Algorithmes d’optimisation

Pr. Faouzia Benabbou (faouzia.benabbou@univh2c.ma)
Département de mathématiques et Informatique

Master Data Science & Big Data

2024-2025

Plan du Module: Algorithmes d’optimisation

01 Introduction

02 Rappels mathématiques

03 Algorithmes d’Optimisation Sans Contraintes

04 Algorithmes d’Optimisation avec Contraintes

05 Programmation Linéaire et non Linéaire

06 Optimisation combinatoire et globale

07 Méthodes heuristiques et métaheuristiques

Les algorithmes d’optimisation

Classes des algorithmes

- Optimisation
  - Sans contraintes
    - Méthodes Analytiques
    - Méthodes numériques
      - Méthode sans gradient
      - Méthode de descente de gradient:
        - ✓ Méthode de gradient à pas fixe et optimal
        - ✓ Méthode de gradient conjugué
        - ✓ Méthode de Newton
        - ✓ Méthodes du gradient quasi-newton
  - Sous contrainte
    - Méthodes générales de résolutions de PCE, PCI
    - Programmation linéaire et non linéaire
    - Optimisation combinatoire et globale
    - Méthodes heuristiques et métaheuristiques
    - Algorithmes multi-objectifs

Méthodes d’optimisation Sous contraintes

- **Méthode des multiplicateurs de Lagrange**
  - La méthode des multiplicateurs de Lagrange est une technique d'optimisation utilisée pour trouver les extrema locaux (maxima ou minima) d'une fonction sous contraintes d'égalité (étendue à inégalité).
  - Elle est particulièrement utile lorsque les contraintes rendent difficile l'expression d'une variable en fonction des autres.
  - L'idée centrale de la méthode est de transformer un problème d'optimisation **contraint** en un problème **non contraint** en introduisant une nouvelle fonction, appelée **fonction de Lagrange**, qui combine la fonction objective et les contraintes.

Méthodes d’optimisation Sous contraintes

- **Méthode des multiplicateurs de Lagrange**
  - Soit le problème PCE suivant :

$$\begin{cases} \min\limits_{x\in\mathbb{R}^n} f(\mathrm{x})\,, f \; une\; fonction\; objective\; de\,. \\ \mathrm{h(x)} = 0\,. \end{cases}$$

Définition. La **fonction de Lagrange** associée à un problème d’optimisation sous contrainte d’égalité est définie par :

$$L(\mathrm{x},\lambda) = \mathrm{f(x)} + \lambda^{\mathrm{T}}\mathrm{h(x)}$$

$= \mathrm{f(x)} + \sum_{i=1}^{i=p} \lambda_i h_i(\mathrm{x})$ où $(\lambda_1,\ldots,\lambda_p)$ sont les multiplicateurs de Lagrange **associés aux contraintes d’égalité**.

Méthodes d’optimisation Sous contraintes

- **Méthode des multiplicateurs de Lagrange**
  - Soient h= $h_1 h_2$ ... $h_p : \mathbb{R}^n \rightarrow \mathbb{R}$ des fonctions de classe C 1 , avec p ∈ IN* .
  - L’ensemble S = {x ∈ $\mathbb{R}^n$, h_i(x) = 0, ∀i = 1, 2, · · · p} est variété de $\mathbb{R}^n$.

Définition.

Si x ∈ S est tel que la famille des vecteurs {∇ h_i(x) i=1,···p} forme un système libre en $\mathbb{R}^n$ alors on dit que x est un **point régulier** de S.

La variété S est dite **régulière** si tous les points de S sont réguliers.

- Une condition équivalente de la régularité de S en x est que : rang(J_h)=p, où J_h est la matrice Jacobienne de h.

Méthodes d'optimisation Sous contraintes

- **Méthode des multiplicateurs de Lagrange**

Théorème de Lagrange. Condition nécessaire d'optimalité du premier ordre.

Soit f et h différentiable en x*. Si x* est une solution **optimale locale** et si les gradients des contraintes **$\nabla h_i(x^*)$ sont linéairement indépendants**, alors il existe un unique vecteur **λ^*** $\in \mathbb{R}^p$ tel que :

$$\begin{cases} \nabla L(x*, \lambda) = \nabla f(x*) + \nabla h(x*)^T \lambda = 0, & \text{Condition de } \textbf{stationnarité}. \\ h(x*) = 0, & \textit{Condition de } \textbf{faisabilité} . \end{cases}$$

- On appellera x* vérifiant ces deux conditions un point stationnaire.
- Le vecteur $\lambda * = \{\lambda_1^*, .., \lambda_p^*\}$est appelé multiplicateur de Lagrange.
- ∇f(x*) est le gradient de la fonction objectif évaluée en x*.
- ∇h(x*) est la matrice **jacobienne** des contraintes évaluées en x*.

Méthodes d’optimisation Sous contraintes

- **Méthode des multiplicateurs de Lagrange**
  - La première condition signifie que le gradient de f(x) est **combiné linéairement** avec les gradients des contraintes.
  - Les conditions de **premier ordre** sont **nécessaires mais pas suffisantes** pour garantir un minimum local.
  - Il faut analyser la courbure de f(x) via la Hessienne de la fonction Lagrange.

$$H_L(x*,\lambda*) = \nabla^2 f(x*) + \nabla^2 \boldsymbol{h}(x*)\, \lambda*$$

  - Pour cela on définit l’**espace tangent des contraintes** :

$$T=\{d\in\mathbb{R}^n,\ \nabla h_i(x*)^T d=0, \forall i\}.$$

Méthodes d'optimisation Sous contraintes

- **Méthode des multiplicateurs de Lagrange**

Théorème. La **condition suffisante du second ordre** est réalisé si x* satisfait:

- Les conditions de premier ordre.
- $H_L(x*, \lambda*)$ est **définie positive** sur **T :** $d^T H_L(x*, \lambda*) d > 0$, $\forall d \in T \setminus \{0\}$.
- alors $x*$ est **un minimum local strict**

Méthodes d'optimisation Sous contraintes

- **Méthode des multiplicateurs de Lagrange**
  - Cas de contraintes **d'égalité linéaires** est définie par:

$$L(x,\lambda)=f(x) + \lambda^T(Ax - b).$$

Les conditions de **premier ordre** donnent :

$$\nabla_x L(x,\lambda)=\nabla f(x) + \nabla(\lambda^T(Ax - b)) = 0.$$

Sachant que L'expression $\boldsymbol{\nabla} (\lambda^T(\mathbf{Ax - b})) \mathbf{=} \mathbf{A^T} \lambda$,

on obtient :

$$\nabla_x L(x,\lambda)=0 \rightarrow \nabla f(x) + \mathbf{A^T}\lambda=0$$

Méthodes d’optimisation Sous contraintes

- **Méthode des multiplicateurs de Lagrange**

Théorème. Si $x*$ est une solution optimale (minimum) du problème d'optimisation sous contrainte **d'égalité linéaire**, alors il existe nécessairement un vecteur $\lambda \in \mathbb{R}^p$ vérifiant :

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

Si de plus A est de rang p (c'est-à-dire que les contraintes sont linéairement indépendantes) alors **λ** est **unique**.

