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 d’Optimisation quadratique Séquentielle (OQS)**
  - La méthode **OQS** plus connue sous le nom de SQP(Sequential Quadratic Programming) est une technique pour résoudre les **problèmes d'optimisation non linéaire sous contraintes**.
  - Elle consiste à remplacer le problème initial par une suite de **problèmes quadratiques** sous contraintes linéaires plus faciles à résoudre.
  - C’est une méthode qui combine la méthode de **Newton** et la **programmation quadratique** pour résoudre efficacement des problèmes **non linéaires** avec **contraintes**.

Méthodes d'optimisation Sous contraintes

▪ La méthode Sequential Quadratic Programming

Considérons un problème d'optimisation (différentiable) avec contraintes d'égalité :

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

où f : $\mathbb{R}^n \to \mathbb{R}$ et h: $\mathbb{R}^n \to \mathbb{R}^p, f\; et\; h\; au\; moins\; différentiable.$

On transforme le problème (P) en (P_λ) $avec\; la\; méthode\; de\; Lagrange$

$$P_\lambda\begin{cases} L(\mathrm{x},\lambda) = \mathrm{f(x)} + \lambda^{\mathrm{T}}\mathrm{h(x)} \\ \mathrm{s.c.\,h(x)} = 0 \end{cases}$$

Méthodes d'optimisation Sous contraintes

- **La méthode Sequential Quadratic Programming**

$$soit\ F(x,\lambda) = \begin{cases} \mathrm{F1}(\mathrm{x},\lambda) = \nabla L(\mathrm{x},\lambda) \\ F2(x,\lambda) = \mathrm{h}(\mathrm{x}) \end{cases} (1)$$

- On cherche à résoudre $F(x,\lambda) = 0$

$$(1) \begin{cases} F1(\mathrm{x},\lambda) = 0\,. \\ F2(\mathrm{x},\lambda) = 0. \end{cases} \Leftrightarrow \begin{cases} \nabla \mathrm{f}(\mathrm{x}) + \nabla \mathrm{h}(\mathrm{x})^{\mathrm{T}}\,\lambda = 0. \\ \mathrm{h}(\mathrm{x}) = 0\,. \end{cases}$$

- Pour résoudre (1) on peut utiliser la méthode de Newton-Raphson **où** la formule itérative suivante:

$x_{k+1}=x_k-\frac{\mathbf{f}(x_k)}{\nabla \mathrm{f}(x_k)}$ est utilisée au lieu de $x_{k+1}=x_k-\frac{\mathbf{f}(x_k)}{\nabla^2 f(x_k)}$.

Méthodes d'optimisation Sous contraintes

- **La méthode Sequential Quadratic Programming**

Newton-Raphson appliquée à $F(x, \lambda)$:

$$\begin{pmatrix} x_{k+1} - x_k \\ \lambda_{k+1} - \lambda_k \end{pmatrix} = -\frac{F(x_k,\lambda_k)}{\nabla F(x_k,\lambda_k)}$$

Donc : $\nabla F(x_k, \lambda_k) \begin{pmatrix} x_{k+1} - x_k \\ \lambda_{k+1} - \lambda_k \end{pmatrix} = -F(x_k, \lambda_k) \quad (2)$

Calculons la matrice jacobienne de F.

$$\nabla F(x, \lambda) = \begin{pmatrix} \frac{\nabla F1(x,\lambda)}{\nabla x} & \frac{\nabla F1(x,\lambda)}{\nabla \lambda} \\ \frac{\nabla F2(x,\lambda)}{\nabla x} & \frac{\nabla F1(x,\lambda)}{\nabla \lambda} \end{pmatrix},$$

Méthodes d'optimisation Sous contraintes

- **La méthode Sequential Quadratic Programming**
- Newton-Raphson appliquée à $F(x,\lambda)$:

$$\nabla F(x,\lambda) = \begin{pmatrix} \frac{\nabla F1(x,\lambda)}{\nabla x} & \frac{\nabla F1(x,\lambda)}{\nabla \lambda} \\ \frac{\nabla F2(x,\lambda)}{\nabla x} & \frac{\nabla F1(x,\lambda)}{\nabla \lambda} \end{pmatrix} = \nabla F(x,\lambda) = \begin{pmatrix} \frac{\nabla^2 L(\mathrm{x},\lambda)}{\nabla x^2} & \nabla \mathrm{h}(\mathrm{x})^{\mathrm{T}} \\ \nabla \mathrm{h}(\mathrm{x}) & 0 \end{pmatrix}$$

on pose $H_L(\mathrm{x},\lambda) = \frac{\nabla^2 L(\mathrm{x},\lambda)}{\nabla x^2}$, $\nabla F(x,\lambda) = \begin{pmatrix} H_L(\mathrm{x},\lambda) & \nabla \mathrm{h}(\mathrm{x})^{\mathrm{T}} \\ \nabla \mathrm{h}(\mathrm{x}) & 0 \end{pmatrix}$

