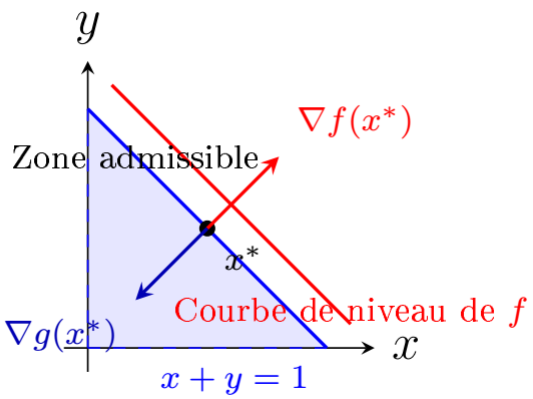


FIGURE 3.6 – Point optimal x^* au bord de la contrainte active $x + y \leq 1$

Chapitre 4

Algorithmes de minimisation sans contrainte

On considère un problème de minimisation sans contrainte :

$$\min_{x \in \mathbb{R}^n} f(x)$$

où $f : \mathbb{R}^n \to \mathbb{R}$ est supposée **continûment différentiable**.

Définition 4.1. *Un vecteur $d \in \mathbb{R}^n$ est une **direction de descente** pour f en un point x si :*

$$\nabla f(x)^T d < 0$$

Cela garantit que pour un pas $\alpha > 0$ suffisamment petit, on a :

$$f(x + \alpha d) < f(x)$$

4.1 Algorithme de descente général

Principe de l'algorithme

L'algorithme de descente général construit une suite $(x_k)_{k \in \mathbb{N}}$ définie par :

$$x_{k+1} = x_k + \alpha_k d_k$$

où :

- $\surd$ $x_k \in \mathbb{R}^n$ est le point courant à l'itération k,
- $\surd$ $d_k \in \mathbb{R}^n$ est une direction de descente telle que $\nabla f(x_k)^T d_k < 0$,

$\surd$ $\alpha_k > 0$ est un pas de descente.

Choix de la direction d_k

Différents choix sont possibles :

$\surd$ Descente du gradient : $d_k = -\nabla f(x_k)$,

$\surd$ Méthode de Newton : $d_k = -[\nabla^2 f(x_k)]^{-1}\nabla f(x_k)$,

$\surd$ Méthodes quasi-Newton : $d_k = -B_k^{-1}\nabla f(x_k)$,

$\surd$ Gradient conjugué, etc.

Choix du pas α_k

Trois approches courantes :

Pas fixe : $\alpha_k = \alpha$,

Recherche linéaire exacte :

$$\alpha_k = \arg\min_{\alpha>0} f(x_k + \alpha d_k),$$

Recherche linéaire inexacte : conditions d'Armijo ou de Wolfe.

Pseudo-code de l'algorithme de descente

Algorithm 1 Algorithme de descente général

Point initial $x_0 \in \mathbb{R}^n$, tolérance $\varepsilon > 0$, max itérations $K_{\max}$ Approximation d'un minimum de f
Initialiser $k \leftarrow 0$

8. **repeat**$\|\nabla f(x_k)\| \leq \varepsilon$ **ou** $k \geq K_{\max}$ Calculer le gradient $\nabla f(x_k)$ Choisir une direction de descente d_k telle que $\nabla f(x_k)^T d_k < 0$ Choisir un pas $\alpha_k > 0$ (par exemple via une recherche linéaire) Mettre à jour : $x_{k+1} \leftarrow x_k + \alpha_k d_k$ $k \leftarrow k+1$ **return** x_k

Critères d'arrêt

L'algorithme s'arrête si l'un des critères suivants est satisfait :

- $\|\nabla f(x_k)\| \leq \varepsilon$
- $\|x_{k+1} - x_k\| \leq \varepsilon$

- $|f(x_{k+1}) - f(x_k)| \leq \varepsilon$
- Nombre maximal d'itérations atteint : $k \geq K_{\max}$

Exercice 4.1. *application de l'algorithme de descente du gradient sur une fonction simple*

(a) Appliquer la méthode de descente du gradient.

(b) Choisir un pas fixe.

(c) Implémenter tout cela dans un algorithme clair.

Problème

On cherche à minimiser la fonction :

$$f(x) = (x-2)^2$$

Son gradient est donné par :

$$\nabla f(x) = 2(x-2)$$

Algorithm 2 Descente du gradient pour $f(x) = (x-2)^2$

Point initial $x_0 = 0$, pas $\alpha = 0.1$, tolérance $\varepsilon = 10^{-4}$, max itérations $K_{\max} = 100$
Approximation d'un minimum de f
Initialiser $k \leftarrow 0$

repeat $|f'(x_k)| \leq \varepsilon$ ou $k \geq K_{\max}$ Calculer le gradient : $g_k \leftarrow 2(x_k - 2)$ Calculer la direction de descente : $d_k \leftarrow -g_k$ Mettre à jour : $x_{k+1} \leftarrow x_k + \alpha \cdot d_k$ $k \leftarrow k+1$
return x_k

4.1.1 Méthode du gradient à pas fixe

On prend $d^{(k)} = -\nabla f(x^{(k)})$, et un pas constant $\alpha_k = \alpha$.

Avantages : simplicité. Inconvénient : lenteur, sensibilité au choix de α.

Principe

On cherche à minimiser une fonction différentiable $f : \mathbb{R}^n \to \mathbb{R}$.

L'algorithme du gradient à pas fixe repose sur la mise à jour suivante :

$$x_{k+1} = x_k - \alpha \nabla f(x_k)$$

où :

— x_k est l'approximation de la solution à l'itération k,

— $\alpha > 0$ est un **pas fixe** (ou *learning rate*),

— $\nabla f(x_k)$ est le gradient de f en x_k.

Algorithme

Algorithm 3 Méthode du gradient à pas fixe

Fonction $f : \mathbb{R}^n \to \mathbb{R}$, gradient ∇f, point initial x_0, pas $\alpha > 0$, tolérance $\varepsilon > 0$, nombre maximal d'itérations $N_{\max}$ $x \leftarrow x_0$ $k \leftarrow 0$ $\|\nabla f(x)\| > \varepsilon$ $k < N_{\max}$ $x \leftarrow x - \alpha \nabla f(x)$ $k \leftarrow k + 1$ x =0

Conditions de convergence

Si f est convexe et de classe $\mathcal{C}^1$ avec un gradient L-Lipschitzien, alors la méthode converge pour :

$$0 < \alpha < \frac{2}{L}$$

Exemple (fonction quadratique)

Considérons :

$$f(x) = \frac{1}{2} x^\top A x - b^\top x$$

avec $A \in \mathbb{R}^{n \times n}$ symétrique définie positive.

Alors :

$$\nabla f(x) = Ax - b$$

Et l'algorithme devient :

$$x_{k+1} = x_k - \alpha(Ax_k - b)$$

4.1.2 Méthode du gradient à pas optimal

Principe de la méthode

Il s'agit d'une méthode de descente où, à chaque itération, on descend dans la direction du gradient négatif, mais le pas α_k est choisi de manière à minimiser exactement la fonction le long de cette direction :

$$\alpha_k = \arg\min_{\alpha > 0} f(x_k - \alpha \nabla f(x_k)).$$

Cette méthode peut être plus efficace que l'utilisation d'un pas fixe ou d'un pas déterminé par une règle de type Armijo ou Wolfe.

Conditions de convergence

Sous hypothèses standard :

- f différentiable avec ∇f lipschitzienne,
- f coercive (tend vers $+\infty$ si $\|x\| \to \infty$),
- α_k choisi selon une règle de type Wolfe ou Armijo,

alors $x^{(k)} \to x^*$, un point stationnaire.

Algorithme

Algorithm 4 Méthode du gradient à pas optimal

Point initial $x_0 \in \mathbb{R}^n$,tolérance $\varepsilon > 0$, $k = 0$ $\|\nabla f(x_k)\| > \varepsilon$ Calcul du gradient : $g_k = \nabla f(x_k)$ Calcul du pas optimal :

$$\alpha_k = \arg\min_{\alpha > 0} f(x_k - \alpha g_k)$$

Mise à jour :

$$x_{k+1} = x_k - \alpha_k g_k$$

$k \leftarrow k + 1$ x_k

Remarque 4.1.

- *Le calcul du pas optimal α_k nécessite une minimisation unidimensionnelle, typiquement réalisée par une méthode de type dichotomie, méthode de la section dorée ou interpolation.*
- *Pour les fonctions quadratiques de la forme $f(x) = \frac{1}{2}x^T Ax - b^T x + c$ avec A symétrique définie positive, le pas optimal admet une formule explicite :*

$$\alpha_k = \frac{g_k^T g_k}{g_k^T A g_k}.$$

Exemple 4.1. *fonction quadratique On souhaite résoudre le problème d'optimisation sans contrainte suivant :*

$$\min_{x \in \mathbb{R}^2} f(x) = (x-1)^2 + 2(y-2)^2$$

La fonction est strictement convexe et admet un unique minimum global en $x^* = (1, 2)$.

Méthode du gradient à pas optimal

La méthode consiste à construire une suite (x_k) selon :

- *Direction de descente : $p_k = -\nabla f(x_k)$*
- *Pas optimal :*

$$\alpha_k = \arg\min_{\alpha>0} f(x_k + \alpha p_k)$$

- *Mise à jour :*

$$x_{k+1} = x_k + \alpha_k p_k$$

*Le pas α_k est déterminé à chaque itération par une **minimisation unidimensionnelle exacte**.*

Fonction choisie

$$f(x, y) = (x-1)^2 + 2(y-2)^2$$

Son gradient est :

$$\nabla f(x, y) = \begin{pmatrix} 2(x-1) \\ 4(y-2) \end{pmatrix}$$

4.1.3 Méthode du Gradient Conjugué

Algorithme : Méthode du gradient conjugué

Données : *matrice symétrique définie positive $A \in \mathbb{R}^{n\times n}$, vecteur $b \in \mathbb{R}^n$, point initial $x_0 \in \mathbb{R}^n$.*

Initialisation :

$$r_0 = b - Ax_0, \quad d_0 = r_0, \quad k = 0.$$

Tant que $r_k \neq 0$ ***faire :***

(a) $\alpha_k = \dfrac{r_k^T r_k}{d_k^T A d_k}$

(b) $x_{k+1} = x_k + \alpha_k d_k$

(c) $r_{k+1} = r_k - \alpha_k A d_k$

(d) $\beta_k = \dfrac{r_{k+1}^T r_{k+1}}{r_k^T r_k}$

(e) $d_{k+1} = r_{k+1} + \beta_k d_k$

(f) $k = k + 1$

Sortie : *approximation x_k de la solution du système $Ax = b$.*

Exercice 4.2. *On considère le système linéaire*

$$Ax = b,$$

où

$$A = \begin{pmatrix} 4 & 1 \\ 1 & 3 \end{pmatrix}, \qquad b = \begin{pmatrix} 1 \\ 2 \end{pmatrix}.$$

(a) Vérifier que la matrice A est symétrique définie positive.

(b) Appliquer la méthode du gradient conjugué avec

$$x_0 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

(c) Calculer x_1 et x_2.

(d) Comparer avec la solution exacte.

Solution

(a) Vérification des hypothèses

La matrice A est symétrique. De plus, pour tout $x \neq 0$,

$$x^T Ax > 0,$$

donc A est définie positive. La méthode du gradient conjugué est applicable.

(b) Initialisation

$$x_0 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \quad r_0 = b - Ax_0 = \begin{pmatrix} 1 \\ 2 \end{pmatrix}, \quad d_0 = r_0.$$

(c) Première itération

$$\alpha_0 = \frac{r_0^T r_0}{d_0^T A d_0}.$$

On calcule :

$$r_0^T r_0 = 5, \quad Ad_0 = \begin{pmatrix} 6 \\ 7 \end{pmatrix}, \quad d_0^T A d_0 = 20.$$

Ainsi,

$$\alpha_0 = \frac{1}{4}.$$

La mise à jour donne :

$$x_1 = x_0 + \alpha_0 d_0 = \begin{pmatrix} \frac{1}{4} \\ \frac{1}{2} \end{pmatrix}.$$

Le nouveau résidu est :

$$r_1 = r_0 - \alpha_0 A d_0 = \begin{pmatrix} -\frac{1}{2} \\ \frac{1}{4} \end{pmatrix}.$$

(d) Deuxième itération

$$\beta_0 = \frac{r_1^T r_1}{r_0^T r_0} = \frac{5/16}{5} = \frac{1}{16}.$$

La direction conjuguée est :

$$d_1 = r_1 + \beta_0 d_0 = \begin{pmatrix} -\frac{7}{16} \\ \frac{3}{8} \end{pmatrix}.$$

On calcule :

$$\alpha_1 = \frac{r_1^T r_1}{d_1^T A d_1} = \frac{4}{11}.$$

Ainsi,

$$x_2 = x_1 + \alpha_1 d_1 = \begin{pmatrix} \frac{1}{11} \\ \frac{7}{11} \end{pmatrix}.$$

(e) Conclusion

La solution exacte du système $Ax = b$ est

$$x = \begin{pmatrix} \frac{1}{11} \\ \frac{7}{11} \end{pmatrix}.$$

La méthode du gradient conjugué converge en au plus $n = 2$ itérations.

4.1.4 Méthode de Newton

Utilise la Hessienne :

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

Avantages : *convergence quadratique près du minimum (si Hessienne définie positive)* ***Inconvénients :*** *coûteuse en calcul ($O(n^3)$), nécessite $\nabla^2 f$*

Principe de la méthode de Newton

La méthode de Newton repose sur l'approximation quadratique locale de la fonction f autour du point courant x_k :

$$f(x) \approx f(x_k) + \nabla f(x_k)^T (x - x_k) + \frac{1}{2}(x - x_k)^T \nabla^2 f(x_k)(x - x_k).$$

En minimisant cette approximation, on obtient l'itération de Newton :

$$x_{k+1} = x_k - H_k^{-1} \nabla f(x_k),$$

où $H_k = \nabla^2 f(x_k)$ est la matrice Hessienne de f au point x_k.

Algorithme

Algorithme : Méthode de Newton

On cherche à résoudre le problème d'optimisation

$$\min_{x \in \mathbb{R}^n} f(x), \quad f \in C^2(\mathbb{R}^n).$$

Données : *point initial $x_0 \in \mathbb{R}^n$, tolérance $\varepsilon > 0$, nombre maximal d'itérations $k_{\max}$.*

Initialisation : $k = 0$.

Tant que $\|\nabla f(x_k)\| > \varepsilon$ *et* $k < k_{\max}$ ***faire :***

(a) Calculer le gradient $g_k = \nabla f(x_k)$

(b) Calculer la Hessienne $H_k = \nabla^2 f(x_k)$

(c) Résoudre le système linéaire

$$H_k d_k = -g_k$$

(d) Mettre à jour

$$x_{k+1} = x_k + d_k$$

(e) $k = k + 1$

Sortie : *approximation x_k d'un point stationnaire de f.*

Remarque 4.2. *La méthode de Newton converge **quadratiquement** si f est strictement convexe et que le point initial est suffisamment proche du minimum.*

Si la Hessienne n'est pas définie positive, la méthode peut ne pas converger vers un minimum. On utilise alors des variantes :

- ***Méthode de Newton modifiée** : rendre la Hessienne définie positive.*
- ***Méthodes quasi-Newton** : comme BFGS ou SR1.*
- ***Méthode de Newton avec ligne de recherche** : utiliser un pas $\alpha_k \in (0, 1]$ choisi par une méthode de type Armijo.*

Complexité par itération : résolution d'un système linéaire (coût $\mathcal{O}(n^3)$ pour la factorisation de Cholesky ou LU).

Exemple 4.2. *Méthode de Newton avec pas α_k*

Problème

On cherche à résoudre :

$$\min_{x \in \mathbb{R}^n} f(x),$$

où $f : \mathbb{R}^n \to \mathbb{R}$ est deux fois continûment différentiable.

Principe de la méthode

La méthode de Newton repose sur l'approximation quadratique de f autour de x_k, et calcule la direction de descente :

$$p_k = -H_k^{-1} \nabla f(x_k),$$

où $H_k = \nabla^2 f(x_k)$ est la Hessienne.