Méthodes d'optimisation Sous contraintes

- **Méthode des multiplicateurs de Lagrange**

Algorithme 8 : Méthode des multiplicateurs de Lagrange

1) Initialisation : $f(x)$: $fonction\ objective$; g(x) : Contraintes.

2) Construire la fonction de Lagrange : $L(x,\lambda) = f(x) + \lambda^T h(x)$

3) Calculer le gradient de la Lagrangienne :

$$L(x,\lambda) = \nabla f(x) + \nabla h(x)^T \lambda$$

4) **Résoudre le système d'équations non linéaires** en x et λ:

$$\begin{cases} \nabla L(x,\lambda) = 0, & \textbf{Condition de stationnarité}. \\ h(x) = 0, & \textbf{Condition de faisabilité}. \end{cases}$$

5) **Vérifier la nature du point** en examinant la matrice Hessienne: $H_L(x,\lambda)$

a)si $H_L(x,\lambda)$ est **définie positive sur T**, on a un **minimum local**

b)Si elle est définie négative, on a un maximum local.

c) Sinon, on ne peut pas conclure

Méthodes d'optimisation Sous contraintes

- **Méthode des multiplicateurs de Lagrange**

Exemple 1. On cherche à minimiser la fonction :
minf(x,y)=x^2+3y^2 sous la contrainte :x+2y=4.
Calcule de la fonction de Lagrange : L(x,y,λ)= $x^2+3y^2+\lambda(x+2y-4)$.

$condition\ Premier\ ordre$: $\frac{\partial L(x,y,\lambda)}{\partial x}=0$ et $\frac{\partial L(x,y,\lambda)}{\partial y}=0$; 2x+λ=0 et 6y+2λ=0

Donc on a le système suivant : $\begin{cases} \mathbf{2x + \lambda = 0} \text{ et } \mathbf{6y + 2\lambda} = 0. \\ \mathrm{x + 2y - 4 = 0} . \end{cases}$

Après substitution on trouve : $\begin{cases} x = 12/7 \\ y = 8/7 \\ \boldsymbol{\lambda = -24/7} \end{cases}$

(12/7, 8/7) est un point critique.

Méthodes d'optimisation Sous contraintes

▪ Méthode des multiplicateurs de Lagrange

Exemple 1. On cherche à minimiser la fonction : $f(x,y)=x^2+3y^2$ sous la contrainte $x+2y=4$.

Gradient $\frac{\partial L(x,y,\lambda)}{\partial x} = \mathbf{2x + \lambda}$ et $\frac{\partial L(x,y,\lambda)}{\partial y} = \mathbf{6y + 2\lambda}$

$H_L(x,y,\lambda)=\begin{pmatrix} 2 & 0 \\ 0 & 6 \end{pmatrix}$, calculons sa quadratique : $(x,y)\begin{pmatrix} 2 & 0 \\ 0 & 6 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} = (x,y)\begin{pmatrix} 2x1 \\ 6x2 \end{pmatrix} = 2x^2+6y^2$.

Trouvons l'espace tangent $T=\{d\in\mathbb{R}^n,\ \nabla h_i(x*)^T d=0, \forall i\}$,

$\nabla h(x,y)=\begin{pmatrix} 1 \\ 2 \end{pmatrix}$, $\nabla h_i(x,y)^T d = (1\ 2)\begin{pmatrix} d1 \\ d2 \end{pmatrix} = d1+2d2=0$, donc $d1=-2d2$ $d=\begin{pmatrix} -2d2 \\ d2 \end{pmatrix} = d2\begin{pmatrix} -2 \\ 1 \end{pmatrix}$

Montrons que $d^T H_L(x, \lambda*) d > 0$,

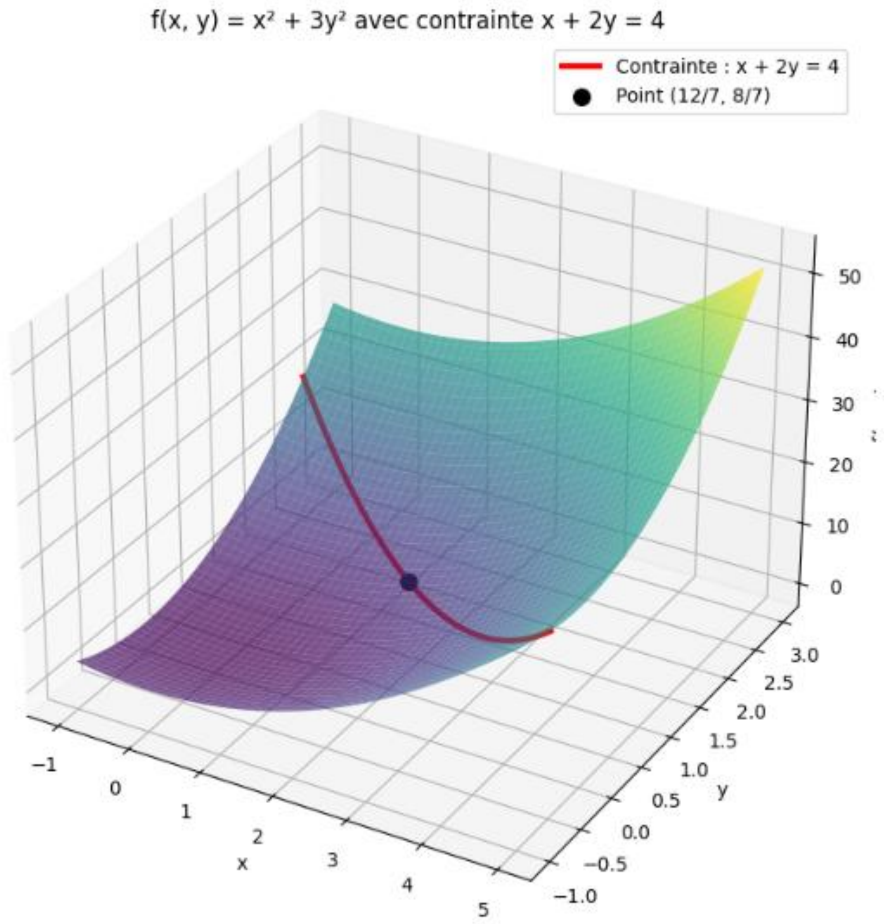
$(d1,d2)\begin{pmatrix} 2 & 0 \\ 0 & 6 \end{pmatrix}\begin{pmatrix} d1 \\ d2 \end{pmatrix} = 2d1^2 + 6d2^2 = 14d2^2 > 0$ car $d\neq 0$ donc $H_L(x,y,\lambda)$ est Définie positive .