(2) $\nabla F(x_k,\lambda_k)\begin{pmatrix} x_{k+1} - x_k \\ \lambda_{k+1} - \lambda_k \end{pmatrix} = -F(x_k,\lambda_k)$

on obtient :

$$\begin{cases} H_L(x_k,\lambda_k)(x_{k+1} - x_k) + \nabla \mathrm{h}(x_k)^{\mathrm{T}}(\lambda_{k+1} - \lambda_k) = -\nabla \mathrm{f}(x_k) - \nabla \mathrm{h}(x_k)^{\mathrm{T}}\lambda_k \\ \nabla \mathrm{h}(x_k)^{\mathrm{T}}(x_{k+1} - x_k) = -\mathrm{h}(x_k) \ . \end{cases}$$

Méthodes d'optimisation Sous contraintes

- **La méthode Sequential Quadratic Programming**

Après simplification on remplace $d_k = x_{k+1} - x_k$, on obtient le système (3) suivant :

$$(3)\begin{cases} H_L(x_k, \lambda_k)\, \boldsymbol{d_k} + \nabla f(x_k) + \nabla h(x_k)^T(\lambda_{k+1}) = 0 \\ \nabla h(x_k)^T\, \boldsymbol{d_k} + h(x_k) = 0\,. \end{cases}$$

Ici on cherche le la variation $\boldsymbol{d_k}$, et le multiplicateur $\boldsymbol{\lambda_{k+1}}$ qui permettent de résoudre le système (2).

On remarque que (3) formule les **conditions d'optimalité** du **problème quadratique** suivant :

$$(4)\begin{cases} \min\limits_{d \in \mathbb{R}^n} \frac{1}{2} d^T H_L(x_k, \lambda_k) d + \nabla f(x_k)^T d\,, f\ \ fonction\ objective \\ \nabla h(x_k)^T d + h(x_k) = 0 \end{cases}$$

Méthodes d'optimisation Sous contraintes

- **La méthode Sequential Quadratic Programming**

$$(3)\begin{cases} H_L(x_k,\lambda_k)\,\boldsymbol{d_k} + \nabla \mathrm{f}(x_k) + \nabla \mathrm{h}(x_k)^{\mathrm{T}}(\lambda_{k+1}) = 0 \\ \nabla \mathrm{h}(x_k)^{\mathrm{T}}\,\boldsymbol{d_k} + h(x_k) = 0\,. \end{cases}$$

$$(\mathrm{QP})\begin{cases} \min_{d\in\mathbb{R}^n} \frac{1}{2} d^T H_L(x_k,\lambda_k)d + \nabla \mathrm{f}(x_k)^T \mathrm{d}\,, f \;\; fonction\ objective \\ \nabla \mathrm{h}(x_k)^{\mathrm{T}}\mathrm{d} + h(x_k) = 0 \end{cases}$$

- **Preuve** : formulons le lagrangien du problème (QP):

$$\mathrm{L}(\mathrm{d},\lambda) = \frac{1}{2} d^T H_L(x_k,\lambda_k)d + \nabla \mathrm{f}(x_k)^T \mathrm{d} + \lambda\left(\nabla \mathrm{h}(x_k)^{\mathrm{T}}\mathrm{d} + h(x_k)\right)$$

la condition d'optimalité du premier ordre est : $\nabla\, \mathrm{L}(\mathrm{d},\lambda) = 0$.

$$\nabla\, \mathrm{L}(\mathrm{d},\lambda) = \nabla_d\left(\frac{1}{2} d^T H_L(x_k,\lambda_k)d + \nabla \mathrm{f}(x_k)^T \mathrm{d} + \lambda\left(\nabla \mathrm{h}(x_k)^{\mathrm{T}}\mathrm{d} + h(x_k)\right)\right).$$

$$= H_L(x_k,\lambda_k)d + \nabla \mathrm{f}(x_k) + \nabla \mathrm{h}(x_k)^{\mathrm{T}}\boldsymbol{\lambda}.$$

$\boldsymbol{\lambda}$ est exactement le λ_{k+1} qu'on cherche dans (3) et d c'est la direction qu'on cherche $\boldsymbol{d_k}$.

Donc pour trouver (d, λ) il suffit de résoudre le système linéaire (2) et trouver $(d_k,\ \lambda_{k+1})$.