*Mais pour garantir la convergence (même loin du minimum), on introduit un **pas $\alpha_k \in (0, 1]$** trouvé par **ligne de recherche** :*

$$x_{k+1} = x_k + \alpha_k p_k.$$

Choix du pas α_k

*On choisit α_k tel que la condition d'**Armijo** soit satisfaite :*

$$f(x_k + \alpha_k p_k) \leq f(x_k) + c\alpha_k \nabla f(x_k)^T p_k,$$

où $c \in (0, 1)$, typiquement $c = 10^{-4}$.

*Le pas est généralement trouvé par **recherche en arrière (backtracking)**, en réduisant le pas jusqu'à satisfaction de cette inégalité.*

Algorithme

Algorithm 5 Méthode de Newton avec ligne de recherche (Armijo)

Point initial $x_0 \in \mathbb{R}^n$, tolérance $\varepsilon > 0$, $c \in (0,1)$, facteur $\rho \in (0,1)$ $k \leftarrow 0$ $\|\nabla f(x_k)\| > \varepsilon$ $g_k = \nabla f(x_k)$ $H_k = \nabla^2 f(x_k)$ Résoudre $H_k p_k = -g_k$ Initialiser $\alpha_k = 1$ $f(x_k + \alpha_k p_k) > f(x_k) + c\alpha_k g_k^T p_k$ $\alpha_k \leftarrow \rho\alpha_k$ (recherche en arrière) $x_{k+1} = x_k + \alpha_k p_k$ $k \leftarrow k+1$ x_k

Remarque 4.3. — *Le pas α_k est déterminé dynamiquement pour assurer une diminution suffisante de la fonction.*

— *Cela permet à la méthode de converger même si l'approximation quadratique n'est pas fiable loin du minimum.*

— *Lorsque x_k est proche du minimum, on retrouve une convergence quadratique.*

— *La condition d'Armijo peut être couplée à la condition de courbure (condition de Wolfe) pour un contrôle plus fin.*

Remarque 4.4. *Remarques pédagogiques :*

- *Ligne de recherche = sécurité : évite que la méthode diverge.*
- *Backtracking (recherche en arrière) est facile à implémenter et efficace.*
- *Condition d'Armijo : garantit que la fonction diminue suffisamment, mais sans gaspiller trop d'itérations.*

Exemple 4.3. *Exemple concret : Méthode de Newton avec ligne de recherche*

Fonction à minimiser

On considère la fonction :

$$f(x,y) = (x-1)^2 + 2(y-2)^2,$$

qui est strictement convexe.

Gradient

$$\nabla f(x,y) = \begin{pmatrix} 2(x-1) \\ 4(y-2) \end{pmatrix}$$

Hessienne

$$H = \nabla^2 f(x,y) = \begin{pmatrix} 2 & 0 \\ 0 & 4 \end{pmatrix}$$

Application de la méthode de Newton avec ligne de recherche

(a) Point initial

$$x_0 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

(b) Itération 0

— *Gradient :* $g_0 = \begin{pmatrix} -2 \\ -8 \end{pmatrix}$

— *Hessienne* H *constante*

— *Direction de Newton :*

$$p_0 = -H^{-1} g_0 = -\begin{pmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{4} \end{pmatrix} \begin{pmatrix} -2 \\ -8 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

— *Test de la condition d'Armijo :*

$$f(x_0 + \alpha p_0) \leq f(x_0) + c\alpha \nabla f(x_0)^T p_0$$

Avec $\alpha = 1$, $c = 10^{-4}$, *on calcule :*

$$f(x_0) = (0-1)^2 + 2(0-2)^2 = 1 + 8 = 9$$

$$f(x_1) = f(1,2) = 0$$

$$\nabla f(x_0)^T p_0 = (-2)(1) + (-8)(2) = -2 - 16 = -18$$

$$f(x_1) = 0 \leq 9 - 0.0018 = 8.9982$$

Donc $\alpha_0 = 1$ *est accepté.*

— *Mise à jour :*

$$x_1 = x_0 + \alpha_0 p_0 = (0,0) + (1,2) = (1,2)$$

— *Gradient au point suivant :* $\nabla f(x_1) = 0$

(c) Conclusion

Le minimum est atteint en une seule itération :

$$x^* = (1,2), \quad f(x^*) = 0$$

(d) Remarque

• *Ici, la Hessienne étant constante et la fonction étant quadratique, la méthode de Newton converge en une seule étape.*

• *Fonction quadratique* $\Rightarrow$ *convergence en une étape.*

- *Ligne de recherche avec Armijo vérifiée automatiquement ici avec* $\alpha = 1$.

- *Idéal pour montrer comment fonctionne Newton avec ligne de recherche dans un cas simple mais rigoureux.*

Code Matlab

```
function newton_armijo()

    % Param tres
    c = 1e-4;
    rho = 0.5;
    tol = 1e-6;
    max_iter = 100;

    % Point initial
    x = [0; 0];

    for k = 1:max_iter
        g = gradient_f(x);
        H = hessian_f(x);

        if norm(g) < tol
            fprintf('Convergence atteinte apr s %d it rations.\n'
                , k-1);
            break;
        end

        p = -H \ g;

        % Ligne de recherche
        alpha = 1;
        f_x = f(x);
        while f(x + alpha * p) > f_x + c * alpha * g' * p
            alpha = rho * alpha;
        end

        x = x + alpha * p;

        fprintf('It ration %d : x = [%f, %f], f(x) = %f, alpha = %
            f\n', ...
                k, x(1), x(2), f(x), alpha);
    end

    fprintf('Solution finale : x* = [%f, %f], f(x*) = %f\n', x(1),
```

```
        x(2), f(x));
end

function val = f(x)
    val = (x(1) - 1)^2 + 2*(x(2) - 2)^2;
end

function g = gradient_f(x)
    g = [2*(x(1) - 1); 4*(x(2) - 2)];
end

function H = hessian_f(~)
    H = [2, 0; 0, 4];
end
```

Listing 4.1 – Méthode de Newton avec Armijo

Résultats attendus

Ce programme renvoie la sortie suivante :

```
Itération 1 : x = [1.000000, 2.000000], f(x) = 0.000000, alpha = 1.000000
Convergence atteinte après 1 itérations.
Solution finale : x* = [1.000000, 2.000000], f(x*) = 0.000000
```

Remarque 4.5. *Commentaires*

- *La fonction est quadratique avec Hessienne constante : convergence en une seule itération.*
- *Le pas $\alpha = 1$ est suffisant pour satisfaire la condition d'Armijo.*
- *Si la Hessienne variait ou si la fonction était non quadratique, plusieurs itérations et ajustements du pas seraient nécessaires.*

4.1.5 Méthodes quasi-Newton (ex : BFGS)

La méthode BFGS (Broyden–Fletcher–Goldfarb–Shanno) est une méthode quasi-Newton destinée à la résolution de problèmes d'optimisation sans contraintes :

$$\min_{x \in \mathbb{R}^n} f(x), \quad f \in C^1(\mathbb{R}^n).$$

Elle constitue une alternative à la méthode de Newton en évitant le calcul explicite de la matrice Hessienne. L'idée principale consiste à construire, à chaque itération, une

approximation H_k de l'inverse de la Hessienne $\nabla^2 f(x_k)^{-1}$ à partir des informations de gradient.

But : approximer la Hessienne sans la calculer.

$$x^{(k+1)} = x^{(k)} - H_k \nabla f(x^{(k)}), \quad H_k \approx [\nabla^2 f(x^{(k)})]^{-1}$$

Ex : BFGS met à jour H_k via des différences finies sur les gradients.

Avantages : *rapide, sans calcul de Hessienne* ***Inconvénients :*** *nécessite du stockage ($O(n^2)$)*

Méthodes quasi-Newton (exemple : BFGS)

On cherche à résoudre le problème d'optimisation

$$\min_{x\in\mathbb{R}^n} f(x), \quad f \in C^1(\mathbb{R}^n).$$

Les méthodes quasi-Newton construisent une approximation H_k de la matrice inverse de la Hessienne $\nabla^2 f(x_k)$, sans la calculer explicitement.

Données : *point initial $x_0 \in \mathbb{R}^n$, tolérance $\varepsilon > 0$, nombre maximal d'itérations $k_{\max}$.*

Initialisation :

$$H_0 = I_n, \quad k = 0.$$

Tant que $\|\nabla f(x_k)\| > \varepsilon$ *et* $k < k_{\max}$ ***faire :***

(a) Calculer le gradient

$$g_k = \nabla f(x_k).$$

(b) Calculer la direction de descente

$$d_k = -H_k g_k.$$

(c) Choisir un pas $\alpha_k > 0$ (recherche linéaire).

(d) Mettre à jour

$$x_{k+1} = x_k + \alpha_k d_k.$$

(e) Poser

$$s_k = x_{k+1} - x_k, \qquad y_k = g_{k+1} - g_k.$$

(f) Mettre à jour la matrice H_k (formule BFGS) :

$$H_{k+1} = \left(I - \frac{s_k y_k^T}{y_k^T s_k}\right) H_k \left(I - \frac{y_k s_k^T}{y_k^T s_k}\right) + \frac{s_k s_k^T}{y_k^T s_k}.$$

(g) $k = k + 1$

Sortie : *approximation x_k d'un point stationnaire de f.*

4.1.6 Gradient conjugué (fonctions quadratiques)

Supposons $f(x) = \frac{1}{2}x^T Ax - b^T x$, avec A symétrique définie positive.

L'algorithme génère une suite de directions conjuguées $d^{(k)}$, et converge en $\leq n$ étapes (exact pour quadratique).

$$x^{(k+1)} = x^{(k)} + \alpha_k d^{(k)}$$

Avec :

$$d^{(k+1)} = -\nabla f(x^{(k+1)}) + \beta_k d^{(k)}$$

Avantages : *très efficace pour grands problèmes linéaires* ***Inconvénients :*** *nécessite symétrie et définie positive*

4.1.7 Méthodes de relaxation

Algorithme : Méthode de Relaxation (SOR)

*La méthode de relaxation, aussi appelée SOR (*Successive Over-Relaxation*), est une méthode itérative pour résoudre un système linéaire*

$$Ax = b, \quad A \in \mathbb{R}^{n \times n}.$$

*Elle généralise la méthode de Gauss-Seidel en introduisant un **paramètre de relaxation** ω.*

Données : *matrice A, vecteur b, point initial $x^{(0)}$, tolérance ε, nombre maximal d'itérations $k_{\max}$, paramètre $\omega \in (0, 2)$.*

Initialisation : *$k = 0$, $x^{(0)}$ donné.*

Tant que *$\|x^{(k+1)} - x^{(k)}\| > \varepsilon$ et $k < k_{\max}$* ***faire :***

(a) Pour $i = 1, 2, \ldots, n$:

$$x_i^{(k+1)} = (1 - \omega) x_i^{(k)} + \frac{\omega}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right)$$

(b) $k = k + 1$

Sortie : *$x^{(k+1)}$ approximation de la solution de $Ax = b$.*

Remarques :

- *$\omega = 1$ correspond à la méthode de Gauss-Seidel classique.*
- *Pour $0 < \omega < 1$, on parle de* under-relaxation *(ralentissement).*
- *Pour $1 < \omega < 2$, on parle de* over-relaxation *(accélération).*
- *La convergence est garantie si A est symétrique définie positive ou strictement diagonale dominante.*

4.1.8 Algorithmes probabilistes

Descente aléatoire (random search)

À chaque étape, on propose un nouveau point aléatoire dans un voisinage :

$$x^{(k+1)} = x^{(k)} + \epsilon_k \cdot v^{(k)}, \quad v^{(k)} \sim \mathcal{U}(voisinage)$$

Si $f(x^{(k+1)}) < f(x^{(k)})$, on accepte.

Avantages : *simple, applicable à fonctions non différentiables* ***Inconvénients :*** *convergence lente*

Recuit simulé (Simulated Annealing)

Inspiré du refroidissement de métaux. On accepte parfois des solutions moins bonnes pour échapper aux minima locaux :

$$P(\textit{accepter mauvaise solution}) = \exp\left(-\frac{\Delta f}{T_k}\right)$$

Où T_k est une "température" qui décroît au cours des itérations.

Avantages : *exploration globale, robuste aux minima locaux* ***Inconvénients :*** *lent, dépend des paramètres de refroidissement*

4.2 Comparaison synthétique

Méthode	Convergence	Complexité	Remarques
Gradient fixe	linéaire lente	faible	dépend du pas
Gradient optimal	linéaire +	moyenne	nécessite line search
Newton	quadratique	élevée	besoin de la Hessienne
Quasi-Newton	superlinéaire	moyenne	bonne efficacité
Gradient conjugué	exacte (quad.)	faible	problème quadratique
Recuit simulé	lente	moyenne/haute	robuste, aléatoire

TABLE 4.1 – Tableau récapitulatif des méthodes sans contraintes

Chapitre 5

Méthodes avec contraintes

Dans de nombreux problèmes réels d'optimisation, les variables ne peuvent pas varier librement et doivent satisfaire un ensemble de contraintes traduisant des limitations physiques, économiques ou techniques. Les méthodes d'optimisation avec contraintes visent à déterminer les solutions optimales tout en respectant ces restrictions, qu'elles soient des contraintes d'égalité ou d'inégalité.

L'introduction des contraintes modifie profondément la nature du problème et nécessite des outils théoriques et algorithmiques spécifiques. Parmi ceux-ci figurent les conditions d'optimalité, telles que les conditions de Karush–Kuhn–Tucker, ainsi que des méthodes numériques adaptées comme les multiplicateurs de Lagrange, les méthodes de pénalisation ou les méthodes de projection. Ce chapitre présente les principes fondamentaux de l'optimisation sous contraintes, les principales approches de résolution et les hypothèses garantissant l'existence et la convergence des solutions.

5.1 Multiplicateurs de Lagrange

Problème : *Minimiser une fonction $f(x)$ sous contraintes d'égalité $h_i(x) = 0$, $i = 1, \ldots, m$.*

Lagrangien :

$$\mathcal{L}(x, \lambda) = f(x) + \sum_{i=1}^{m} \lambda_i h_i(x)$$

Conditions nécessaires (si x^* est optimal) :

$$\nabla_x \mathcal{L}(x^*, \lambda^*) = 0 \quad et \quad h_i(x^*) = 0 \quad \forall i$$

Théorème des multiplicateurs de Lagrange

Soit le problème :

$$\min_{x \in \mathbb{R}^n} f(x) \quad \textit{sous les contraintes} \quad h_i(x) = 0, \quad i = 1, \ldots, m$$

où $f : \mathbb{R}^n \to \mathbb{R}$ *et* $h_i : \mathbb{R}^n \to \mathbb{R}$ *sont de classe* $\mathcal{C}^1$.

Theorem 5.1.1: Théorème de Lagrange

Si x^* *est une solution locale, et si les gradients des contraintes* $\nabla h_1(x^*), \ldots, \nabla h_m(x^*)$ *sont linéairement indépendants, alors il existe* $\lambda_1, \ldots, \lambda_m \in \mathbb{R}$ *tels que :*

$$\nabla f(x^*) + \sum_{i=1}^{m} \lambda_i \nabla h_i(x^*) = 0$$

ou, en notation matricielle :

$$\nabla f(x^*) + J_h(x^*)^T \lambda = 0$$

où $J_h(x^*)$ *est la jacobienne des contraintes, et* $\lambda = (\lambda_1, \ldots, \lambda_m)^T$.