La solution (12/7,8/7) est un minimum locale et $f(x^*,y^*)=48/7$.

L'unique multiplicateur de Lagrange est $\boldsymbol{\lambda = -\frac{24}{7}}$.

Méthodes d'optimisation Sous contraintes

- **Méthode des multiplicateurs de Lagrange**
- **Exemple 1.** On cherche à minimiser la fonction : $f(x,y)=x^2+3y^2$ sous la contrainte $x+2y=4$.



Méthodes d'optimisation Sous contraintes

- **Méthode des multiplicateurs de Lagrange**
- **Exemple 2.** Trouver x minimisant f(x)= $x_1 x_2$ sous h(x)= $x_1 - 2x_2 + 4$
- Le PS devient :

$$\begin{cases} \text{f(x)} = x_1 x_2 \\ x_1 - 2x_2 + 4 = 0\,. \end{cases}$$

Calcule de la fonction de Lagrange : L(x,y,λ)= x_1x_2 +λ($x_1 - 2x_2 + 4$).

- Les conditions nécessaires d'optimalité du premier ordre sont :
- $$\begin{cases} \frac{\partial L(x_1,x_2,\lambda)}{\partial x_1} = 0 \\ \frac{\partial L(x_1,x_2,\lambda)}{\partial x_2} = 0 \\ x_1 - 2x_2 + 4 = 0 \end{cases} \rightarrow \begin{cases} x_1 - 2\lambda = 0 \\ x_2 + \lambda = 0 \\ x_1 - 2x_2 + 4 = 0 \end{cases} \rightarrow \begin{cases} x_1 = 2\lambda \\ x_2 = -\lambda \\ 2\lambda + 2\lambda + 4 = 0 \end{cases} \rightarrow \begin{cases} x_1 = -2 \\ x_2 = 1 \\ \lambda = -1 \end{cases}$$

(-2, 1) est un point critique.

Méthodes d'optimisation Sous contraintes

- **Méthode des multiplicateurs de Lagrange**
  - **Exemple 2.** Trouver x minimisant f(x)= x_1x_2 sous h(x)= $x_1 - 2x_2 + 4$

Le PS devient :

$$\begin{cases} f(x) = x_1x_2 \\ x_1 - 2x_2 + 4 = 0\,. \end{cases}$$

Calcule de la fonction de Lagrange : L(x,y,λ)= x_1x_2 +λ($x_1 - 2x_2 + 4$).

Condition suffisante du second degré :

$H_L(x_1, x_2$,λ) est **définie positive ?**

$H_L(x_1, x_2, \lambda)$=$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$, calculons sa quadratique : $(x_1, x_2)\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$ =$(x_1$,$x_2)\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$=$2x_1x_2 \geq 0$?

Trouvons l'espace tangent T={d∈$\mathbb{R}^n$, ∇h_i(x∗)^{T}d=0,∀i}, d=(d1,d2)

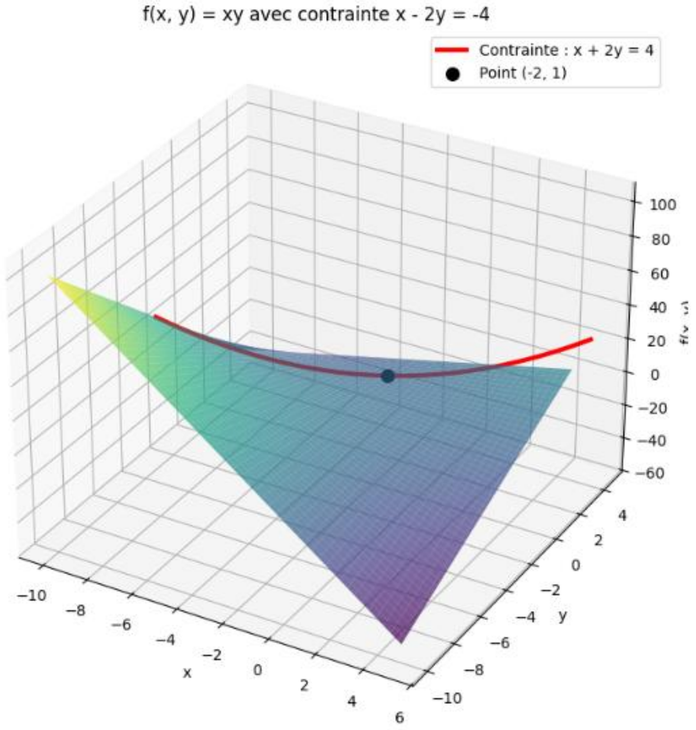
∇h (x,y)=$\begin{pmatrix} 1 \\ -2 \end{pmatrix}$, (1 -2) $\begin{pmatrix} d1 \\ d2 \end{pmatrix}$=d1-2d2=0, donc d1=2d2 d=$\begin{pmatrix} 2d2 \\ d2 \end{pmatrix}$=$d2\begin{pmatrix} 2 \\ 1 \end{pmatrix}$

On remplace dans la forme quadratique on obtient : $4d2^2$ donc $H_L(x_1, x_2, \lambda)$ est Définie Positive.

Le minimum local est (-2, 1) et f(-2, 1)=-2.