Méthodes d’optimisation Sous contraintes

- **La méthode Sequential Quadratic Programming**

Algorithme 10. Méthode SQP sous contrainte d’égalité

1. **Initialisation** : $x_0, \varepsilon, \mathrm{k} = 0, \mathrm{max\_iter}, f\colon \mathbb{R}^n \to \mathbb{R}, \mathrm{h}\colon \mathbb{R}^n \to \mathbb{R}^p, \lambda_k$
2. **Répéter**
   a) Résoudre le problème quadratique pour obtenir la solution primale d_k et le multiplicateur λ_{k+1}:

$$QP \begin{cases} \min\limits_{d \in \mathbb{R}^n} \frac{1}{2} d^T H_L(x_k, \lambda_k) d + \nabla \mathrm{f}(x_k)^T \mathrm{d} \ . \\ \quad s.c.\ \nabla \mathrm{h}(x)^{\mathrm{T}} \mathrm{d} + h(\mathrm{x}) = 0\ . \end{cases}$$

   a) Mettre à jour la solution : $x_{k+1} = x_k + \mathrm{d}_k$
   b) Mettre à jour λ_{k+1} avec la nouvelle valeur trouvée
   c) k++

3. jusqu’à ce que $\|\nabla \mathrm{L}(x_n, \lambda_k)\| < \varepsilon$ ou k>max_iter

4. retourner $\boldsymbol{x_k}$

Méthodes d'optimisation Sous contraintes

- **La méthode Sequential Quadratic Programming**

Exemples. Trouver le minimum de la fonction f(x,y)= $x_1^2 + 2(x_2 - 2)^2$ sous contrainte h(x,y)= $x_1^2 + x_2^2 - 1$

a) Résoudre le problème quadratique QP qui revient à résoudre le problème linéaire :

$$\nabla F(x_k, \lambda_k) \begin{pmatrix} x_{k+1} - x_k \\ \lambda_{k+1} - \lambda_k \end{pmatrix} = -F(x_k, \lambda_k) \quad (2)$$

Lagrangien : $L(x_1, x_2, \lambda) = x_1^2 + 2(x_2 - 2)^2 + \lambda(x_1^2 + x_2^2 - 1)$

Le gradient de L : $\nabla L(x_1, x_2, \lambda) = \begin{pmatrix} 2x_1 + 2\lambda x_1 \\ 4(x_2 - 2) + 2\lambda x_2 \end{pmatrix}$, $\nabla h(x_1, x_2) = \begin{pmatrix} 2x_1 \\ 2x_2 \end{pmatrix}$

La hessienne de L : $H_L(x_1, x_2, \lambda) = \begin{pmatrix} 2 + 2\lambda & 0 \\ 0 & 4 + 2\lambda \end{pmatrix}$,

$\nabla f(x_1, x_2) = (2x_1 \quad 4x_2 - 8)$

Méthodes d'optimisation Sous contraintes

- **La méthode Sequential Quadratic Programming**

Exemples. Trouver le minimum de la fonction f(x,y)= $x_1^2 + 2(x_2 - 2)^2$ sous contrainte h(x,y)= $x^2 + y^2 - 1$

Itération 0 :

k=0 ; $x_0=\left(-\frac{1}{2},\frac{1}{2}\right)$, $\lambda_0=1$.

$H_L(x_0, 1)=\begin{pmatrix} 4 & 0 \\ 0 & 6 \end{pmatrix}$, $\nabla f(x_0) = (-1, -6)^t$, $\nabla h(x_0)^T = (-1, \quad 1)^t$

$h(x_0)= -\frac{1}{2}$

Méthodes d'optimisation Sous contraintes

- **La méthode Sequential Quadratic Programming**

Exemples. Trouver le minimum de la fonction f(x,y)= $x_1^2 \ + \ 2(x_2 \ - \ 2)^2$ sous contrainte h(x,y)= $x^2 + y^2 - 1$

Itération 0 :

la résolution du sous-problème quadratique est équivalente à la résolution du système linéaire (2) pour k0:

on a $\nabla F(x_0, \lambda_0) = \begin{pmatrix} \frac{\nabla^2 L(x_0,\lambda_0)}{\nabla x^2} & \nabla \mathrm{h}(x_0)^{\mathrm{T}} \\ \nabla \mathrm{h}(x_0) & 0 \end{pmatrix} = \begin{pmatrix} 4 & 0 & 1 \\ 0 & 6 & -1 \\ -1 & 1 & 0 \end{pmatrix}$

$$\begin{pmatrix} 4 & 0 & 1 \\ 0 & 6 & -1 \\ -1 & 1 & 0 \end{pmatrix} \begin{pmatrix} d_1 \\ d_2 \\ \lambda_1 \end{pmatrix} = -\begin{pmatrix} \nabla \mathrm{f}(x_0) \\ \mathrm{h}(x_0) \end{pmatrix} = -\begin{pmatrix} -1 \\ -6 \\ -\frac{1}{2} \end{pmatrix}$$

$$on\ obtient\ le\ système\ suivant \begin{cases} 4d_1 + \lambda_1 = 1 \\ 6d_2 - \lambda_1 = 6 \\ -d_1 + d_2 = \frac{1}{2} \end{cases},$$