5.2 Théorème de Karush-Kuhn-Tucker (KKT)

Considérons le problème d'optimisation non linéaire suivant :

$$\begin{aligned} \textit{Minimiser} \quad & f(x) \\ \textit{sous les contraintes} \quad & g_i(x) \leq 0, \quad i = 1, \ldots, m \\ & h_j(x) = 0, \quad j = 1, \ldots, p \end{aligned}$$

où $f : \mathbb{R}^n \to \mathbb{R}$, $g_i : \mathbb{R}^n \to \mathbb{R}$, *et* $h_j : \mathbb{R}^n \to \mathbb{R}$ *sont toutes des fonctions continûment différentiables.*

Theorem 5.2.1: (Conditions KKT)

Si x^* est un point optimal local, et que certaines conditions de régularité (comme celle de Slater) sont satisfaites, alors il existe des multiplicateurs de Lagrange $\lambda_i \geq 0$ pour $i = 1, \ldots, m$ et $\mu_j \in \mathbb{R}$ pour $j = 1, \ldots, p$, tels que les conditions suivantes soient vérifiées :

(a) ***Stationnarité*** *:*

$$\nabla f(x^*) + \sum_{i=1}^{m} \lambda_i \nabla g_i(x^*) + \sum_{j=1}^{p} \mu_j \nabla h_j(x^*) = 0$$

(b) ***Primalité (feasibility)*** *:*

$$g_i(x^*) \leq 0, \quad \forall i = 1, \ldots, m$$

$$h_j(x^*) = 0, \quad \forall j = 1, \ldots, p$$

(c) ***Dualité (dual feasibility)*** *:*

$$\lambda_i \geq 0, \quad \forall i = 1, \ldots, m$$

(d) ***Complémentarité*** *:*

$$\lambda_i \cdot g_i(x^*) = 0, \quad \forall i = 1, \ldots, m$$

Remarque 5.1. *Pour que les conditions KKT soient* nécessaires, *une condition de qualification des contraintes doit être satisfaite. Par exemple, la* condition de Slater *: il existe un point strictement admissible x tel que :*

$$g_i(x) < 0 \quad et \quad h_j(x) = 0$$

alors les conditions KKT sont nécessaires pour un minimum local.

Algorithm 6 Méthode de Lagrange (pour contraintes d'égalité)

1: **Entrée :** Fonction objectif $f(x)$, contraintes $g_i(x) = 0$, $i = 1, \ldots, m$
2: **Sortie :** Point stationnaire x^*, multiplicateurs de Lagrange λ^*

3: Former le Lagrangien :

$$\mathcal{L}(x, \lambda) = f(x) + \sum_{i=1}^{m} \lambda_i g_i(x)$$

4: Calculer le gradient de $\mathcal{L}$:

$$\nabla_x \mathcal{L}(x, \lambda) = \nabla f(x) + \sum_{i=1}^{m} \lambda_i \nabla g_i(x)$$

5: Écrire le système :

$$\begin{cases} \nabla_x \mathcal{L}(x, \lambda) = 0 \\ g_i(x) = 0 \quad \text{pour } i = 1, \ldots, m \end{cases}$$

6: Résoudre ce système non linéaire pour obtenir (x^*, λ^*)
7: Vérifier si x^* est un minimum (Hessienne ou convexité de f)
8: **return** (x^*, λ^*)

Exercice 5.1. *Algorithme de Lagrange*

Minimiser $f(x, y) = x^2 + y^2$

Sous la contrainte $g(x, y) = x + y - 1 = 0$

Exercice 5.2. *Conditions KKT (contrainte d'inégalité)*

Minimiser $f(x, y) = x^2 + y^2$

Sous la contrainte $h(x, y) = x + y - 1 \leq 0$

Algorithm 7 Résolution avec multiplicateurs de Lagrange

1: **Définir** $f(x, y) = x^2 + y^2$ et $g(x, y) = x + y - 1$
2: **Former** le Lagrangien :

$$\mathcal{L}(x, y, \lambda) = f(x, y) + \lambda g(x, y) = x^2 + y^2 + \lambda(x + y - 1)$$

3: **Calculer les dérivées partielles** :

$$\frac{\partial \mathcal{L}}{\partial x} = 2x + \lambda, \quad \frac{\partial \mathcal{L}}{\partial y} = 2y + \lambda, \quad \frac{\partial \mathcal{L}}{\partial \lambda} = x + y - 1$$

4: **Résoudre le système** :

$$\begin{aligned} 2x + \lambda &= 0 \\ 2y + \lambda &= 0 \\ x + y &= 1 \end{aligned}$$

5: **Déduire** : $x = y = \frac{1}{2}$, $\lambda = -1$
6: **Calculer le minimum** : $f\left(\frac{1}{2}, \frac{1}{2}\right) = \frac{1}{2}$
7: **return** $x^* = \frac{1}{2}$, $y^* = \frac{1}{2}$, $\lambda^* = -1$, $f(x^*, y^*) = \frac{1}{2}$

Algorithm 8 Méthode KKT pour contrainte d'inégalité

1: **Former** le Lagrangien :

$$\mathcal{L}(x, y, \mu) = x^2 + y^2 + \mu(x + y - 1)$$

2: **Écrire les conditions KKT** :

$$\begin{aligned} 2x + \mu &= 0 \\ 2y + \mu &= 0 \\ x + y - 1 &\leq 0 \\ \mu &\geq 0 \\ \mu(x + y - 1) &= 0 \end{aligned}$$

3: **Cas 1 :** $\mu = 0 \Rightarrow x = y = 0$
4: Vérifier : $x + y - 1 = -1 \leq 0 \Rightarrow$ admissible
5: **Cas 2 :** $x + y - 1 = 0 \Rightarrow x = y = \frac{1}{2}$
6: $\mu = -1 \Rightarrow$ rejeté car $\mu < 0$
7: **return** $x^* = 0$, $y^* = 0$, $\mu^* = 0$, $f^* = 0$

5.3 Méthodes primales-duales

5.3.1 Méthode d'Uzawa (pour contraintes d'égalité)

Méthode d'Uzawa : explication

*La **méthode d'Uzawa** est une méthode itérative pour résoudre des problèmes d'optimisation avec contraintes d'égalité :*

$$\min f(x) \quad sous\ h(x) = 0,$$

ou dans sa forme quadratique linéaire :

$$\min_x \frac{1}{2} x^T A x - b^T x \quad sous\ Cx = d.$$

Principe :

*(a) On construit le **Lagrangien** :*

$$\mathcal{L}(x, \lambda) = f(x) - \lambda^T (Cx - d)$$

(b) On alterne entre :

— ***Minimisation par rapport à** x :*

$$x^{(k+1)} = \arg\min_x \mathcal{L}(x, \lambda^{(k)}) \quad \Rightarrow \quad Ax^{(k+1)} = b - C^T \lambda^{(k)}$$

— ***Mise à jour du multiplicateur de Lagrange** :*

$$\lambda^{(k+1)} = \lambda^{(k)} + \rho(Cx^{(k+1)} - d)$$

avec $\rho > 0$ le pas de descente pour le dual.

(c) Répéter jusqu'à ce que la contrainte soit satisfaite :

$$\|Cx^{(k+1)} - d\| < \varepsilon$$

Remarques :

— *Méthode adaptée aux problèmes quadratiques avec contraintes linéaires.*

— *ρ doit être choisi petit pour garantir la convergence.*

— *C'est une méthode de type **dual** : on ajuste x pour minimiser le Lagrangien et λ pour réduire le viol de contrainte.*

— *L'intuition : alternance entre la mise à jour **primal** (x) et **dual** (λ), jusqu'à satisfaction des contraintes.*

Algorithme : Méthode d'Uzawa

*La méthode d'Uzawa est une méthode itérative pour résoudre un problème d'optimisation avec **contraintes d'égalité** :*

$$\min_{x \in \mathbb{R}^n} f(x) \quad \textit{sous } h(x) = 0, \quad h(x) \in \mathbb{R}^m$$

ou, dans sa forme linéaire classique :

$$\min_x \frac{1}{2} x^T A x - b^T x \quad \textit{sous } Cx = d$$

—

Principe : *itérer sur le **multiplicateur de Lagrange** λ et mettre à jour x via un **gradient sur le Lagrangien**.*

—

Données : *- Matrice $A \in \mathbb{R}^{n \times n}$, vecteurs $b \in \mathbb{R}^n$, $d \in \mathbb{R}^m$ - Matrice de contraintes $C \in \mathbb{R}^{m \times n}$ - Pas de descente $\rho > 0$, tolérance ε, itérations maximales $k_{\max}$*

Algorithme :

(a) Initialiser $\lambda^{(0)} = 0$

(b) Pour $k = 0, 1, 2, \ldots, k_{\max}$:

i. Résoudre le système :

$$Ax^{(k+1)} = b - C^T \lambda^{(k)}$$

ii. Mettre à jour le multiplicateur de Lagrange :

$$\lambda^{(k+1)} = \lambda^{(k)} + \rho(Cx^{(k+1)} - d)$$

iii. Vérifier le critère d'arrêt : $\|Cx^{(k+1)} - d\| < \varepsilon$

(c) Retourner $x^{(k+1)}, \lambda^{(k+1)}$

Remarques :

- *ρ doit être choisi suffisamment petit pour garantir la convergence.*
- *Méthode particulièrement adaptée pour les problèmes quadratiques et contraintes linéaires.*
- *C'est une méthode **itérative duale**, souvent utilisée pour la programmation quadratique.*

Exemple numérique : Méthode d'Uzawa

On considère le problème quadratique avec contrainte linéaire :

$$\min_{x\in\mathbb{R}^2} f(x) = \frac{1}{2}x^T Ax - b^T x \quad \textit{sous } Cx = d$$

avec

$$A = \begin{pmatrix} 4 & 1 \\ 1 & 3 \end{pmatrix}, \quad b = \begin{pmatrix} 1 \\ 2 \end{pmatrix}, \quad C = \begin{pmatrix} 1 & 1 \end{pmatrix}, \quad d = 1.$$

Paramètres : $\rho = 0.1$, *tolérance* $\varepsilon = 10^{-6}$, $\lambda^{(0)} = 0$.

Itération 0 :

$$Ax^{(1)} = b - C^T\lambda^{(0)} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

Résolution :

$$\begin{pmatrix} 4 & 1 \\ 1 & 3 \end{pmatrix}\begin{pmatrix} x_1^{(1)} \\ x_2^{(1)} \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \end{pmatrix} \implies x^{(1)} = \begin{pmatrix} 0.2 \\ 0.6 \end{pmatrix}$$

Mise à jour du multiplicateur :

$$\lambda^{(1)} = \lambda^{(0)} + \rho(Cx^{(1)} - d) = 0 + 0.1((0.2 + 0.6) - 1) = -0.02$$

Itération 1 :

$$Ax^{(2)} = b - C^T\lambda^{(1)} = \begin{pmatrix} 1.02 \\ 2.02 \end{pmatrix}$$

Résolution :

$$x^{(2)} = \begin{pmatrix} 0.205 \\ 0.605 \end{pmatrix}$$

Mise à jour du multiplicateur :

$$\lambda^{(2)} = \lambda^{(1)} + \rho(Cx^{(2)} - d) = -0.02 + 0.1((0.205 + 0.605) - 1) = -0.014$$

Itérations suivantes : *Le processus se répète jusqu'à ce que :*

$$|Cx^{(k)} - d| < \varepsilon$$

Après quelques itérations, on obtient la solution approximative :

$$x^* \approx \begin{pmatrix} 0.25 \\ 0.75 \end{pmatrix}, \quad \lambda^* \approx 0.25$$

Conclusion : *- La méthode d'Uzawa a permis de trouver le minimum du problème quadratique avec contrainte linéaire. - La convergence dépend du paramètre ρ et de la matrice A. - On obtient le minimum de $f(x)$ sous $x_1 + x_2 = 1$ en itérations successives sur x et λ.*

√ *Remarques : La méthode est simple mais converge lentement sauf pour des problèmes très bien conditionnés (convexes, linéaires, ...).*

Méthode simple, adaptée aux problèmes convexes. Elle peut être vue comme un gradient ascent sur le dual, avec minimisation exacte sur le primal à chaque itération.

```
function [x, lambda] = uzawa(Q, c, A, b, rho, tol, max_iter)
    if nargin < 7, max_iter = 1000; end
    if nargin < 6, tol = 1e-6; end
    if nargin < 5, rho = 0.1; end

    n = size(Q,1);
    m = size(A,1);

    x = zeros(n,1);
    lambda = zeros(m,1);
    Qinv = inv(Q);  % Pr -calcul de l'inverse

    for k = 1:max_iter
        %  tape  1 : minimisation du Lagrangien
        x = -Qinv * (c + A' * lambda);

        %  tape  2 : mise    jour du multiplicateur
        r = A * x - b;
        lambda = lambda + rho * r;

        if norm(r) < tol
            break;
        end
    end
end

% Exemple d'utilisation :
Q = [2 0; 0 2];
c = [-2; -5];
A = [1 1];
b = 1;

[x_star, lambda_star] = uzawa(Q, c, A, b);
disp("Solution optimale x* = "), disp(x_star)
disp("Multiplicateur lambda* = "), disp(lambda_star)
```

Listing 5.1 – Méthode d'Uzawa en MATLAB

5.3.2 Lagrangien Augmenté

Lagrangien Augmenté pour des contraintes d'égalité

On considère le problème d'optimisation sous contraintes d'égalité suivant :

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ s.c. \quad & h_i(x) = 0, \quad i = 1, \dots, m \end{aligned}$$

où :

- $f(x)$ *est la fonction objectif à minimiser,*
- $h(x) = (h_1(x), \dots, h_m(x))^\top$ *est le vecteur des contraintes d'égalité.*

(a) *Principe de la méthode*

La méthode du ***Lagrangien augmenté*** *combine :*

- *la méthode classique des multiplicateurs de Lagrange,*
- *la méthode de pénalisation quadratique.*

(b) *Le Lagrangien augmenté*

Le Lagrangien augmenté s'écrit :

$$\mathcal{L}_A(x, \lambda; \mu) = f(x) + \lambda^\top h(x) + \frac{\mu}{2} \|h(x)\|^2$$

où :

- $\lambda \in \mathbb{R}^m$ *est le vecteur des multiplicateurs de Lagrange,*
- $\mu > 0$ *est un paramètre de pénalisation.*

(c) *Algorithme de la méthode*

Entrées :

- *Fonction* $f(x)$*, contraintes* $h(x)$*,*
- *Initialisation :* x^0*,* λ^0*,* $\mu > 0$*,*
- *Tolérance* $\varepsilon > 0$*.*

Étapes de l'algorithme :

i. ***Initialisation*** *: Choisir* x^0*,* λ^0*,* $\mu > 0$*,* $k = 0$

ii. ***Minimisation du Lagrangien augmenté :***

$$x^{k+1} = \arg\min_x \left[f(x) + (\lambda^k)^\top h(x) + \frac{\mu}{2} \|h(x)\|^2 \right]$$

iii. ***Mise à jour des multiplicateurs de Lagrange :***

$$\lambda^{k+1} = \lambda^k + \mu h(x^{k+1})$$

iv. ***Critère d'arrêt :*** *arrêter si*

$$\|h(x^{k+1})\| \leq \varepsilon \quad et \quad \|\nabla f(x^{k+1}) + \nabla h(x^{k+1})^\top \lambda^{k+1}\| \leq \varepsilon$$

v. ***Sinon :*** *poser $k \leftarrow k+1$, retourner à l'étape 2.*

(d) *Remarques*

- *Le paramètre μ peut rester fixe ou être augmenté à chaque itération.*
- *L'étape de minimisation peut être effectuée par une méthode numérique (gradient, Newton, etc.).*
- *L'approche est plus stable numériquement que la méthode de pénalisation seule.*

(e) *Extensions*

La méthode peut être étendue à des contraintes d'inégalités $g_i(x) \leq 0$, mais cela nécessite des adaptations supplémentaires (méthode du Lagrangien augmenté généralisé).