Méthodes d'optimisation Sous contraintes

- **Méthode des multiplicateurs de Lagrange**
- **Exemple 2.** Trouver x minimisant f(x)= $x_1 x_2$ sous h(x)= $x_1 - 2x_2 + 4$



```python
import sympy as sp
#Méthode des multiplicateurs de Lagrange ex 1

# Variables
x1, x2, lam = sp.symbols('x1 x2 lambda')
d2 = sp.symbols('d2')  # direction d2 (d1 = 2 * d2)

# Fonction objectif et contrainte
f = x1 * x2
h = x1 - 2 * x2 + 4

# Lagrangienne (convention f + lambda * h)
L = f + lam * h

# Gradient de L
grad_L = [sp.diff(L, var) for var in (x1, x2, lam)]

# Résolution du système stationnaire
solution = sp.solve(grad_L, (x1, x2, lam), dict=True)[0]
x1_sol, x2_sol, lam_sol = solution[x1], solution[x2], solution[lam]

print("Point critique trouvé :")
print(f"x1 = {x1_sol}, x2 = {x2_sol}, lambda = {lam_sol}")

# Hessienne de L par rapport à x uniquement
H_L = sp.hessian(L, (x1, x2))

# Affichage de la Hessienne
print("\nHessienne du Lagrangien L :")
sp.pprint(H_L)

# Substitution des valeurs du point critique dans la Hessienne
H_L_at_sol = H_L.subs({x1: x1_sol, x2: x2_sol, lam: lam_sol})

# Affichage de la Hessienne au point critique
print("\nHessienne au point critique :")
sp.pprint(H_L_at_sol)

# Direction tangentielle : grad_h^T * d = 0
# Donc : (1, -2)·(d1, d2) = 0 → d1 - 2 d2 = 0 → d1 = 2 d2
# On pose d = (2*d2, d2)

d_vec = sp.Matrix([2 * d2, d2])

# Forme quadratique q(d) = d^T H_L d
q_d = (d_vec.T * H_L_at_sol * d_vec)[0]

q_d_simplified = sp.simplify(q_d)

print("\nForme quadratique sur l'espace tangent en fonction de d2 :")
sp.pprint(q_d_simplified)
```

tion Sous contraintes

licateurs de Lagrange

er x minimisant f(x)= $x_1 x_2$ sous

```
Point critique trouvé :
x1 = -2, x2 = 1, lambda = -1

Hessienne du Lagrangien L :
⎡0  1⎤
⎢    ⎥
⎣1  0⎦

Hessienne au point critique :
⎡0  1⎤
⎢    ⎥
⎣1  0⎦

Forme quadratique sur l'espace tangent en fonction de d2 :
    2
4·d₂
```

Méthodes d’optimisation Sous contraintes

- **Méthode des multiplicateurs de Lagrange**
- **Exemple 2.** Trouver x minimisant f(x)= $x_1 x_2$ sous h(x)= $x_1 - 2x_2 + 4$

```python
# Remplacer d2 par 1
q_d_numeric = q_d_simplified.subs(d2, 1)

print("\nForme quadratique après substitution de d2 = 1 :")
sp.pprint(q_d_numeric)

# Vérification du signe de q(d) numériquement
if q_d_numeric > 0:
    print(f"\n • Point minimum : ({x1_sol}, {x2_sol})")
elif q_d_numeric < 0:
    print(f"\n • Point rmaximum : x1 = {x1_sol}, x2 = {x2_sol}")
else:
    print("\n • Point selle")
```

```
Point critique trouvé :
x1 = -2, x2 = 1, lambda = -1

Hessienne du Lagrangien L :
⎡0  1⎤
⎢    ⎥
⎣1  0⎦

Hessienne au point critique :
⎡0  1⎤
⎢    ⎥
⎣1  0⎦

Forme quadratique sur l'espace tangent en fonction de d2 :
    2
4·d₂

Forme quadratique après substitution de d2 = 1 :
4

 • Point minimum : (-2, 1)
```

Méthodes d'optimisation Sous contraintes

- **Méthode des multiplicateurs de Lagrange**
- **Exemple 3.** Trouver x minimisant:

f(x)= $3x_1 + 5x_2$ sous h(x)= $x_1 + x_2 - 10$

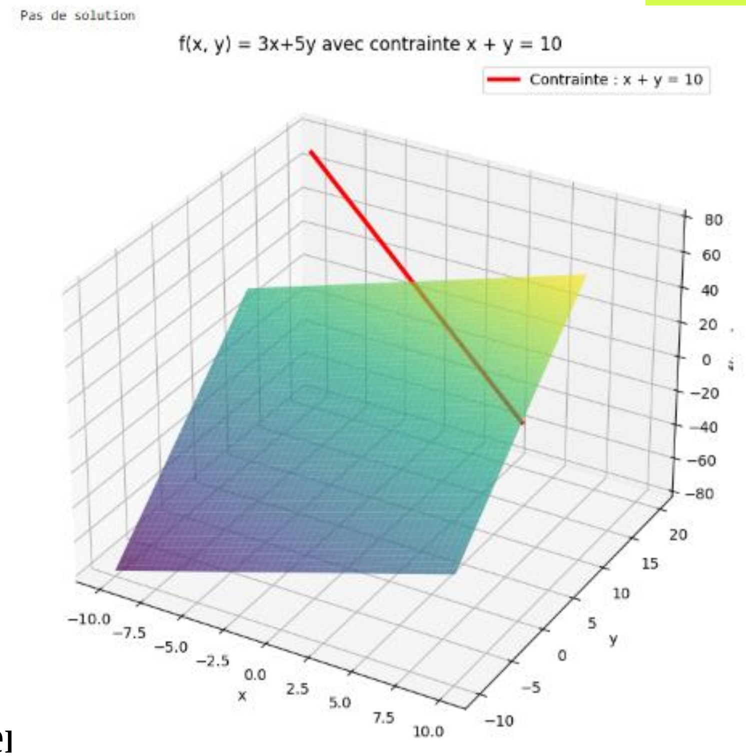
L(x_1,x_2, λ) = $3x_1 + 5x_2$ + λ($x_1 + x_2 - 10$)

Les conditions nécessaires d'optimalité du premier ordre sont :

$$\begin{cases} \frac{\partial L(x_1,x_2,\lambda)}{\partial x_1} = 3 \\ \frac{\partial L(x_1,x_2,\lambda)}{\partial x_2} = 5 \\ x_1 + x_2 - 10 = 0 \end{cases} \rightarrow \begin{cases} 3 + \lambda = 0 \\ 5 + \lambda = 0 \\ x_1 + x_2 - 10 = 0 \end{cases}$$

$$\begin{cases} \lambda = -3 \\ \lambda = -\mathbf{5} \\ 2\lambda + 2\lambda + 4 = 0 \end{cases}$$

- pas de solution, car ces deux droites sont parallèles, elles ne se rencontrent pas, donc il n'y a pas de point qui satisfait simultanémer la contrainte et le maximum ou minimum de la fonction objective.