Lagrangien Augmenté pour des contraintes d'inégalité

Lorsqu'on a des contraintes d'inégalité, la méthode du Lagrangien augmenté peut toujours être utilisée, mais elle doit être adaptée, car les multiplicateurs de Lagrange associés aux contraintes d'inégalité doivent respecter une condition de non-négativité (contrairement aux contraintes d'égalité)
On considère le problème d'optimisation suivant :

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ s.c. \quad & g_i(x) \leq 0, \quad i = 1, \ldots, m \end{aligned}$$

où :

- *$f(x)$ est une fonction objectif différentiable,*
- *$g_i(x)$ sont les fonctions de contraintes (également différentiables).*

Lagrangien Augmenté

Algorithm 9 Méthode du Lagrangien Augmenté (pour contraintes d'égalité)

Entrée : fonction objectif $f(x)$, contraintes $h(x) = 0$
Paramètres : $\mu > 0$, tolérance $\varepsilon > 0$
Initialisation : choisir $x^0 \in \mathbb{R}^n$, $\lambda^0 \in \mathbb{R}^m$, $k \leftarrow 0$
while critère d'arrêt non satisfait **do**
    Minimiser le Lagrangien augmenté :

$$x^{k+1} \leftarrow \arg\min_x \left[f(x) + (\lambda^k)^\top h(x) + \frac{\mu}{2} \|h(x)\|^2 \right]$$

    Mettre à jour les multiplicateurs de Lagrange :

$$\lambda^{k+1} \leftarrow \lambda^k + \mu h(x^{k+1})$$

    Vérifier les conditions d'arrêt :
    if $\|h(x^{k+1})\| \leq \varepsilon$ **et** $\|\nabla f(x^{k+1}) + \nabla h(x^{k+1})^\top \lambda^{k+1}\| \leq \varepsilon$ **then**
        Arrêter
    else
        $k \leftarrow k + 1$
    end if
end while
Sortie : solution x^{k+1}, multiplicateurs λ^{k+1}

Le Lagrangien augmenté associé à ce problème est donné par :

$$\mathcal{L}_A(x, \lambda; \mu) = f(x) + \sum_{i=1}^{m} \lambda_i g_i(x) + \frac{\mu}{2} \sum_{i=1}^{m} [\max(0, g_i(x))]^2$$

avec :

— $\lambda_i \geq 0$ *: multiplicateurs de Lagrange,*

— $\mu > 0$ *: paramètre de pénalisation.*

Conditions KKT (Karush-Kuhn-Tucker)

Les conditions nécessaires d'optimalité (KKT) sont :

$$\begin{cases} \nabla f(x^*) + \sum_{i=1}^{m} \lambda_i \nabla g_i(x^*) = 0 \\ g_i(x^*) \leq 0, \quad \lambda_i \geq 0 \\ \lambda_i \cdot g_i(x^*) = 0, \quad \forall i \end{cases}$$

Algorithme du Lagrangien Augmenté (contraintes d'inégalité)

Algorithm 10 Méthode du Lagrangien Augmenté avec contraintes d'inégalité

1: **Entrée :** fonction $f(x)$, contraintes $g(x) \leq 0$
2: **Paramètres :** $\mu > 0$, tolérance $\varepsilon > 0$
3: **Initialisation :** $x^0 \in \mathbb{R}^n$, $\lambda^0 \in \mathbb{R}^m_+$, $k \leftarrow 0$
4: **while** critère d'arrêt non satisfait **do**
5: Minimiser le Lagrangien augmenté :

$$x^{k+1} \leftarrow \arg\min_x \left[f(x) + \sum_{i=1}^{m} \lambda_i^k g_i(x) + \frac{\mu}{2} \sum_{i=1}^{m} \left(\max(0, g_i(x))\right)^2 \right]$$

6: Mettre à jour les multiplicateurs de Lagrange :

$$\lambda_i^{k+1} \leftarrow \max\left(0, \lambda_i^k + \mu \cdot g_i(x^{k+1})\right), \quad \forall i$$

7: Vérifier les conditions d'arrêt :
8: **if** $\max_i \left\{\max(0, g_i(x^{k+1}))\right\} \leq \varepsilon$ **et** $\|\nabla f(x^{k+1}) + \sum_i \lambda_i^{k+1} \nabla g_i(x^{k+1})\| \leq \varepsilon$ **then**
9: **Arrêter**
10: **else**
11: $k \leftarrow k + 1$
12: **end if**
13: **end while**
14: **Sortie :** solution x^{k+1}, multiplicateurs λ^{k+1}

Remarque 5.2. *Remarques*

- *Les multiplicateurs λ_i doivent rester positifs : on les projette donc sur $\mathbb{R}_+$.*
- *Le terme $\max(0, g_i(x))$ permet de ne pénaliser que les contraintes actives ou violées.*
- *Le choix du paramètre μ influe sur la convergence : il peut être fixe ou croître à chaque itération.*
- *L'étape de minimisation (ligne 5 de l'algorithme) est en général résolue numériquement.*

Remarque 5.3.
- *Ici, on utilise fminunc pour minimiser le Lagrangien augmenté à chaque itération. Pour des problèmes plus complexes, un solveur de gradient contraint ou une méthode de Newton peut être préférable.*
- *Le gradient de chaque contrainte est supposé être connu.*
- *Vous pouvez ajouter un affichage plus visuel avec plot ou disp du history.*

Code complet avec visualisation

```
1
2 clear; clc; close all;
```

```
3
4  % Définir la fonction objectif et son gradient
5  f = @(x) (x(1)-1)^2 + (x(2)-2)^2;
6  grad_f = @(x) [2*(x(1)-1); 2*(x(2)-2)];
7
8  % Définir la contrainte d'inégalité g(x) <= 0 et son gradient
9  g = @(x) [x(1) + x(2) - 1];
10 grad_g = @(x) [1, 1];  % 1 contrainte, 2 variables
11
12 % Paramètres
13 x0 = [0.5; 0.5];              % point initial
14 lambda0 = [0];                % multiplicateur initial
15 mu = 10;                       % pénalité
16 epsilon = 1e-6;                % tolérance
17 max_iter = 50;                 % nombre max d'itérations
18
19 % Appel de l'algorithme
20 [x_sol, lambda_sol, history] = lagrangien_augmente_inegalite(f, g,
       grad_f, grad_g, x0, lambda0, mu, epsilon, max_iter);
21
22 % Affichage des résultats
23 fprintf('Solution optimale : x = [%.4f, %.4f]\n', x_sol(1), x_sol
       (2));
24 fprintf('Multiplicateur de Lagrange : lambda = %.4f\n', lambda_sol)
       ;
25
26 % Visualisation graphique
27 [X1, X2] = meshgrid(-1:0.05:3, -1:0.05:3);
28 Z = (X1 - 1).^2 + (X2 - 2).^2;
29 G = X1 + X2 - 1;
30
31 figure;
32 contour(X1, X2, Z, 30); hold on;
33 contour(X1, X2, G, [0 0], 'r', 'LineWidth', 2); % frontière de la
       contrainte
34 fill([-1 3 3 -1], [-1 -1 3 3], [0.9 0.9 0.9], 'EdgeColor', 'none');
        % zone interdite
35 contour(X1, X2, G, [0 0], 'r', 'LineWidth', 2); % re-tracer
       frontière
36 plot(x_sol(1), x_sol(2), 'ko', 'MarkerSize', 10, 'MarkerFaceColor',
        'k');
37 title('Méthode du Lagrangien Augmenté avec contrainte d''
       inégalité');
38 xlabel('x_1'); ylabel('x_2');
```

```
legend('Courbes de niveau de f(x)', 'Frontiere g(x)=0', 'Zone
    interdite', 'Solution optimale');
axis equal;
grid on;
```

```
function [x, lambda, history] = lagrangien_augmente_inegalite(f, g,
    grad_f, grad_g, x0, lambda0, mu, epsilon, max_iter)
% Methode du Lagrangien augmente pour contraintes d'inegalite

x = x0;
lambda = lambda0;
k = 0;
history = x';

while k < max_iter
    % Lagrangien augmente
    L_aug = @(xk) f(xk) + lambda' * g(xk) + (mu/2) * sum(max(0, g(
        xk)).^2);

    options = optimoptions('fminunc', 'Display', 'none', 'Algorithm
        ', 'quasi-newton');
    x_new = fminunc(L_aug, x, options);

    % Mise a jour multiplicateurs
    g_val = g(x_new);
    lambda_new = max(0, lambda + mu * g_val);

    % Verification convergence
    stationarite = norm(grad_f(x_new) + grad_g(x_new)' * lambda_new
        );
    violation = max(max(0, g_val));

    % Historique du chemin
    history = [history; x_new'];

    if stationarite <= epsilon && violation <= epsilon
        break;
    end

    % Mise a jour
    x = x_new;
    lambda = lambda_new;
    k = k + 1;
end

```

```
37 end
```

La méthode de Lagrangien augmenté (aussi appelée méthode de pénalité augmentée) combine :

(a) La méthode de Lagrange (via multiplicateurs) : pour imposer les contraintes "de manière douce",

(b) La méthode de pénalité quadratique : pour pénaliser fortement les violations des contraintes

Formule :

$$\mathcal{L}_\rho(x, \lambda) = f(x) + \lambda^T g(x) + \frac{\rho}{2}\|g(x)\|^2$$

Combine pénalité quadratique et multiplicateurs de Lagrange. Plus stable que Uzawa.

Avantages de la méthode

Plus stable numériquement que les méthodes de pénalisation pures

Permet une meilleure convergence des multiplicateurs de Lagrange

Très utilisée dans des solveurs modernes : IPOPT, KNITRO, etc.

Inconvénients / limites

Non trivial à implémenter pour grandes dimensions ou contraintes complexes

La minimisation du Lagrangien augmenté peut être coûteuse

La mise à jour des multiplicateurs doit être bien calibrée (risque de divergence)

5.3.3 SQP (Sequential Quadratic Programming)

Principe : *approximer le problème par une suite de QP :*

$$\min_d \nabla f(x_k)^T d + \frac{1}{2} d^T H_k d$$
$$\text{tel que } g_i(x_k) + \nabla g_i(x_k)^T d = 0$$

Méthode puissante, superlinéaire, mais coûteuse numériquement.

5.3.4 Méthode de Newton-KKT

Résolution directe des conditions KKT par un système linéaire :

$$\begin{bmatrix} \nabla^2_{xx}\mathcal{L} & \nabla g^T \\ \nabla g & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta \lambda \end{bmatrix} = - \begin{bmatrix} \nabla_x \mathcal{L} \\ g(x) \end{bmatrix}$$

Convergence rapide mais méthode numériquement délicate.

5.4 Comparaison synthétique

Méthode	Contraintes	Type	Avantages	Inconvénients
Lagrange	Égalités	NL	Simple à formuler	Ne gère pas les inégalités
KKT	Égalités + inégalités	NL	Conditions générales	Conditions de régularité
Uzawa	Égalités	Convexe	Facile à coder	Convergence lente
Lagrangien Augmenté	Égalités	Général	Plus stable que Uzawa	Paramètres à régler
SQP	Générales	NL	Haute précision	Calculs lourds
Newton-KKT	Générales	NL	Convergence rapide	Complexité numérique

TABLE 5.1 – Tableau récapitulatif des méthodes avec contraintes

NL : Non Linéaire

5.5 Exemples et cas d'application

Exemple 5.1. *Considérons le problème d'optimisation avec contrainte suivant :*
Minimiser

$$(x_1 - 2)^2 + (x_2 - 3)^2$$

sujet à

$$x_1 \geqslant 3$$

Si nous appliquons la condition d'optimalité du premier ordre sur la fonction objectif, le minimum de la fonction est obtenu en $(2, 3)$.
Cependant, en présence d'une contrainte, le minimum se produit à (3, 3). Voir Figure 6.1, où les contours de la fonction sont tracés avec la contrainte.
Notons que le gradient de la fonction (∇f) *et le gradient de la contrainte* (∇g) *sont parallèles entre elles au point optimal.*
Au points de la contrainte (par exemple, le point A), les gradients ne sont pas parallèles les uns aux autres.
D'autres conditions d'optimalité pour les problèmes d'optimisation sous contrainte sont discutées dans la section suivante.

Exemple 5.2. *Considérons le problème d'optimisation avec contrainte suivant :*
Minimiser

$$f(x) = (x_1 - 1)^2 + (x_2 - 5)^2$$

sujet à

$$g_1(x) = -x_1^2 + x_2 - 4 \geqslant 0$$

$$g_2(x) = -(x_1 - 2)^2 + x_2 - 3 \geqslant 0$$

Tracez les contours de la fonction avec les contraintes.

Vérifiez si les conditions de KKT sont satisfaites au point A (0.75, 4.5625).

Les contours de la fonction sont donnés dans la Figure 6.3 avec les contraintes.
Le gradient de la fonction et les contraintes sont donnés par

$$\nabla f(x) = \begin{bmatrix} 2(x_1 - 1) \\ 2(x_2 - 5) \end{bmatrix} = \begin{bmatrix} 2(0.75 - 1) \\ 2(4.5625 - 5) \end{bmatrix} = \begin{bmatrix} -0.5 \\ -0.875 \end{bmatrix}$$

$$\nabla g_1(x) = \begin{bmatrix} -2x_1 \\ 1 \end{bmatrix} = \begin{bmatrix} -1.5 \\ 1 \end{bmatrix}$$

$$\nabla g_2(x) = \begin{bmatrix} -2(x_1 - 2) \\ 1 \end{bmatrix} = \begin{bmatrix} 2.5 \\ 1 \end{bmatrix}$$

Vérifions la condition d'optimalité :

$$-\nabla f(x) = \mu_1 \nabla g_1(x) + \mu_2 = \nabla g_2(x)$$

pour certains μ_1 *et* μ_2 *qui sont* $\geqslant 0$.
On peut montrer qu'au point A

$$\begin{bmatrix} 0.5 \\ 0.875 \end{bmatrix} \approx 0.4218 \begin{bmatrix} -1.5 \\ 1.875 \end{bmatrix} + 0.4531 \begin{bmatrix} 2.5 \\ 1 \end{bmatrix}$$

Ainsi, pour une valeur positive des multiplicateurs ($\mu_1 = 0.4218 et \mu_2 = 0.4513$), *le négatif du gradient de la fonction objectif peut être exprimé comme une combinaison linéaire du gradient des contraintes.*
Les conditions de KKT sont satisfaites au point A. Ainsi, le point A est candidat au minimum de la fonction.
Vérifions la condition de second ordre comme suit :

$$\nabla^2 L = \begin{bmatrix} 2 - 2(\mu_1 + \mu_2) & 0 \\ 0 & 2 \end{bmatrix} = \begin{bmatrix} 0.25 & 0 \\ 0 & 2 \end{bmatrix}$$

Comme cette matrice est définie positive, le minimum de la fonction se produit au point A.

5.5.1 Programmation quadratique

Définition 5.1. *Un problème de **programmation quadratique** (QP) est un problème d'optimisation de la forme :*

$$\text{Minimiser } f(x) = \frac{1}{2}x^T Qx + c^T x$$
$$\text{sous les contraintes : } Ax \leq b \quad ou \quad Ax = b$$

où :

- $x \in \mathbb{R}^n$ *: vecteur des variables,*
- $Q \in \mathbb{R}^{n\times n}$ *: matrice symétrique,*
- $c \in \mathbb{R}^n$, $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$.