Méthodes d’optimisation Sous contraintes

- **Méthode des multiplicateurs de Lagrange**
  - La méthode des multiplicateurs de Lagrange identifie les points stationnaires du Lagrangien, qui sont des candidats pour les extrema locaux (minima, maxima ou points de selle).
  - Elle ne garantit pas que la solution trouvée soit un extremum global
  - La méthode fonctionne si les gradients de contraintes sont linéairement indépendants au point optimal (on parle de régularité, qualifications de contraintes).
  - La méthode repose sur le calcul des dérivées partielles des fonctions objective et de contrainte.
  - Elle **ne gère pas directement les inégalités** $g(x) \leq 0$, pour cela, il faut utiliser la **méthode KKT (Karush-Kuhn-Tucker)**.
  - Pour les problèmes avec un grand nombre de variables et de contraintes, la mise en place et la résolution du système d'équations peuvent devenir très coûteuses en termes de calcul.

Méthodes d’optimisation Sous contraintes

Méthodes du gradient projeté

- Les méthodes du gradient projeté sont des algorithmes d'optimisation utilisés pour résoudre des problèmes où l'on cherche à minimiser une fonction sous certaines contraintes.
- Elles combinent une descente de gradient classique avec une projection sur l'ensemble **admissible** pour garantir que les itérés restent dans le domaine de contraintes.
- Dans les méthodes de descente de gradient la solution est améliorée à chaque itération par : $\boldsymbol{x_{k+1}} = \boldsymbol{x_k} + \boldsymbol{\alpha_k}\ \mathbf{d_k}$ où d_k est la direction choisie de sorte que : $f(x_k + \alpha_k\ d_k) < f(x_k)$, dans un problème de minimisation.
- Le problème est que si $x_k \in S$, ensemble des point admissibles, rien ne garantit que $x_{k+1} \in S$.

Méthodes d’optimisation Sous contraintes

- **Méthodes du gradient projeté**
  - L'idée centrale des méthodes du gradient projeté est d'adapter la méthode de descente de gradient standard pour tenir compte des contraintes.
  - Au lieu de simplement suivre la direction du gradient négatif, l'algorithme projette cette direction sur l'ensemble des contraintes pour s'assurer que les itérations restent réalisables.
  - La projection de x_{k+1}, assure qu’à chaque itération la solution est admissible
  - La projection peut se faire de plusieurs manières.

Méthodes d'optimisation Sous contraintes

- ## Méthodes du gradient projeté

Algorithme 9 : Méthode du gradient projeté.

1. Initialisation **:** $x_0 \in \mathbb{R}^n$, f une fonction objective de classe 1, $\alpha_0 \mathbb{R}^{+*}$**.**

ε : tolérence, max_iter k=0

2. Répéter
   a) Calculer la direction de la descente d_k= - $\nabla f(x_k)$
   b) Pour le choix de la longueur du pas α_k , on peut utiliser un pas fixe ou utiliser la **Recherche linéaire.**
   c) Projeter le point x_k + α_k d_k sur l'ensemble admissible S, où α_k est la longueur du pas. La projection est notée P(x_k + α_k d_k).
   d) Mettre à jour la solution : x_{k+1}=P(x_k + α_k d_k)
   e) k++
3. Jusqu'à Vérifier la condition d'arrêt : $\|\nabla f(x_k)\|>\varepsilon$ et k<max_iter

Méthodes d'optimisation Sous contraintes

- **Méthodes du gradient projeté**
  - la projection d'un point y$\in \mathbb{R}^n$ sur un ensemble S est définie par :
  - $\text{ProjC}(y) = \underset{x \in S}{\operatorname{argmin}} \| x - y \|^2$, où $\|y\| = \sqrt{{y_1}^2 + {y_2}^2}$
  - La manière dont la projection est effectuée dépend de la nature des contraintes.
    - ✓ **Contraintes linéaires**
    - ✓ **Contraintes simples (boîtes, boules)**
    - ✓ **Contraintes non linéaires**

Méthodes d’optimisation Sous contraintes

- ## Méthodes du gradient projeté

Ensemble S	Projection de y
$\mathbb{R}^{+n}$	max(0,y) (coordonnée par coordonnée)
$[ai, bi]^n$	ai si yi<ai, bi si yi>bi, yi si ai≤yi≤bi pour chaque i
Boule de rayon r et centre c $\|x-c\| \leq r$	y si $\|y-c\| \leq r$, sinon $c+r\frac{y-c}{\|y-c\|}$
Hyperplan affine $\mathbf{a^T x=b,\ a,\ b} \in \mathbb{R}^{\mathbf{n}}$	$y-\frac{a^T y-b}{\|a\|^2}\,a$
Demi-espace $\mathbf{a^T x \leq b}$	$y-\frac{a^T y-b}{\|a\|^2}\,a$
Contrainte linéaire $\mathbf{Ax=b,\ A} \in \mathbb{R}^{\mathrm{mx,n}} b \in \mathbb{R}^m, \boldsymbol{A}$ $\boldsymbol{de\ rang\ m}$	Méthode des moindre carrées : $y-A^T(AA^T)^{-1}(Ay-b)$, sous la condition AA^T est inversible(si A est de rang m).

Méthodes d'optimisation Sous contraintes

Ensemble S	Projection de y
Droite **: ax+by+c=0**	$(y_1 y_2) - \lambda(a\ b)$ avec $\lambda = \frac{ay_1+by_2+c}{a^2+b^2}$
Contrainte quadratique :A Définie Positive	
Si $x^T Ax \leq 1$	y
Si $x^T Ax > 1$ où	$\frac{y}{\|y\|_A}$ où $\| y \|_A = \sqrt{x^T Ax}$
Contrainte quelconque	utiliser les méthodes itératives

Méthodes d'optimisation Sous contraintes

- **Méthodes du gradient projeté**

Exemple. Trouver x minimisant f(x)= $x_1 x_2$ sous h(x)= $x_1 - 2x_2 + 4$

Méthodes d'optimisation Sous contraintes

- **Méthodes du gradient projeté**

Exemple. Trouver x minimisant f(x)= $x_1 x_2$ sous h(x)= $x_1 - 2x_2 + 4$

Calcul de la projection sur une droite: ax+by+c=0

Pour la projection sur une droite ax+by+c=0

d'un point $(y_1 y_2)$:

$$\begin{cases} Py_1 = y_1 - \lambda a \\ Py_2 = y_2 - \lambda b \end{cases}$$

avec $\lambda = \frac{ay_1 + by_2 + c}{a^2 + b^2}$

Méthodes d'optimisation Sous contraintes

- **Méthodes du gradient projeté**

Exemple. Trouver x minimisant f(x)= $x_1 x_2$ sous h(x)= $x_1 - 2x_2 + 4$

Calcul de la projection sur une droite: ax+by+c=0

Pour la projection sur une droite ax+by+c=0

d'un point $(y_1 y_2)$:

$$\begin{cases} Py_1 = y_1 - \lambda a \\ Py_2 = y_2 - \lambda b \end{cases}$$

avec $\lambda = \frac{ay_1 + by_2 + c}{a^2 + b^2}$

Méthodes d'optimisation Sous contraintes

- **Méthodes du gradient projeté**

Exemple. Trouver x minimisant f(x)= $x_1 x_2$ sous h(x)= $x_1 - 2x_2 + 4$

Calcul de la projection sur une droite: ax+by+c=0

```python
#  Projection orthogonale sur la droite x - 2y + 4 = 0
def projection(point, a, ):
    x0, y0 = point
    a, b, c = 1, -2, 4  # x - 2y + 4 = 0
    denom = a**2 + b**2
    lambda_ = (a * x0 + b * y0 + c) / denom
    x_proj = x0 - lambda_ * a
    y_proj = y0 - lambda_ * b
    return np.array([x_proj, y_proj])
```

Méthodes d'optimisation Sous contraintes

- **Méthodes du gradient projeté**

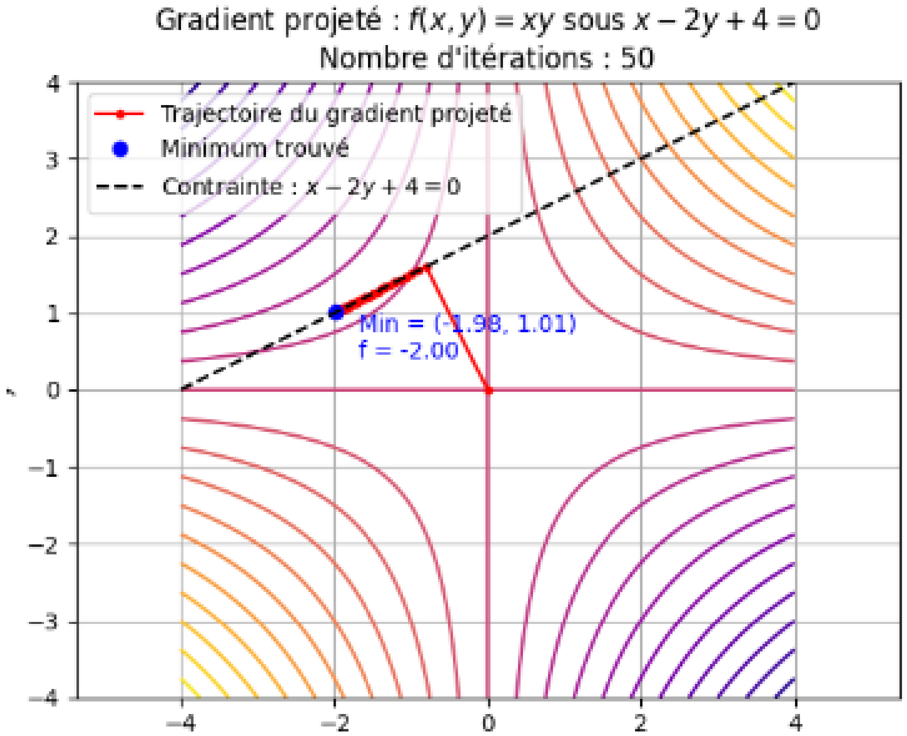
Exemple. Trouver x minimisant f(x)= $x_1 x_2$ sous h(x)= $x_1 - 2x_2 + 4$

```
def gradient_projete_auto(f_func, grad_func, projection, x0, alpha=0.1, max_iter=50, tol=1e-6, verbose=True):
    xk = np.array(x0, dtype=float)
    traj = [xk.copy()]

    for k in range(max_iter):
        grad = np.array(grad_func(*xk))
        yk = xk - alpha * grad
        x_next = projection(yk)
        traj.append(x_next.copy())

        if verbose:
            print(f"Iteration {k+1}: x = {x_next}, f = {f_func(*x_next)}")

        if np.linalg.norm(x_next - xk) < tol:
            break

        xk = x_next

    return xk, np.array(traj)
```