Considérons le problème suivant :

$$\begin{aligned} \text{Minimiser} \quad & \frac{1}{2}x^T Qx \\ \text{sous les contraintes} \quad & Ax = b, \end{aligned}$$

où $Q > 0$, $A \in \mathbb{R}^{m\times n}$, $m < n$, $rang(A) = m$. Ce problème est un cas particulier de ce qu'on appelle un problème de programmation quadratique (la forme générale d'un problème de programmation quadratique inclut la contrainte $x \geq 0$). Notons que l'ensemble des contraintes contient un nombre infini de points. Nous allons maintenant montrer, à l'aide du théorème de Lagrange, qu'il existe une solution unique au problème d'optimisation ci-dessus. Ensuite, nous fournissons un exemple illustrant l'application de cette solution à un problème de contrôle optimal.
Pour résoudre le problème, nous formons d'abord la fonction lagrangienne.

$$l(x, \lambda) = \frac{1}{2}x^T Qx + \lambda^T(b - Ax)$$

La condition de Lagrange donne :

$$D_x l(x^*, \lambda^*) = x^{*T}Q - \lambda^{*T}A = 0^T$$

En réécrivant, nous obtenons :

$$x^* = Q^{-1}A^T\lambda^*$$

En prémultipliant les deux membres de l'équation ci-dessus par A, on obtient :

$$Ax^* = AQ^{-1}A^T\lambda^*$$

Sachant que $Ax^* = b$ *et que* $AQ^{-1}A^T$ *est inversible car* $Q > 0$ *et que* A *est de rang* m*, nous pouvons résoudre pour* λ^* *et obtenir :*

$$\lambda^* = (AQ^{-1}A^T)^{-1}b$$

Par conséquent, nous obtenons :

$$x^* = Q^{-1}A^T(AQ^{-1}A^T)^{-1}b$$

Le point x^* *est le seul candidat de minimisation. Pour établir que* x^* *est bien un minimum, nous vérifions que* x^* *satisfait les conditions suffisantes du second ordre.*

Pour cela, nous calculons d'abord la matrice hessienne de la fonction lagrangienne au point (x^*, λ^*)*. Nous avons :*

$$L(x^*, \lambda^*) = Q,$$

qui est définie positive. Ainsi, le point x^* *est un minimum local strict. Nous verrons plus loin que* x^* *est, en fait, un minimum global.*
Le cas particulier où $Q = I_n$*, la matrice identité* $n \times n$*, se réduit au problème d'une équation linéaire de norme minimale. Plus précisément, le problème consiste à minimiser la norme* $\| x \|$ *sous la contrainte* $Ax = b$*. La fonction objectif est ici* $f(x) = \| x \|$*, qui n'est pas différentiable en* $x = 0$*. Ceci empêche l'utilisation du théorème de Lagrange, car ce théorème requiert la différentiabilité de la fonction objectif.*

Nous pouvons surmonter cette difficulté en considérant un problème d'optimisation équivalent :

$$\begin{aligned} \textit{Minimiser} \quad & \frac{1}{2} \| x \|^2 \\ S.C \quad & Ax = b, \end{aligned}$$

La fonction objectif $\frac{1}{2} \| x \|^2$ *admet le même minimum que la fonction objectif précédente* $\| x \|$*. En effet, si* x^* *est tel que pour tout* $x \in \mathbb{R}^n$ *satisfaisant* $Ax = b$*,* $\| x^* \| \leq \| x \|$*, alors* $\frac{1}{2} \| x^* \|^2 \leq \frac{1}{2} \| x \|^2$*. La réciproque est également vraie. Puisque le problème de minimisation de* $\frac{1}{2} \| x \|^2$ *sous la contrainte* $Ax = b$ *est simplement le problème considéré ci-dessus avec* $Q = I_n$*, nous en déduisons facilement que la solution est* $x^* = A^T(AA^T)^{-1}b$*.*

Exemple 5.3. *Considérons le modèle de système linéaire à temps discret avec condi-*

tion initiale x_0 donnée. On peut considérer $\{x_k\}$ comme un signal à temps discret commandé par un signal d'entrée externe $\{u_k\}$. Dans la littérature sur le contrôle, $\{x_k\}$ est appelé l'état à l'instant k. Pour un x_0 donné, notre objectif est de choisir le signal de commande $\{u_k\}$ de sorte que l'état reste « petit » sur un intervalle de temps $[1,\ N]$, tout en ne laissant pas le signal de commande « trop grand ». Pour exprimer la volonté de maintenir l'état $\{x_k\}$ petit, on choisit la séquence de commande afin de minimiser

$$\frac{1}{2}\sum_{i=1}^{N} x_i^2$$

D'autre part, en maintenant un signal de commande suffisamment faible, on minimise

$$\frac{1}{2}\sum_{i=1}^{N} u_i^2$$

Les deux objectifs ci-dessus sont contradictoires en ce sens qu'ils ne peuvent, en général, être atteints simultanément : minimiser le premier peut entraîner un effort de commande important, tandis que minimiser le second peut entraîner des états importants. Il s'agit clairement d'un problème qui nécessite un compromis. Une façon d'aborder le problème consiste à minimiser une somme pondérée des deux fonctions ci-dessus. Plus précisément, on peut formuler le problème comme suit :

$$minimiser \quad \frac{1}{2}\sum_{i=1}^{N}(qx_i^2 + ru_i^2)$$

$$\begin{aligned} Minimiser \quad & \frac{1}{2}\sum_{i=1}^{N}(qx_i^2 + ru_i^2) \\ S.C \quad & x_k = ax_{k-1} + bu_k \quad k = 1...N, \quad x_0 donne \end{aligned}$$

où les paramètres q et r reflètent l'importance relative de maintenir l'état faible par rapport à la limitation de l'effort de commande.

Ce problème est un cas particulier du problème du régulateur linéaire quadratique (LQR) (voir, par exemple, [15], [20], [85], [86], ou [99]). La combinaison des deux objectifs contradictoires que sont la minimisation de l'état et la minimisation de l'effort de contrôle est un cas particulier de l'approche par somme pondérée (voir section 24.4).

Pour résoudre le problème ci-dessus, on peut le reformuler comme un problème de programmation quadratique. On défini,

$$Q = \begin{bmatrix} qI_N & 0 \\ 0 & rI_N \end{bmatrix}$$

$$A = \begin{bmatrix} 1 & & ... & 0 & -b & & ... & 0 \\ -a & 1 & & . & & -b & . & . \\ . & . & & & & & & . \\ & . & & & & & & . \\ & & . & & & & & \\ 0 & & -a & 1 & 0 & ... & & -b \end{bmatrix}$$

$$b = \begin{bmatrix} ax_0 \\ 0 \\ . \\ . \\ . \\ 0 \end{bmatrix}, z = \begin{bmatrix} x_1, ..., x_N, u_1, ..., u_N \end{bmatrix}^T$$

Avec ces définitions, le problème se ramène au problème de programmation quadratique précédemment considéré

$$Minimiser\ f(x) = \frac{1}{2} z^T Q z + c^T z$$

$$S.C : Az \leq b \quad ou \quad Az = b$$

où Q est un système $2N \times 2N$, A est un système $N \times 2N$ et $b \in N$.
La solution est

$$z^* = Q^{-1} A^T (A Q^{-1} A^T)^{-1} b$$

Applications en optimisation continue

(a) ***Finance – Optimisation de portefeuille (Markowitz)***
Objectif : minimiser le risque (variance du portefeuille) pour un rendement donné.

$$\min \frac{1}{2} x^T Q x \quad sous : \sum x_i = 1, \quad x_i \geq 0$$

où Q est la matrice de covariance des rendements.

(b) ***Apprentissage automatique – Machines à vecteurs de support (SVM)***
En classification binaire, les SVM cherchent à maximiser la marge entre deux

classes. Le problème dual est un QP.

(c) ***Contrôle optimal – MPC (Model Predictive Control)***
Utilisé en ingénierie pour le contrôle de systèmes dynamiques (automobile, robotique). À chaque pas, un QP est résolu pour minimiser une fonction coût soumise à des contraintes physiques.

(d) ***Planification de trajectoires – Robotique et drones***
Optimisation de l'énergie, la courbure ou la distance parcourue sous contraintes dynamiques. Cela se modélise comme un QP convexe.

(e) ***Réseaux électriques – Optimisation de la puissance (OPF)***
Minimisation des pertes ou du coût de production. Approximations quadratiques aboutissent à des QP avec contraintes.

(f) ***Économie – Équilibres de Nash quadratiques***
Certains jeux non-coopératifs (où les fonctions de coût sont quadratiques) peuvent être formulés comme des problèmes QP.

Méthodes de résolution

Parmi les méthodes de résolution classiques des QP, on trouve :

- *Méthode du gradient conjugué (pour QP sans contraintes),*
- *Conditions KKT (Karush-Kuhn-Tucker),*
- *Méthodes Active Set, Interior Point, SQP (Sequential Quadratic Programming),*
- *Solveurs spécialisés :* `quadprog`, `cvxopt`, `OSQP`, `Gurobi`, `CPLEX`.

Remarque 5.4. *Si la matrice Q est* ***définie positive****, alors le problème est* ***convexe****. Cela implique que tout minimum local est également un minimum global. On peut alors utiliser des algorithmes efficaces garantissant la convergence.*

Exemple 5.4. *Exemple simple*
Minimiser la fonction quadratique suivante :

$$f(x, y) = x^2 + y^2 + 3x + 4y$$

Sous les contraintes :

$$x + y \leq 1, \quad x \geq 0, \quad y \geq 0$$

Ce problème est équivalent à :

$$Q = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}, \quad c = \begin{bmatrix} 3 \\ 4 \end{bmatrix}$$

Exemple 5.5. *(a) Problème*

On cherche à minimiser la fonction quadratique suivante :

$$f(x, y) = x^2 + y^2 + 3x + 4y$$

Sous les contraintes :

$$x + y \leq 1, \quad x \geq 0, \quad y \geq 0$$

(b) Formulation standard

On réécrit le problème sous la forme standard :

$$\min_{x \in \mathbb{R}^2} \quad \frac{1}{2} x^T Q x + c^T x$$

Avec :

$$Q = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}, \quad c = \begin{bmatrix} 3 \\ 4 \end{bmatrix}$$

Contraintes sous forme matricielle :

$$A = \begin{bmatrix} 1 & 1 \\ -1 & 0 \\ 0 & -1 \end{bmatrix}, \quad b = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

(c) Code MATLAB

```
% Exemple de programmation quadratique en MATLAB

% Matrice Q (doit  tre  sym trique et d finie positive)
Q = [2 0; 0 2];

% Vecteur c
c = [3; 4];

% Contraintes lin aires : A * x <= b
A = [1 1; -1 0; 0 -1];
b = [1; 0; 0];

% Pas de contraintes d'  galit
```

```
Aeq = [];
beq = [];

% Bornes (optionnel)
lb = []; % ou lb = [0; 0];
ub = [];

% R solution avec quadprog
options = optimoptions('quadprog', 'Display', 'off');
[x_opt, fval] = quadprog(Q, c, A, b, Aeq, beq, lb, ub, [],
    options);

% Affichage des r sultats
fprintf('Solution optimale : x = %.4f, y = %.4f\n', x_opt(1),
    x_opt(2));
fprintf('Valeur minimale de la fonction objectif : f(x) = %.4f\
    n', fval);
```

Listing 5.2 – Résolution d'un problème QP avec `quadprog` en MATLAB

(d) *Résultat*

Le solveur renvoie la solution optimale (x^*, y^*) *qui minimise la fonction quadratique sous les contraintes données. Cette solution est unique car la fonction est convexe (la matrice* Q *est définie positive).*

```
% D finition de la grille pour x et y
[x, y] = meshgrid(linspace(-1, 2, 100), linspace(-1, 2, 100));

% Fonction quadratique     minimiser
f = x.^2 + y.^2 + 3*x + 4*y;

% Affichage des courbes de niveau
figure;
contour(x, y, f, 50); % 50 niveaux
xlabel('x');
ylabel('y');
title('Visualisation du probl me quadratique');
hold on;

% Contraintes : x + y <= 1, x >= 0, y >= 0
% Tracer la r gion admissible

% Cr ation de la r gion admissible avec fill()
x1 = [0, 1, 0];
y1 = [1, 0, 0];
```

```
fill(x1, y1, [0.9 0.9 1], 'FaceAlpha', 0.5, 'EdgeColor', 'none');
legend('Courbes de niveau', 'R gion admissible');

% Formulation QP
Q = [2 0; 0 2];
c = [3; 4];
A = [1 1; -1 0; 0 -1];
b = [1; 0; 0];

% R solution avec quadprog
options = optimoptions('quadprog', 'Display', 'off');
[x_opt, fval] = quadprog(Q, c, A, b, [], [], [], [], options);

% Afficher la solution optimale
plot(x_opt(1), x_opt(2), 'ro', 'MarkerSize', 10, 'MarkerFaceColor',
    'r');
text(x_opt(1)+0.05, x_opt(2), sprintf('Minimum (%.2f, %.2f)', x_opt
    (1), x_opt(2)));

hold off;
grid on;
axis([ -0.5 1.5 -0.5 1.5 ])
```

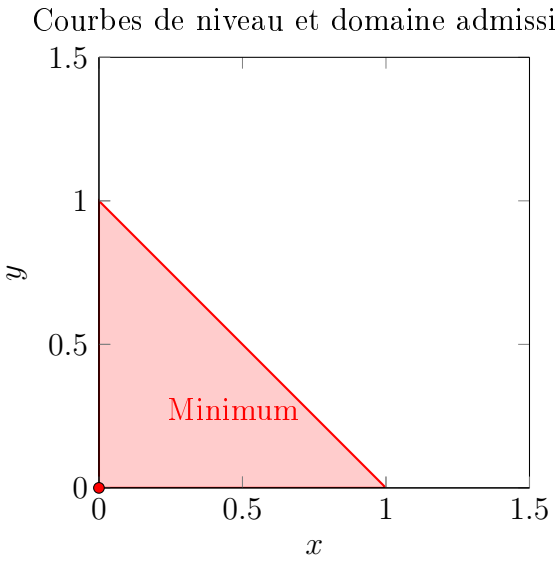


FIGURE 5.1 – Courbes de niveau de $f(x, y) = x^2 + y^2 + 3x + 4y$ avec domaine admissible

5.5.2 Optimisation en apprentissage automatique

*L'**optimisation** joue un rôle central en **apprentissage automatique**, car la plupart des algorithmes apprennent à partir des données en **minimisant une fonction***

de coût. *Cette fonction mesure l'erreur entre les prédictions du modèle et les sorties attendues. Le choix de la méthode d'optimisation influence la vitesse, la stabilité et la performance du modèle.*

Formulation générale

Soit un modèle paramétré par un vecteur de paramètres θ, et une fonction de coût $J(\theta)$ que l'on souhaite minimiser :

$$\min_{\theta} \; J(\theta)$$

où $J(\theta)$ peut être, par exemple, l'erreur quadratique moyenne, la perte logistique, la cross-entropy, etc.

Exemples classiques

a) Régression linéaire

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

où $h_\theta(x) = \theta^T x$ est le prédicteur linéaire.

b) Régression logistique

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^{m} \left[y^{(i)} \log(h_\theta(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)})) \right]$$

c) Réseaux de neurones

Minimisation de la fonction de perte (cross-entropy, MSE...) en ajustant des milliers (voire millions) de paramètres θ.

d) Machines à vecteurs de support (SVM)

Le problème dual des SVM est un problème de ***programmation quadratique***.

e) K-means (clustering)

Minimisation de la somme des distances intra-cluster :

$$J = \sum_{i=1}^{k} \sum_{x \in C_i} \|x - \mu_i\|^2$$

Méthodes d'optimisation courantes

a) Descente de gradient (Gradient Descent)

$$\theta := \theta - \alpha \nabla_\theta J(\theta)$$

b) Descente de gradient stochastique (SGD)

Utilise un sous-échantillon (batch ou mini-batch) pour mettre à jour les paramètres :

$$\theta := \theta - \alpha \nabla_\theta J(\theta; x^{(i)}, y^{(i)})$$

c) Méthodes avancées

- ***Momentum***
- ***RMSProp***
- ***Adam (Adaptive Moment Estimation)***

Ces méthodes adaptent dynamiquement le taux d'apprentissage et accélèrent la convergence.

d) Optimisation convexe vs non convexe

- *Si $J(\theta)$ est **convexe**, toute solution locale est aussi une solution globale.*
- *Dans les réseaux de neurones, $J(\theta)$ est souvent **non convexe**, ce qui complique la recherche du minimum global.*

Applications pratiques

- *Classification et régression supervisée*
- *Réduction de dimension (PCA, auto-encodeurs)*
- *Détection d'anomalies*
- *Traitement du langage naturel (NLP)*
- *Vision par ordinateur (convolutional neural networks)*

Conclusion

L'optimisation est un moteur essentiel de l'apprentissage automatique. Choisir une fonction de coût appropriée et une méthode d'optimisation efficace est indispensable pour entraîner des modèles performants sur des données complexes.

5.5.3 Exemple de problème inverse

Définition 5.2. *Un **problème inverse en optimisation** consiste à **déterminer les paramètres d'un problème d'optimisation** (par exemple la fonction objectif ou les contraintes) à partir de **données observées**, comme une solution optimale connue.*

Cela s'oppose au problème direct, où l'on cherche la solution optimale à partir d'une fonction objectif et de contraintes données.

Exemple 5.6. *Problème direct*

Soit le problème suivant :

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) = c^T x \\ \textit{sous contraintes} \quad & Ax \leq b \end{aligned}$$

Exemple 5.7. *Problème inverse*

On connaît :

- *une solution optimale x^**
- *les contraintes $Ax \leq b$*

On cherche :

- *le vecteur $c \in \mathbb{R}^n$ tel que x^* soit solution optimale de*

$$\min\{c^T x \mid Ax \leq b\}$$

- *éventuellement sous une condition de normalisation, par exemple $\|c\| = 1$*

Exemple 5.8. *Exemple numérique*

Considérons :

$$x^* = \begin{pmatrix} 2 \\ 3 \end{pmatrix}, \quad A = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ -1 & -1 \end{pmatrix}, \quad b = \begin{pmatrix} 2 \\ 3 \\ -5 \end{pmatrix}$$

On cherche $c = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix}$ tel que x^* soit optimal.

Conditions d'optimalité (KKT)

Le problème étant linéaire, la condition nécessaire d'optimalité est :

$$c = A^T \lambda$$

avec :

$\lambda \geq 0$ *(multiplicateurs de Lagrange associés aux contraintes)*

$\lambda_i = 0$ *si la contrainte i n'est pas active à x^**

On peut ainsi identifier les contraintes actives à x^*, puis résoudre le système pour retrouver c.