Méthodes d’optimisation Sous contraintes

- **Méthodes du gradient projeté**

Exemple. Trouver x minimisant f(x)= $x_1 x_2$ sous h(x)= $x_1 - 2x_2 + 4$

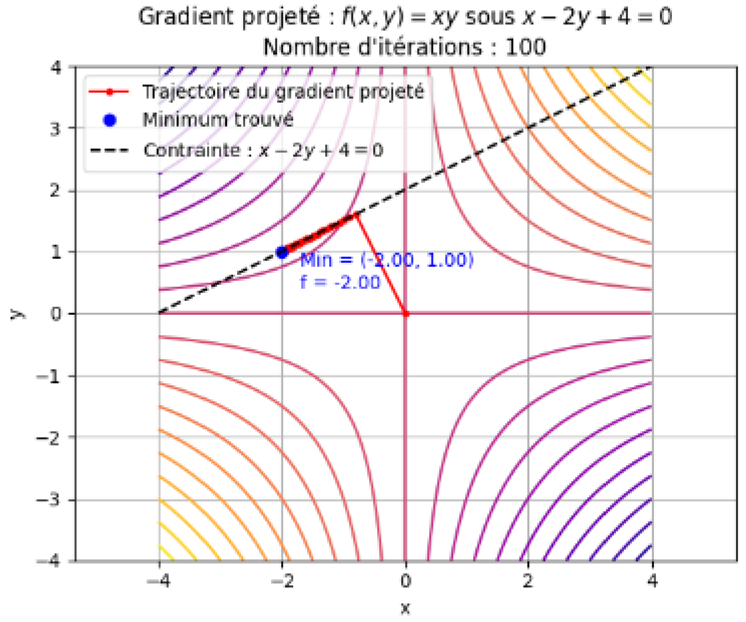


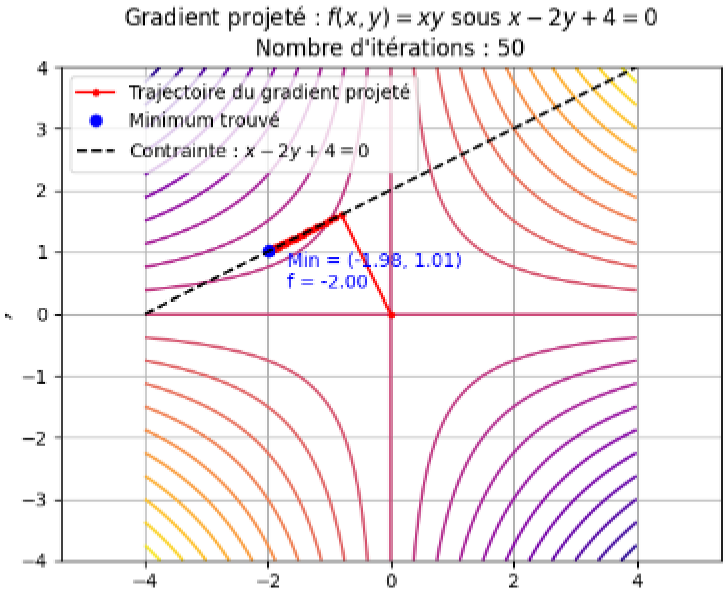

```
Iteration 1: x = [-0.8  1.6], f = -1.2800000000000002
Iteration 2: x = [-0.896  1.552], f = -1.390592
Iteration 3: x = [-0.98432  1.50784], f = -1.4841970688
Iteration 4: x = [-1.0655744  1.4672128], f = -1.5634243990323207
Iteration 5: x = [-1.14032845  1.42983578], f = -1.630482411340956
Iteration 6: x = [-1.20910217  1.39544891], f = -1.6872403129589852
Iteration 7: x = [-1.272374  1.363813], f = -1.735280200884851
Iteration 8: x = [-1.33058408  1.33470796], f = -1.7759411620320136
Iteration 9: x = [-1.38413735  1.30793132], f = -1.8103565995438966
Iteration 10: x = [-1.43340636  1.28329682], f = -1.839485825853954
Iteration 11: x = [-1.47873385  1.26063307], f = -1.8641408030027866
Iteration 12: x = [-1.52043515  1.23978243], f = -1.8850087756615588
Iteration 13: x = [-1.55880033  1.22059983], f = -1.9026714277199435
Iteration 14: x = [-1.59409631  1.20295185], f = -1.9176210964221603
Iteration 15: x = [-1.6265686  1.1867157], f = -1.930274496011715 7
Iteration 16: x = [-1.65644312  1.17177844], f = -1.9409843334243162
Iteration 17: x = [-1.68392767  1.15803617], f = -1.9500491398103421
Iteration 18: x = [-1.70921345  1.14539327], f = -1.957721591935473
Iteration 19: x = [-1.73247638  1.13376181], f = -1.964215555414184
Iteration 20: x = [-1.75387827  1.12306087], f = -1.969712046102566
Iteration 21: x = [-1.77356801  1.113216  ], f = -1.9743642758212117
Iteration 22: x = [-1.79168256  1.10415872], f = -1.9783019230550736
Iteration 23: x = [-1.80834796  1.09582602], f = -1.9816347476738145
Iteration 24: x = [-1.82368012  1.08815994], f = -1.9844556504311164
Iteration 25: x = [-1.83778571  1.08110714], f = -1.9868432625248968
Iteration 26: x = [-1.85076286  1.07461857], f = -1.9888641374010723
Iteration 27: x = [-1.86270183  1.06864909], f = -1.9905746058962677
Iteration 28: x = [-1.87368568  1.06315716], f = -1.9920223464306008
Iteration 29: x = [-1.88379083  1.05810459], f = -1.9932477140188605
Iteration 30: x = [-1.89308756  1.05345622], f = -1.9942848651455642
Iteration 31: x = [-1.90164056  1.04917972], f = -1.995162709859205
Iteration 32: x = [-1.90950931  1.04524534], f = -1.995905717624831
Iteration 33: x = [-1.91674857  1.04162572], f = -1.9965345993976569
Iteration 34: x = [-1.92340868  1.03829566], f = -1.9970668849301774
Iteration 35: x = [-1.92953599  1.03523201], f = -1.9975174114049021
Iteration 36: x = [-1.93517311  1.03241345], f = -1.9978987370131087
Iteration 37: x = [-1.94035926  1.02982037], f = -1.998221491007895
Iteration 38: x = [-1.94513052  1.02743474], f = -1.998494669989082
Iteration 39: x = [-1.94952008  1.02523996], f = -1.9987258886787596
Iteration 40: x = [-1.95355847  1.02322076], f = -1.9989215921777026
Iteration 41: x = [-1.95727379  1.0213631 ], f = -1.9990872356192075
Iteration 42: x = [-1.96069189  1.01965406], f = -1.9992274362280964
Iteration 43: x = [-1.96383654  1.01808173], f = -1.9993461020234609
Iteration 44: x = [-1.96672962  1.01663519], f = -1.999446540752658
Iteration 45: x = [-1.96939125  1.01530438], f = -1.9995315520930497
Iteration 46: x = [-1.97183995  1.01408003], f = -1.9996035056915575
Iteration 47: x = [-1.97409275  1.01295362], f = -1.9996644072173342
Iteration 48: x = [-1.97616533  1.01191733], f = -1.9997159542687515
Iteration 49: x = [-1.9780721  1.01096395], f = -1.9997595836930706
Iteration 50: x = [-1.97982634  1.01008683], f = -1.9997965116378158
```