Exemples typiques des problèmes inverses

Science des données *: apprentissage de préférences à partir de données observées.*

Intelligence artificielle *: apprentissage inverse (inverse reinforcement learning).*

Optimisation paramétrique *: calibrage d'un modèle d'optimisation sur des données réelles.*

Chapitre 6

Implémentation de divers algorithmes et applications

L'implémentation des algorithmes en optimisation continue constitue une étape essentielle pour la résolution numérique de problèmes de minimisation et de maximisation de fonctions continues, différentiables ou non, avec ou sans contraintes. Elle permet de traduire les méthodes théoriques, telles que les algorithmes de gradient, de Newton et quasi-Newton, en procédures numériques exploitables, tout en analysant leurs propriétés de convergence, leur stabilité et leur complexité computationnelle.

MATLAB (Matrix Laboratory) est un environnement de calcul scientifique et de programmation numérique largement utilisé pour la modélisation mathématique, l'analyse algorithmique et la résolution de problèmes d'optimisation. Il offre des fonctionnalités avancées de calcul matriciel, de visualisation et une bibliothèque riche de solveurs d'optimisation intégrés, notamment fminunc, fmincon et lsqnonlin, facilitant la mise en œuvre des méthodes d'optimisation continue.

L'implémentation des algorithmes d'optimisation continue sous MATLAB présente un intérêt majeur, car elle permet une programmation rapide et fiable, une analyse précise des performances numériques et une visualisation efficace des résultats. De plus, MATLAB offre un cadre expérimental adapté à la comparaison des algorithmes, à l'étude de l'influence des paramètres numériques et à la validation des solutions obtenues, ce qui en fait un outil privilégié pour l'optimisation continue.

6.1 Implémentation des algorithmes sans contraintes

6.1.1 Méthode du Gradient

```
function [x, fval, iter] = gradient_descente(f, gradf, x0, alpha,
    tol, iter_max)
% f         : fonction a minimiser
% gradf     : gradient de f
% x0        : point initial
% alpha     : pas de descente
% tol       : tolerance
% iter_max  : nombre maximal d'iterations

x = x0;

for iter = 1:iter_max
     g = gradf(x);
     if norm(g) < tol
          break;
     end
     x = x - alpha * g;
end

fval = f(x);
end
```

Listing 6.1 – Algorithme de la descente de gradient

```
f = @(x) x(1)^2 + x(2)^2;
gradf = @(x) [2*x(1); 2*x(2)];

x0 = [2; -1];
alpha = 0.1;
tol = 1e-6;
iter_max = 100;

[x_opt, f_min] = gradient_descente(f, gradf, x0, alpha, tol,
    iter_max);
```

6.1.2 Méthode du Gradient à pas fixe

Fonction `gradient_fixe.m`

```
function [x, k, history] = gradient_fixe(f, grad_f, x0, alpha,
    epsilon, N_max)
% METHODE DU GRADIENT     PAS FIXE
%
% Entr es :
%    f         : handle vers la fonction     minimiser f(x)
%    grad_f    : handle vers le gradient de f, i.e. grad_f(x)
%    x0        : point initial (vecteur colonne)
%    alpha     : pas fixe (learning rate)
%    epsilon   : tol rance sur la norme du gradient
%    N_max     : nombre maximal d'it rations
%
% Sorties :
%    x         : solution approch e
%    k         : nombre    d  itrations    effectu es
%    history   : historique des points (pour affichage)

    x = x0;
    k = 0;
    history = x;

    while norm(grad_f(x)) > epsilon && k < N_max
        x = x - alpha * grad_f(x);
        k = k + 1;
        history = [history, x];
    end
end
```

Listing 6.2 – Méthode du gradient à pas fixe

Exemple d'utilisation

```
% D finition d'une fonction quadratique
A = [3 0; 0 1];
b = [1; 2];

f = @(x) 0.5 * x' * A * x - b' * x;
grad_f = @(x) A * x - b;

% Param tres de l'algorithme
x0 = [0; 0];
alpha = 0.1;
epsilon = 1e-6;
N_max = 1000;

```

```
% Appel de la m thode
[x_sol, k, hist] = gradient_fixe(f, grad_f, x0, alpha, epsilon,
    N_max);

% Affichage du r sultat
fprintf('Solution trouv e en %d it rations :\n', k);
disp(x_sol);

% Affichage de la trajectoire
plot(hist(1,:), hist(2,:), '-o');
xlabel('x_1'); ylabel('x_2');
title('Trajectoire du gradient   pas fixe');
grid on;
```

Listing 6.3 – Minimisation d'une fonction quadratique

Exemple : Cas d'une Fonction Non Linéaire

Fonction choisie : Fonction de Rosenbrock

$$f(x, y) = (1 - x)^2 + 100(y - x^2)^2$$

Gradient :

$$\nabla f(x, y) = \begin{bmatrix} -2(1 - x) - 400x(y - x^2) \\ 200(y - x^2) \end{bmatrix}$$

Code MATLAB

```
f = @(x) (1 - x(1))^2 + 100 * (x(2) - x(1)^2)^2;

grad_f = @(x) [
    -2 * (1 - x(1)) - 400 * x(1) * (x(2) - x(1)^2);
    200 * (x(2) - x(1)^2)
];

x0 = [-1.2; 1];
alpha = 1e-3;
epsilon = 1e-6;
N_max = 10000;

[x_sol, k, hist] = gradient_fixe(f, grad_f, x0, alpha, epsilon,
    N_max);

```

```
fprintf('Solution trouvée après %d itérations :\n', k);
disp(x_sol);

plot(hist(1,:), hist(2,:), '-o');
xlabel('x'); ylabel('y');
title('Trajectoire sur la fonction de Rosenbrock');
grid on;
```

Listing 6.4 – Minimisation de la fonction de Rosenbrock

Le point converge vers $(1, 1)$.

6.1.3 Méthode du Gradient à pas optimale

Code MATLAB

```
function gradient_optimal()

    % Paramètres
    tol = 1e-6;
    max_iter = 100;
    x = [0; 0];  % Point initial

    for k = 1:max_iter
        g = gradient_f(x);

        if norm(g) < tol
            fprintf('Convergence atteinte après %d itérations.\n'
                , k-1);
            break;
        end

        % Direction de descente
        p = -g;

        % Pas optimal (minimisation unidimensionnelle)
        alpha = fminbnd(@(a) f(x + a * p), 0, 1);

        % Mise à jour
        x = x + alpha * p;

        % Affichage
        fprintf('Itération %d : x = [%f, %f], f(x) = %f, alpha = %
            f\n', ...
                k, x(1), x(2), f(x), alpha);
    end
```

```
29
30      fprintf('Solution finale : x* = [%f, %f], f(x*) = %f\n', x(1),
            x(2), f(x));
31  end
32
33  function val = f(x)
34      val = (x(1) - 1)^2 + 2 * (x(2) - 2)^2;
35  end
36
37  function g = gradient_f(x)
38      g = [2*(x(1) - 1); 4*(x(2) - 2)];
39  end
```

Listing 6.5 – Méthode du gradient à pas optimal

Résultats attendus

À chaque itération, la méthode calcule la direction de descente, puis effectue une minimisation exacte sur le pas.

Voici un extrait typique de la sortie :

```
Itération 1 : x = [0.500000, 1.000000], f(x) = 2.250000, alpha = 0.25
Itération 2 : x = [0.750000, 1.500000], f(x) = 0.562500, alpha = 0.25
Itération 3 : x = [0.875000, 1.750000], f(x) = 0.140625, alpha = 0.25
...
Convergence atteinte après 14 itérations.
Solution finale : x* = [1.000000, 2.000000], f(x*) = 0.000000
```

Commentaires

- *Le pas optimal assure une convergence plus rapide que le gradient à pas fixe.*
- *Ici, la fonction est quadratique, donc le pas est constant ($\alpha_k = 0.25$).*
- *L'algorithme converge en une quinzaine d'itérations avec bonne précision.*
- *La méthode 'fminbnd' permet une minimisation unidimensionnelle automatique.*

Exercice 6.1. *Minimiser* $f(x) = (x_1 - 2)^2 + (x_2 + 3)^2$

$$\nabla f(x) = \begin{pmatrix} 2(x_1 - 2) \\ 2(x_2 + 3) \end{pmatrix}, \quad x^* = (2, -3)$$

Première itération depuis $x^{(0)} = (0, 0)$*, direction :* $-\nabla f(0, 0) = (-4, 6)$

$\longrightarrow$ *calculer* α*, mise à jour de* x*, etc.*

6.1.4 Méthode du Gradient Conjugué

```
function [x, k] = gradient_conjugue(A, b, x0, tol, kmax)
% GRADIENT_CONJUGUE  Methode du gradient conjugue pour Ax = b
%
% Entrees :
%   A     : matrice symetrique definie positive
%   b     : vecteur second membre
%   x0    : point initial
%   tol   : tolerance sur le residu
%   kmax  : nombre maximal d'iterations
%
% Sorties :
%   x     : solution approchee
%   k     : nombre d'iterations effectuees

% Initialisation
x = x0;
r = b - A*x;
d = r;
k = 0;

while norm(r) > tol && k < kmax
    alpha = (r'*r) / (d'*A*d);
    x = x + alpha*d;
    r_new = r - alpha*A*d;

    beta = (r_new'*r_new) / (r'*r);
    d = r_new + beta*d;

    r = r_new;
    k = k + 1;
end
end
    Exemple d utilisation
A = [4 1; 1 3];
b = [1; 2];
x0 = [0; 0];
tol = 1e-6;
kmax = 100;

[x, k] = gradient_conjugue(A, b, x0, tol, kmax);

disp('Solution approchee :');
disp(x)
```

```
44
45 disp('Nombre d''it rations :');
46 disp(k)
47       Remarques importantes
48         La matrice doit  tre  sym trique d finie positive
49
50         En dimension
51
52 n, la m thode converge en au plus
53
54 n it rations (en arithm tique exacte)
55
56         Beaucoup plus rapide que la descente de gradient
57
58       Variante ultra-courte (pour TD)
59 x = x0;
60 r = b - A*x;
61 d = r;
62
63 for k = 1:kmax
64     alpha = (r'*r)/(d'*A*d);
65     x = x + alpha*d;
66     r_new = r - alpha*A*d;
67     if norm(r_new) < tol, break; end
68     beta = (r_new'*r_new)/(r'*r);
69     d = r_new + beta*d;
70     r = r_new;
71 end
```

Listing 6.6 – Méthode du gradient à pas fixe

Exemple 6.1. *A = [4 1 ; 1 3]; b = [1 ; 2]; x0 = [0 ; 0]; tol = 1e-6; kmax = 100;*
[x, k] = gradient_conjugue(A, b, x0, tol, kmax);
disp('Solution approchée :'); disp(x)
disp('Nombre d"itérations :'); disp(k)

6.1.5 Méthode de Newton

```
1 function [x, k] = newton_method(f, gradf, hessf, x0, tol, kmax)
2 % NEWTON_METHOD  M thode de Newton pour l'optimisation
3 %
4 % Entr es :
5 %   f       : fonction @(x) f(x)
6 %   gradf   : fonction @(x) gradient de f
```

```
%    hessf   : fonction @(x) Hessienne de f
%    x0      : point initial
%    tol     : tolrance sur le gradient
%    kmax    : nombre maximal d'itrations
%
% Sorties :
%    x       : approximation du minimum
%    k       : nombre d'itrations effectues

x = x0;
k = 0;

while norm(gradf(x)) > tol && k < kmax
    g = gradf(x);          % Gradient
    H = hessf(x);          % Hessienne

    % Vrifier que Hessienne est inversible
    if rcond(H) < 1e-12
        warning('Hessienne proche de singulire, arrt.');
        break;
    end

    % Direction de Newton
    d = -H\g;

    % Mise    jour du point (Newton pur, alpha=1)
    x = x + d;

    k = k + 1;
end

if k == kmax
    warning('Nombre maximal d iterations atteint');
end
end
```

Listing 6.7 – Méthode du gradient à pas fixe

Exemple 6.2. *On considère la fonction*

$$f(x, y) = (x - 1)^2 + 2(y - 2)^2$$

et on cherche à trouver son minimum à l'aide de la méthode de Newton.

(a) Calcul du gradient et de la Hessienne

$$\nabla f(x,y) = \begin{pmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{pmatrix} = \begin{pmatrix} 2(x-1) \\ 4(y-2) \end{pmatrix}, \quad \nabla^2 f(x,y) = \begin{pmatrix} 2 & 0 \\ 0 & 4 \end{pmatrix}.$$

(b) Point initial

$$x_0 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

(c) Première itération

$$g_0 = \nabla f(x_0) = \begin{pmatrix} -2 \\ -8 \end{pmatrix}, \quad H_0 = \nabla^2 f(x_0) = \begin{pmatrix} 2 & 0 \\ 0 & 4 \end{pmatrix}.$$

Direction de Newton :

$$d_0 = -H_0^{-1} g_0 = - \begin{pmatrix} 1/2 & 0 \\ 0 & 1/4 \end{pmatrix} \begin{pmatrix} -2 \\ -8 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}.$$

Mise à jour :

$$x_1 = x_0 + d_0 = \begin{pmatrix} 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 1 \\ 2 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}.$$

(d) Vérification du gradient au nouveau point

$$g_1 = \nabla f(x_1) = \begin{pmatrix} 2(1-1) \\ 4(2-2) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

(e) Conclusion

La méthode converge en une seule itération à cause de la fonction quadratique, et le minimum est

$$x^* = \begin{pmatrix} 1 \\ 2 \end{pmatrix}.$$

Remarque 6.1. *Pour les fonctions quadratiques, la méthode de Newton converge en **une seule étape** si le Hessien est constant et définie positive.*

6.1.6 Méthode de Relaxation

```
function [x, k] = SOR(A, b, x0, omega, tol, kmax)
% SOR    Methode de relaxation successive (Successive Over-
    Relaxation)
%
% Entres :
```

```
%    A       : matrice du syst me (n x n)
%    b       : vecteur du syst me (n x 1)
%    x0      : vecteur initial (n x 1)
%    omega   : param tre de relaxation (0 < omega < 2)
%    tol     : tol rance sur la diff rence entre it rations
    successives
%    kmax    : nombre maximal d'it rations
%
% Sorties :
%    x       : solution approximative
%    k       : nombre d'it rations effectu es

n = length(b);
x = x0;
k = 0;

while k < kmax
    x_old = x;
    for i = 1:n
        sigma = 0;
        for j = 1:i-1
            sigma = sigma + A(i,j)*x(j);
        end
        for j = i+1:n
            sigma = sigma + A(i,j)*x(j_old);
        end
        x(i) = (1-omega)*x(i) + (omega/A(i,i))*(b(i) - sigma);
    end

    % Crit re d'arr t
    if norm(x - x_old, inf) < tol
        break;
    end
    k = k + 1;
end

if k == kmax
    warning('Nombre maximal d iterations atteint');
end
end
```

Listing 6.8 – Méthode du gradient à pas fixe

```

% Exemple : syst me 2x2
A = [4 -1; -2 5];
```

```
b = [3; -1];
x0 = [0; 0];
omega = 1.25;    % param tre de relaxation
tol = 1e-6;
kmax = 100;

[x, k] = SOR(A, b, x0, omega, tol, kmax);

disp('Solution approximative :');
disp(x)
disp('Nombre d iterations :');
disp(k)
```

6.2 Implémentation des algorithmes avec contraintes

6.2.1 Méthode des multiplicateurs de Lagrange

```
function [x_star, lambda_star] = lagrange_method(f, gradf, h, gradh
    , x0)
% LAGRANGE_METHOD : R solution d'un probl me d'optimisation avec
    contraintes d'  galit
%
% Entr es :
%   f       : fonction objectif @(x) f(x)
%   gradf   : gradient de f @(x) gradf(x)
%   h       : contraintes @(x) h(x), vecteur m x 1
%   gradh   : gradient jacobien des contraintes @(x) gradh(x),
    matrice m x n
%   x0      : point initial (n x 1)
%
% Sorties :
%   x_star      : solution optimale
%   lambda_star : multiplicateurs de Lagrange

x = x0;
tol = 1e-6;
maxIter = 100;
n = length(x0);
m = length(h(x0));
lambda = zeros(m,1);  % initialisation des multiplicateurs

for k = 1:maxIter
    % Calcul des gradients et jacobien
```

```
    g = gradf(x);        % gradient de f
    H = gradh(x);        % Jacobien des contraintes h (m x n)

    % Syst me de KKT
    % | H^T  gradf^T | | dx      | = | -g |
    % | H      0     | | lambda  |   | -h(x) |
    KKT = [eye(n), H'; H, zeros(m)];
    rhs = [-g; -h(x)];

    sol = KKT\rhs;
    dx = sol(1:n);
    dlambda = sol(n+1:end);

    % Mise    jour
    x = x + dx;
    lambda = lambda + dlambda;

    % Crit re d'arr t
    if norm(dx) < tol && norm(h(x)) < tol
        break;
    end
end

x_star = x;
lambda_star = lambda;

if k == maxIter
    warning('Nombre maximal d iterations atteint');
end

end
```