Méthodes d'optimisation Sous contraintes

- **Méthodes du gradient projeté**

Exemple. Trouver x minimisant f(x)= x_1x_2 sous h(x)= $x_1 - 2x_2 + 4$





Méthodes d’optimisation Sous contraintes

- ## Méthodes du gradient projeté

Exemple. Trouver x minimisant $f(x)= f(x,y)= x^2-10x-y^2$ sur l’ellipse d’équation $h(x,y)=x^2+ 4y^2=16$

```python
def projection_ellipse(point, tol=1e-8, max_iter=100):

    x0, y0 = point
    # Vérification : point déjà est assez proche de la surface de l'ellipse
    # les nombres réels sont représentés approximativement en virgule flottante.
    # Cela signifie que même si mathématiquement deux expressions devraient être égales,
    #leur représentation binaire peut les rendre légèrement différentes, on ne teste pas sur l'égalité
    if abs(x0**2 + 4*y0**2 - 16) < 1e-8:
         return np.array([x0, y0])
    # On doit trouver lambda  tel que f(lambda) = (x0/(1+lambda))^2 + 4*(y0/(1+4*lambda))^2 - 16 = 0.
    lambda_val = 0.0  # initialisation
    for i in range(max_iter):
        # Calcul de f(lambda)
        f_val = (x0/(1+lambda_val))**2 + 4*(y0/(1+4*lambda_val))**2 - 16
        #gradient en lamda calculé analytiquement
        df_dlambda = -2*x0**2/(1+lambda_val)**3 - 32*y0**2/(1+4*lambda_val)**3
        # Mise à jour par Newton
        lambda_new = lambda_val - f_val/df_dlambda
        #test sur la qualité de la solution
        if abs(lambda_new - lambda_val) < tol:
            lambda_val = lambda_new
            break
        lambda_val = lambda_new

    x_proj = x0/(1+lambda_val)
    y_proj = y0/(1+4*lambda_val)
    return np.array([x_proj, y_proj])
```

trouver le point $(x*,y*)$ qui minimise

$$\min_{(x,y)} \|(x,y)-(x_0, y_0)\|^2 \text{ sous contrainte } x^2+ 4y^2=16$$

on utilise la méthode des multiplicateurs de Lagrange, on forme la fonction de Lagrange

$$L(x,y,\lambda)=(x- x_0)^2+(y- y_0)^2 +\lambda(x^2+4 y^2 -16).$$

Calcul du gradient:

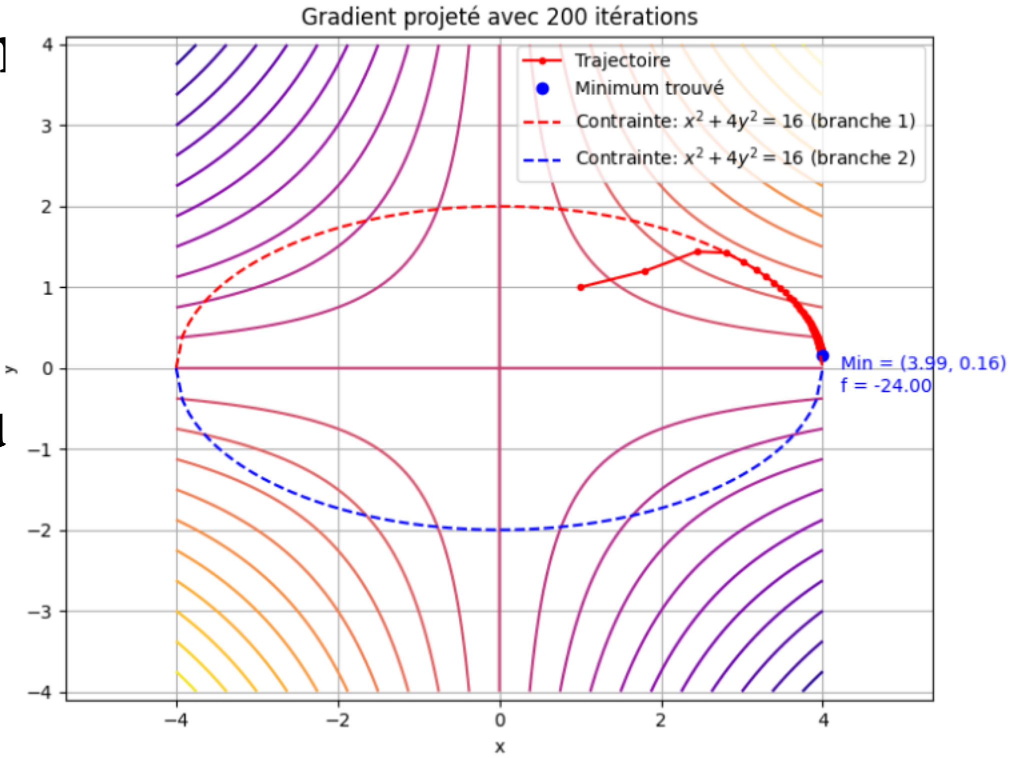
$$\begin{cases} 2(x-x_0)+2\lambda x=0 \\ 2(y-y_0)+8\lambda y=0 \end{cases} \rightarrow \begin{cases} x(1+\lambda)=x_0 \\ y(1+4\lambda)=y_0 \end{cases} \rightarrow \begin{cases} x=x_0/(1+\lambda) \\ y=y_0/(1+4\lambda) \end{cases}$$

on va résoudre l’équation en λ par newton Raphson $:(\frac{x_0}{1+\lambda})^2+ 4(\frac{y_0}{1+4\lambda})^2=16$

Méthodes d'optimisation Sous contraintes

- **Méthodes du gradient projeté**

Exemple. Trouver x minimisant f(x)= f(x , y)= 10x-y^2sur l'ellipse d'équ h(x , y)=x^2+ $4y^2$=16



Méthodes d’optimisation Sous contraintes

- **Méthodes du gradient projeté**
  - Simplicité conceptuelle et d'implémentation
  - Si la projection sur l'ensemble contraint est peu coûteuse (par exemple, pour des contraintes de bornes ou des contraintes linéaires simples), la méthode peut être efficace pour des problèmes de grande dimension.
  - Inefficace pour les contraintes complexes
  - La complexité de la projection peut être élevée.
  - La performance dépend aussi du choix de la longueur du pas