Exemple 6.3. *Minimiser* $f(x, y) = x^2 + v^2$ *sous la contraint :* $x + y - 1 = 0$

```
f       = @(X) X(1)^2 + X(2)^2;
gradf   = @(X) [2*X(1); 2*X(2)];
h       = @(X) X(1) + X(2) - 1;
gradh   = @(X) [1 1];  % Jacobien m x n
x0      = [0; 0];

[x_star, lambda_star] = lagrange_method(f, gradf, h, gradh, x0);

disp('Solution optimale x* :');
disp(x_star)
disp('Multiplicateur de Lagrange lambda* :');
```

```
disp(lambda_star)
```

6.2.2 Méthode d'Uzawa

```
function [x, lambda, k] = uzawa(A, b, C, d, rho, tol, kmax)
% UZAWA : Methode d'Uzawa pour un probleme quadratique avec
    contrainte lineaire
%
% Minimise : (1/2)x'Ax - b'x   sous   Cx = d
%
% Entrees :
%   A, b       : donnees du probleme quadratique
%   C, d       : contraintes lineaires
%   rho        : pas de descente pour le dual
%   tol        : tolerance sur la contrainte
%   kmax       : nombre maximal d'iterations
%
% Sorties :
%   x          : solution optimale
%   lambda     : multiplicateur de Lagrange
%   k          : nombre d'iterations effectuees

% Initialisation
[n, ~] = size(A);
[m, ~] = size(C);
x = zeros(n,1);
lambda = zeros(m,1);

for k = 1:kmax
    %  etape 1 : resoudre pour x
    x = A \ (b - C' * lambda);

    %  etape 2 : mise    jour du multiplicateur
    lambda = lambda + rho * (C*x - d);

    % Critere d'arret
    if norm(C*x - d) < tol
        break;
    end
end

if k == kmax
    warning('Nombre maximal d iterations atteint');
end
```

```
end
```

Listing 6.9 – Méthode d'Uzawa en MATLAB

Exemple 6.4.

```
A = [4 1; 1 3];
b = [1; 2];
C = [1 1];
d = 1;

rho = 0.1;
tol = 1e-6;
kmax = 100;

[x_star, lambda_star, k] = uzawa(A,b,C,d,rho,tol,kmax);

disp('Solution x* :');
disp(x_star)
disp('Multiplicateur lambda* :');
disp(lambda_star)
disp('Nombre d iterations :');
disp(k)
```

Méthode de Lagrange augmenté

```
function [x, lambda, history] = lagrangien_augmente_inegalite(f, g,
    grad_f, grad_g, x0, lambda0, mu, epsilon, max_iter)
% M thode du Lagrangien augment  pour des contraintes d'
    in galit
%
% Entr es :
%   f         : handle fonction objectif f(x)
%   g         : handle fonction contraintes g(x), retourne un
    vecteur
%   grad_f    : handle gradient de f(x)
%   grad_g    : handle gradient de g(x), retourne une matrice m x n
%   x0        : point initial
%   lambda0   : multiplicateurs de Lagrange initiaux (m x 1)
%   mu        : param tre de p nalisation (> 0)
%   epsilon   : tol rance d'arr t
%   max_iter  : nombre max d'it rations

% Initialisation
x = x0;
lambda = lambda0;
```

```
k = 0;
history = [];

while k < max_iter
    % Minimisation du Lagrangien augment  (r solu ici par fminunc
        pour simplicit )
    L_aug = @(xk) f(xk) + lambda' * g(xk) + (mu/2) * sum(max(0, g(
        xk)).^2);
    options = optimoptions('fminunc', 'Display', 'none', 'Algorithm
        ', 'quasi-newton');
    x_new = fminunc(L_aug, x, options);

    % Mise    jour des multiplicateurs de Lagrange
    g_val = g(x_new);
    lambda_new = max(0, lambda + mu * g_val);

    % Crit re d'arr t
    stationarite = norm(grad_f(x_new) + grad_g(x_new)' * lambda_new
        );
    violation = max(max(0, g_val));
    history = [history; norm(g_val), stationarite];

    if violation <= epsilon && stationarite <= epsilon
        break;
    end

    % Mise    jour
    x = x_new;
    lambda = lambda_new;
    k = k + 1;
end

end
```

Listing 6.10 – Méthode du Lagrangien augmenté

Exemple 6.5.

```
% Exemple : minimiser f(x) = (x1-1)^2 + (x2-2)^2
% sous la contrainte : g(x) = x1 + x2 - 1 <= 0

f = @(x) (x(1)-1)^2 + (x(2)-2)^2;
grad_f = @(x) [2*(x(1)-1); 2*(x(2)-2)];

g = @(x) [x(1) + x(2) - 1];
grad_g = @(x) [1, 1];  % 1 contrainte, 2 variables

```

```
x0 = [0; 0];
lambda0 = [0];
mu = 10;
epsilon = 1e-6;
max_iter = 100;

[x_sol, lambda_sol, histo] = lagrangien_augmente_inegalite(f, g,
    grad_f, grad_g, x0, lambda0, mu, epsilon, max_iter);

fprintf('Solution optimale : x = [%.4f, %.4f]\n', x_sol(1), x_sol
    (2));
fprintf('Multiplicateur de Lagrange : lambda = %.4f\n', lambda_sol)
    ;
```

Chapitre 7

Annexe

7.1 Examens

Examen Final 2020, Durée 60 minutes

Exercice 1 *On considère la fonction f définie sur $\mathbb{R}^2$ par :*

$$f(x,y) = x^4 + y^4 - 2(x-y)^2$$

(a) Montrer qu'il existe $(\alpha, \beta) \in \mathbb{R}^2_+$ (et les déterminer) tel que

$$f(x,y) \geq \alpha \parallel (x,y) \parallel^2 + \beta$$

pour tous $(x,y) \in \mathbb{R}^2$, où la notation $\| \, . \, \|$ désigne la norme euclidienne de $\mathbb{R}^2$.
En déduire que le problème :

$$\min f(x,y), \qquad (x,y) \in R^2 \qquad (P)$$

possède au moin une solution.

(b) La fonction f est-elle convexe sur $\mathbb{R}^2$?

(c) Déterminer les points critiques de f, et préciser leur nature (minimum local, maximum local, point selle, ...). Résoudre alors le problème (P).

(d) Pouvez vous appliquer la méthode du gradient ! Sinon proposer une autre méthode d'optimisation sans contraintes.

Exercice 2 *Soit $T(v) = \frac{1}{2}(Av, v) - (b, v)$ où A est une matrice symétrique de $\mathbb{R}^n$ dans $\mathbb{R}^n$ et $v \in \mathbb{R}^n$, une fonctionnelle quadratique de $\mathbb{R}^n$ dans $\mathbb{R}$.*

Examen

Exercice Final 2023, Durée 90 minutes :

Exercice 1 *Résoudre le problème d'optimisation avec contraintes ci-dessous (calculer tous les points) en appliquant les conditions de KKT pour la fonction objectif $f(x,y)$ soumis à la fonction de contrainte $g_1(x,y)$ et $g_2(x,y)$.*

$$(P): \begin{cases} Minimiser f(x,y) = (x-2)^2 + 2(y-3)^2 \\ g_1(x,y): \quad x + 4y \leq 3 \\ g_2(x,y): \quad -x + y \leq 0 \end{cases}$$

Exercice 2 *Considérons le problème d'optimisation avec contraintes suivant :*

$$(P): \begin{cases} Maximiser(x_1^3 - 3x_2) \\ 2x_1 + x_2 - 2 \geq 0 \\ x_1 + 2x_2 - 10 \leq 0 \\ 7x_1 + 2x_2 - 28 \leq 0 \\ x_1 \ \ et \ \ x_2 \ \geq 0 \end{cases}$$

(a) Déterminer géométriquement le domaine réalisable constitué par les contraintes.

(b) Résoudre le problème au moyen des conditions de Karush-Kuhn-Tucker (KKT).

Bonne chance

Examen Final 2021, Durée 60 minutes

Remarque 7.1. *Résoudre trois problèmes selon les étapes suivantes :*

(a) Indiquer le type du problème d'optimisation à résoudre.

(b) Énoncer le théorème appliqué dans chaque problème.

(c) Résoudre le problème convenablement.

Exercice 1

Étant donné une zone fixe de carton, nous souhaitons construire une boîte en carton fermée avec un volume maximal. Notons, les dimensions de la boîte avec un volume maximum par x_1, x_2 et x_3, et soit A la zone fixe donnée de carton.
Le problème peut alors être formulé comme suit :

$$P1 : \quad \begin{cases} Maximiser : & x_1x_2x_3 \\ SC & x_1x_2 + x_2x_3 + x_3x_1 = \dfrac{A}{2} \end{cases} \tag{7.1}$$

Exercice 2

$$P2 : \quad \begin{cases} Minimiser : & f(x_1, x_2) \\ SC & x_1 \geq 0, \\ & x_2 \geq 0 \end{cases} \tag{7.2}$$

où $f(x_1, x_2) = x_1^2 + x_2^2 + x_1x_2 - 3x_1$

Solution de l'exercice 1.

Soit la fonction de Lagrange :

$$L(x, y, \lambda_1, \lambda_2) = f(x, y) + \lambda_1 h_1(x, y) + \lambda_2 h_2(x, y)$$

$$= (x-2)^2 + 2(y-1)^2 + \lambda_1(x + 4y - 3) + \lambda_2(-x + y)$$

où $\lambda_1 \geq 0$ *et* $\lambda_1 \geq 0$

Calculant des dérivées partielles pour trouver le point optimal comme ci-dessous :

$$\frac{\partial L}{\partial x} = 2(x-2) + \lambda_1 - \lambda_2 = 0 \tag{7.3}$$

$$\frac{\partial L}{\partial y} = 4(y-1) + 4\lambda_1 - \lambda_2 = 0 \tag{7.4}$$

$$\lambda_1(x + 4y - 3) = 0 \tag{7.5}$$

$$\lambda_2(-x + y) = 0 \tag{7.6}$$

$$x + 4y \leq 3 \tag{7.7}$$

$$-x + y \leq 0 \tag{7.8}$$

$$\lambda_1 \geqslant 0 \tag{7.9}$$

$$\lambda_2 \geqslant 0 \tag{7.10}$$

Nous devons vérifier quatre cas possibles comme indiqué ci-dessous :

Cas 1 : Lorsque $\lambda_1 = 0$ *et* $\lambda_2 = 0$.

En utilisant les équations (1) *et* (2), *on obtient,* $x = 2$ *et* $y = 1$ *; ces valeurs de* x *et* y *ne satisfont pas les équations* (5) *et* (6).
Ce n'est donc pas une solution possible ou optimale.

Cas 2 : Lorsque la contrainte primale (A) est inactive et la contrainte primale (B) est active, c'est-à-dire, $\lambda_1 = 0$ *et* $-x + y = 0$.
Puis en remplaçant les valeurs ci-dessus dans les équations (1) et (2), on obtient :

$$2(x - 2) - \lambda_2 = 0$$

et

$$4(y - 1) + \lambda_2 = 0$$

En résolvant les deux équations ci-dessus, on obtient :

$$x = \frac{4}{3}, \quad y = \frac{4}{3}, \lambda_2 = -\frac{4}{3}$$

La valeur de λ_2 *ne satisfont pas les équations (7) et (8). Donc, ce n'est pas une solution possible ou optimale.*

Cas 3 : Lorsque la contrainte primale (A) est active et la contrainte primale (B) est inactive, c'est-à-dire, $x + 4y = 3$ *et* $\lambda_2 = 0$.
Puis en remplaçant les valeurs ci-dessus dans les équations (1) et (2), on obtient :

$$2(x - 2) + \lambda_1 = 0$$

et

$$4(y - 1) + 4\lambda_1 = 0$$

En résolvant les deux équations ci-dessus, on obtient :

$$x = \frac{5}{3}, \quad y = \frac{1}{3}, \lambda_1 = \frac{2}{3}$$

Les valeurs de x, y, λ_1, λ_2 satisfont les équations $(1) - (8)$. Donc, ce cas est l'une des solutions possibles ou optimales.

Cas 4 : Lorsque les deux contraintes (A) et (B) sont actives, c'est-à-dire, $x+4y = 3$ et $-x + y = 0$.

Puis en résolvant les deux équations ci-dessus, on obtient :

$$x = \frac{3}{5}, \quad y = \frac{3}{5}.$$

En remplaçant les valeurs ci-dessus dans les équations (1) et (2), on obtient : $\lambda_1 = \frac{22}{25}$, $\lambda_2 = -\frac{48}{25}$

La valeur de λ_2 ne satisfont pas l'équation (8).

: Ce n'est donc pas une solution possible ou optimale. D'où la seule solution possible à la fonction objectif f est le cas 3. Ainsi,

La solution optimale :

$$x^* = \frac{5}{3}, \quad y^* = \frac{1}{3}$$

La valeur optimal : $f(x^*, y^*) = (\frac{5}{3} - 2)^2 + 2(\frac{1}{3} - 1)^2$

$$f(x^*, y^*) = 1$$

Les valeurs de x, y, λ_1, λ_2 satisfont les équations $(1) - (8)$. Donc, ce cas est l'une des solutions possibles ou optimales.

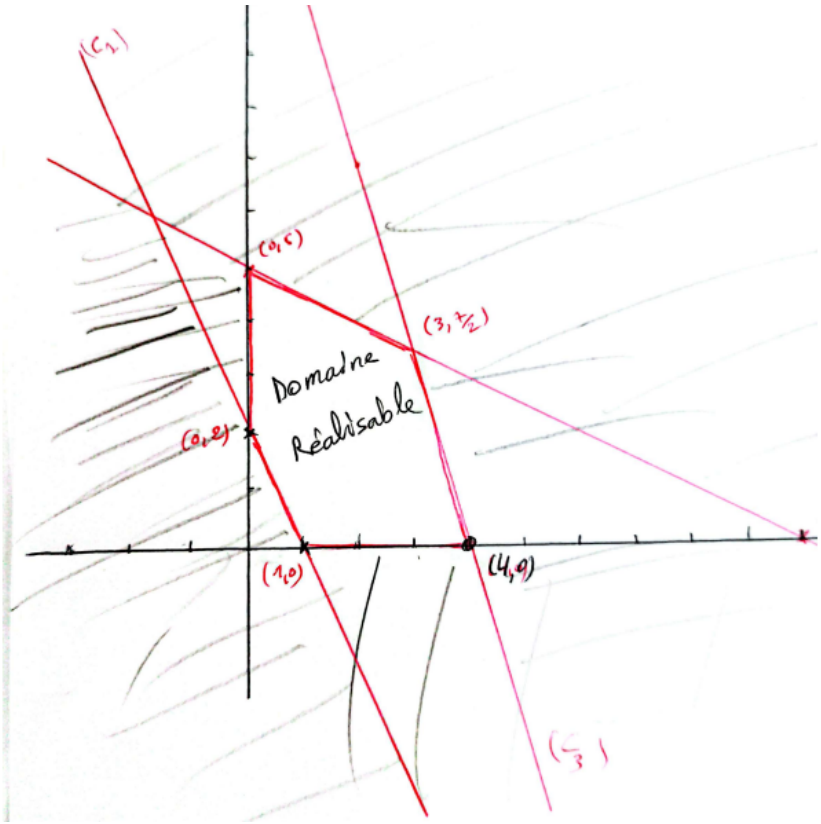
Solution de l'exercice 2

Considérons le problème d'optimisation avec contraintes suivant :

$$(P): \begin{cases} Maximiser(x_1^3 - 3x_2) \\ x_1 - x_2 + 2 \geq 0 \\ 2x_1 + x_2 - 2 \geq 0 \\ x_1 + 2x_2 - 10 \leq 0 \\ 7x_1 + 2x_2 - 28 \leq 0 \\ x_1 \quad et \quad x_2 \quad \geq 0 \end{cases}$$

Solution

(a) Déterminer géométriquement le domaine réalisable constitué par les contraintes.



(b) Résoudre le problème au moyen des conditions de Karush-Kuhn-Tucker (KKT). On a deux méthodes :

Méthode 1 :

Le problème doit écrit comme suit :

$$(P): \begin{cases} Maximiser(-x_1^3 + 3x_2) \\ -x_1 + x_2 - 2 \leq 0 \\ -2x_1 - x_2 + 2 \leq 0 \\ x_1 + 2x_2 - 10 \leq 0 \\ 7x_1 + 2x_2 - 28 \leq 0 \\ x_1 \quad et \quad x_2 \quad \geq 0 \end{cases}$$

Le Lagrangien s'écrit :

$$L(x,\lambda) = -x_1^3+3x_2+\lambda_1(-x_1+x_2-2)+\lambda_2(-2x_1-x_2+2)+\lambda_3(x_1+2x_2-10)+\lambda_4(7x_1+2x_2-28)$$

Les conditions de KKT sont donc les suivantes. Il y a tout d'abord les conditions de signe sur les variables :

$$x_1 \geq 0, \quad x_2 \geq 0, \quad \lambda_1 \geq 0, \quad \lambda_2 \geq 0, \quad \lambda_3 \geq 0, \quad \lambda_4 \geq 0$$

Il faut ensuite calculer les dérivées partielles par rapport à ces variables et poser les conditions de signe correspondantes.

$$3x_1^2 - 3x_2 + \lambda_1 + 2\lambda_2 - \lambda_3 - 7\lambda_4 = 0$$

$$-3 - \lambda_1 + \lambda_2 - 2\lambda_3 - 2\lambda_4 = 0$$

$$\lambda_1(-x_1 + x_2 - 2) = 0$$

$$\lambda_2(-2x_1 - x_2 + 2) = 0$$

$$\lambda_1(x_1 + 2x_2 - 10) = 0$$

$$\lambda_2(7x_1 + 2x_2 - 28) = 0$$

Si on discute tous les cas (qui sont 16 cas), ça devient très long et presque impossible manuellement.
Alors, en utilisant la présentation graphique de la question précédente.
Les point extrêmes du domaine réalisable sont :

$$(1,0), \quad (4,0) \quad (3,3.5) \quad (2,4) et (0,2)$$

On remplace dans la fonction objectif, on obtient : $F(1,0) = 1$,
$F(4,0) = 64$, *optimale*

$F(3, 3.5) = 16.5,$
$F(2, 4) = -4,$
$F(0, 2) = -6$

Alors, en appliquant KKT pour vérifier et confirmer la solution optimale : pour le point (4,0), la dernière contrainte est active et les autres contraintes sont non actives, donc on a :

$$\lambda_1 = \lambda_2 = \lambda_3 = 0, \quad 7x_1 + 2x_2 - 28 = 0 \quad \lambda_4 \geq 0$$

on trouve, $\lambda_4 = \frac{48}{7} positif,$

Alors l'optimum est :

$$x_1 = 4 \quad x_2 = 0, \quad \lambda_1 = \lambda_2 = \lambda_3 = 0, \quad \lambda_4 = \frac{48}{7}$$

Méthode 2 :

Les deux premières contraintes doivent être réécrites sous la forme :

$$-x_1 + x_2 - 2 \leq 0$$

$$-2x_1 - x_2 + 2 \leq 0$$

Le lagrangien s'écrit de la manière suivante :

$$L(x, \lambda) = x_1^3 - 3x_2 - \lambda_1(-x_1 + x_2 - 2) - \lambda_2(-2x_1 - x_2 + 2) - \lambda_3(x_1 + 2x_2 - 10) - \lambda_4(7x_1 + 2x_2 - 28)$$

Les conditions de KKT sont donc les suivantes. Il y a tout d'abord les conditions de signe sur les variables :

$$x_1 \geq 0, \quad x_2 \geq 0, \quad \lambda_1 \geq 0, \quad \lambda_2 \geq 0, \quad \lambda_3 \geq 0, \quad \lambda_4 \geq 0$$

Il faut ensuite calculer les dérivées partielles par rapport à ces variables et poser les conditions de signe correspondantes. Dans le cas des dérivées par rapport aux coefficients λ*, on retrouve les contraintes :*

$$3x_1^2 - 3x_2 + \lambda_1 + 2\lambda_2 - \lambda_3 - 7\lambda_4 = 0$$

$$-3 - \lambda_1 + \lambda_2 - 2\lambda_3 - 2\lambda_4 = 0$$

$$\lambda_1(-x_1 + x_2 - 2) = 0$$

$$\lambda_2(-2x_1 - x_2 + 2) = 0$$

$$\lambda_1(x_1 + 2x_2 - 10) = 0$$

$$\lambda_2(7x_1 + 2x_2 - 28) = 0$$

Ce sont ces dernières conditions qui servent à mener la discussion car elles présentent une alternative. L'un des deux termes du produit doit être nul. On discute donc en testant les deux possibilités.

On cherche à maximiser la fonction objectif : $f = x_1^3 - 3x_2$. Intuitivement, on voit qu'il faut choisir x_1 le plus grand possible et x_2 le plus petit possible. On va donc chercher $x_2 = 0$ (les variables doivent être positives).
On est conduit, en remplaçant dans les conditions à $x_1 = 4$, puis $= 0$ et $\lambda_1 = \frac{48}{7}$. C'est l'optimum du problème et on peut vérifier qu'il remplit toutes les conditions de KKT.

Quiz : Méthode du Gradient

Partie A : Questions à choix multiple

(a) Le gradient d'une fonction pointe dans la direction de :

A) La décroissance la plus rapide

B) La croissance la plus lente

C) La croissance la plus rapide

D) Aucune variation

Réponse : C

(b) La formule de mise à jour de la méthode de descente de gradient est :

A) $x_{k+1} = x_k + \alpha \nabla f(x_k)$

B) $x_{k+1} = x_k - \alpha \nabla f(x_k)$

C) $x_{k+1} = \nabla f(x_k)$

D) $x_{k+1} = \frac{x_k}{\alpha}$

Réponse : B

(c) Le paramètre α est appelé :

A) Moment

B) Hessienne

C) Pas de descente (taux d'apprentissage)

D) Direction de recherche

Réponse : C

(d) Si le pas de descente est trop grand, l'algorithme peut :

A) Converger plus rapidement

B) Osciller ou diverger

C) Toujours trouver le minimum

D) S'arrêter immédiatement

Réponse : B

Partie B : Questions à réponse courte

(a) Quelle condition caractérise un point stationnaire ?

Réponse :

$$\nabla f(x) = 0$$

(b) Pourquoi la méthode du gradient est-elle une méthode d'optimisation locale ?

Réponse : *Parce que la convergence dépend du point initial et peut mener à un minimum local plutôt qu'au minimum global.*

(c) Donner un avantage et un inconvénient de la méthode du gradient.

Réponse :

— *Avantage : méthode simple et peu coûteuse en calcul.*

— *Inconvénient : sensible au choix du pas et peut rester bloquée dans des minima locaux.*

Partie C : Vrai ou Faux

(a) La méthode de descente de gradient trouve toujours le minimum global. ***Réponse : Faux***

(b) Le gradient est un vecteur de dérivées partielles. ***Réponse : Vrai***

(c) La méthode de montée de gradient est utilisée pour les problèmes de maximisation. ***Réponse : Vrai***

Partie A : Questions à choix multiple

(a) La méthode du gradient conjugué est principalement utilisée pour résoudre :

A) Des problèmes non linéaires quelconques

B) Des systèmes linéaires $Ax = b$ avec A symétrique définie positive

C) Des systèmes surdéterminés

D) Des problèmes de maximisation non contraints

Réponse : B

(b) Par rapport à la descente de gradient classique, la méthode du gradient conjugué :

A) Utilise uniquement la direction du gradient

B) Converge en un nombre fini d'itérations (en arithmétique exacte)

C) Nécessite le calcul de la matrice Hessienne

D) Est plus lente

Réponse : B

(c) Les directions de recherche dans la méthode du gradient conjugué sont :

A) Orthogonales

B) Aléatoires

C) Conjuguées par rapport à la matrice A

D) Identiques au gradient

Réponse : C

(d) Le paramètre β_k dans la méthode du gradient conjugué permet de :

A) Choisir le pas optimal

B) Construire la nouvelle direction de recherche

C) Calculer le résidu initial

D) Tester la convergence

Réponse : B

Partie B : Questions à réponse courte

(a) Écrire la fonction objectif équivalente au système linéaire $Ax = b$.

Réponse :

$$f(x) = \frac{1}{2}x^T Ax - b^T x$$

(b) *Quelle est la condition sur la matrice A pour garantir la convergence de la méthode du gradient conjugué ?*

Réponse : *La matrice A doit être* ***symétrique définie positive.***

(c) *Donner une différence essentielle entre la méthode du gradient conjugué et la descente de gradient.*

Réponse : *La méthode du gradient conjugué utilise des directions de recherche conjuguées et converge en un nombre fini d'itérations (en arithmétique exacte), contrairement à la descente de gradient classique.*

Partie C : Vrai ou Faux

(a) *La méthode du gradient conjugué nécessite un pas de descente fixé à l'avance.* ***Réponse : Faux***

(b) *La méthode du gradient conjugué est une méthode de type Krylov.* ***Réponse : Vrai***

(c) *En arithmétique exacte, la méthode du gradient conjugué converge en au plus n itérations pour un problème de dimension n.* ***Réponse : Vrai***

Quiz : Méthode de Newton

Partie A : Questions à choix multiple

(a) *La méthode de Newton est principalement utilisée pour :*

A) Résoudre des systèmes linéaires

B) Trouver les racines d'une fonction ou les points critiques

C) Calculer des intégrales

D) Résoudre uniquement des problèmes convexes

Réponse : B

(b) *La formule d'itération de la méthode de Newton pour l'optimisation est :*

A) $x_{k+1} = x_k - \alpha \nabla f(x_k)$

B) $x_{k+1} = x_k - [\nabla^2 f(x_k)]^{-1} \nabla f(x_k)$

C) $x_{k+1} = \nabla f(x_k)$

D) $x_{k+1} = x_k + \nabla^2 f(x_k)$

Réponse : B

(c) *La matrice $\nabla^2 f(x)$ représente :*

A) Le gradient

B) Le jacobien

C) La matrice Hessienne

D) La matrice identité

Réponse : C

(d) *Par rapport à la descente de gradient, la méthode de Newton :*

A) Converge plus lentement

B) Ne dépend pas du point initial

C) A une convergence plus rapide près de la solution

D) Ne nécessite pas de dérivées

Réponse : C

Partie B : Questions à réponse courte

(a) *Quelle est l'hypothèse principale sur la matrice Hessienne pour appliquer la méthode de Newton en optimisation ?*

Réponse : *La matrice Hessienne doit être inversible et, pour un minimum local, définie positive au voisinage de la solution.*

(b) *Donner un avantage et un inconvénient de la méthode de Newton.*

Réponse :

- *Avantage : convergence rapide (quadratique) près de la solution.*
- *Inconvénient : coût de calcul élevé et nécessité de calculer et inverser la Hessienne.*

(c) *Pourquoi la méthode de Newton peut-elle diverger ?*

Réponse : *Elle peut diverger si le point initial est mal choisi ou si la matrice Hessienne n'est pas définie positive.*

Partie C : Vrai ou Faux

(a) *La méthode de Newton a une convergence quadratique sous certaines hypothèses.* ***Réponse : Vrai***

(b) *La méthode de Newton ne nécessite que le gradient de la fonction objectif.* ***Réponse : Faux***

(c) *La méthode de Newton est une méthode locale.* ***Réponse : Vrai***

Quiz : Méthode de Relaxation

Partie A : Questions à choix multiple

(a) Les méthodes de relaxation sont principalement utilisées pour :

A) Résoudre des équations différentielles

B) Résoudre des systèmes linéaires $Ax = b$

C) Calculer des valeurs propres

D) Optimiser des fonctions non linéaires

Réponse : B

(b) La méthode de Jacobi se caractérise par :

A) L'utilisation immédiate des nouvelles valeurs calculées

B) L'utilisation uniquement des valeurs de l'itération précédente

C) Un paramètre de relaxation ω

D) Une convergence en une seule itération

Réponse : B

(c) La méthode de Gauss-Seidel :

A) Est toujours plus lente que Jacobi

B) Utilise partiellement les valeurs mises à jour

C) Nécessite une matrice symétrique définie positive

D) Est une méthode directe

Réponse : B

(d) Dans la méthode de relaxation successive (SOR), le paramètre ω vérifie :

A) $\omega = 1$

B) $\omega < 0$

C) $0 < \omega < 2$

D) $\omega > 2$

Réponse : C

Partie B : Questions à réponse courte

(a) Écrire la formule générale de mise à jour de la méthode de relaxation SOR.

Réponse :

$$x_i^{(k+1)} = (1-\omega)x_i^{(k)} + \frac{\omega}{a_{ii}}\left(b_i - \sum_{j<i} a_{ij}x_j^{(k+1)} - \sum_{j>i} a_{ij}x_j^{(k)}\right)$$

(b) *Donner une condition suffisante de convergence des méthodes de relaxation.*

Réponse : *La matrice A est strictement diagonale dominante (ou symétrique définie positive pour Gauss-Seidel).*

(c) *Comparer brièvement Jacobi et Gauss-Seidel.*

Réponse : *La méthode de Gauss-Seidel converge généralement plus vite que Jacobi car elle utilise les valeurs mises à jour au cours de l'itération.*

Partie C : Vrai ou Faux

(a) *La méthode de Jacobi est une méthode itérative.* ***Réponse : Vrai***

(b) *La méthode SOR se réduit à Gauss-Seidel pour $\omega = 1$.* ***Réponse : Vrai***

(c) *Les méthodes de relaxation sont des méthodes directes.* ***Réponse : Faux***

Bibliographie

[1] *J. Nocedal, S. Wright, Numerical Optimization, Springer. 2006.*

[2] *D. P. Bertsekas, Nonlinear Programming. Journal of the Operational Research Society (1997) 48, 332-334.*

[3] *S. Boyd, L. Vandenberghe, Convex Optimization.*

[4] *Bastien, J., and Martin, J. N. Introduction à l'analyse numérique : applications sous Matlab : cours et exercices corrigés. Dunod. 2003.*

[5] *M. S. Bazaraa, H. D. Sherali, and C. M. Shetty, Nonlinear Programming : Theory and Algorithms, 2nd ed. New York : Wiley, 1993.*

[6] *J. E. Dennis, Jr. and R. B. Schnabel, Numerical Methods for Unconstrained Optimization and Nonlinear Equations. Englewood Cliffs, NJ : Prentice Hall, 1983.*

[7] *R. Fletcher, Practical Methods of Optimization, 2nd ed. Chichester, England : Wiley, 1987.*

[8] *P. E. Gill, W. Murray, and M. H. Wright, Practical Optimization. London : Academic Press, 1981.*

[9] *D. G. Luenberger, Optimization by Vector Space Methods. New York : Wiley, 1969.*

[10] *D. G. Luenberger and Y. Ye, Linear and Nonlinear Programming, 3rd ed. New York, NY : Springer Science + Business Media, 2008.*

[11] *A. L. Peressini, F. E. Sullivan, and J. J. Uhl, Jr., The Mathematics of Nonlinear Programming. New York : Springer-Verlag, 1988.*

[12] *P. P. Varaiya, Notes on Optimization. New York : Van Nostrand Reinhold Co., 1972.*

[13] *J. S. Arora, Introduction to Optimum Design. New York : McGraw-Hill Book Co., 1989.*

[14] *G. V. Reklaitis, A. Ravindran, and K. M. Ragsdell, Engineering Optimization : Methods and Applications. New York : Wiley-Interscience, 1983.*

[15] J. Nocedal and S. J. Wright, Numerical Optimization*, 2nd Edition, Springer, 2006.*

[16] R. Fletcher, Practical Methods of Optimization*, 2nd Edition, Wiley, 1987.*