Méthodes d’optimisation Sous contraintes

- **La méthode Sequential Quadratic Programming**

Exemples. Trouver le minimum de la fonction f(x,y)= $x_1^2 + 2(x_2 - 2)^2$ sous contrainte h(x,y)= $x^2 + y^2 - 1$

Itération 0 : (3) $\begin{cases} 4d_1 - 1 + \lambda_1 = 0 \\ 6d_2 - 6 - \lambda_1 = 0 \\ -d_1 + d_2 = \frac{1}{2} \end{cases}$, $\begin{cases} d_1 = 2/5 \\ d_2 = 9/10 \\ \lambda_1 = -\frac{3}{5} \end{cases}$,

Mise à jour : $x_1 = x_0 + d = \begin{pmatrix} -\frac{1}{10} \\ \frac{14}{10} \end{pmatrix}, \lambda_1 = -\frac{3}{5}$

…

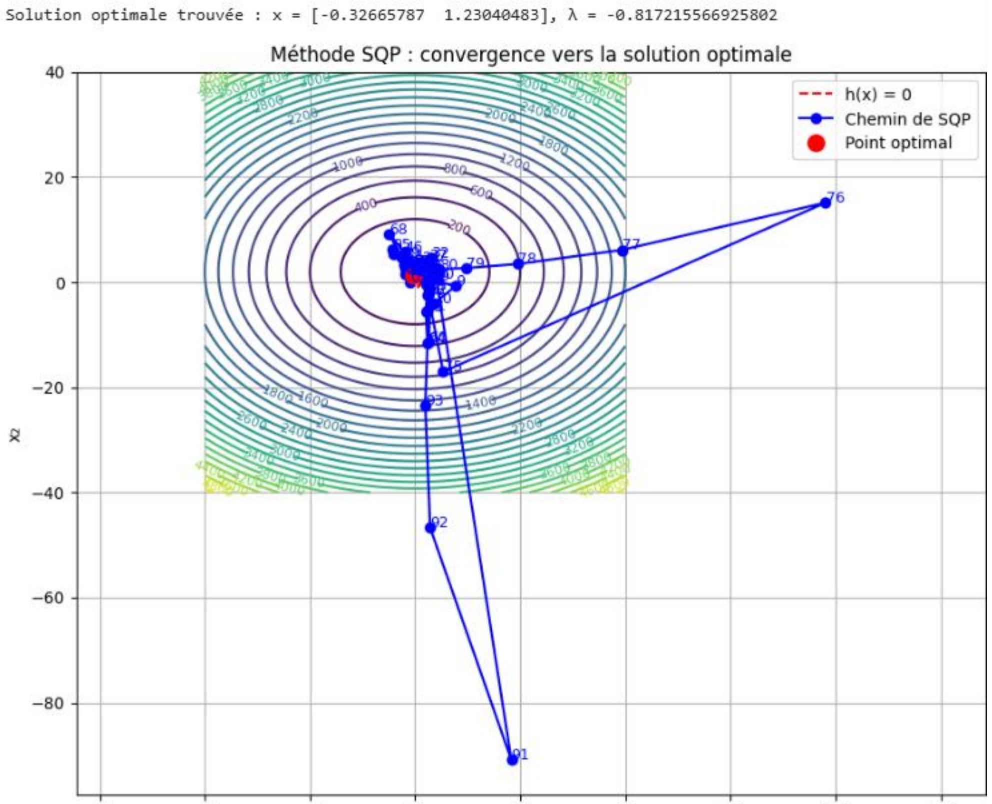
Méthodes d'optimisation Sous contraintes

- **La méthode Sequential Quadratic Programming**

Exemples. Trouver le minimum de la fonction f(x,y)= $x_1^2 + 2(x_2 - 2)^2$ sous contrainte h(x,y)= x^2+ y^2 – 1. tol=1e-6. max_

```
=== ITERATION 99 ===
x_k = [0.25917234 1.75709675]
λ_k = -0.19606056685356382
∇f(x_k) = [ 0.51834468 -0.97161299]
∇h(x_k) = [0.51834468 3.51419351]
H_f(x_k) =
[[2. 0.]
 [0. 4.]]
H_h(x_k) =
[[2. 0.]
 [0. 2.]]
H (modifiée) H_f + λ * H_h =
[[1.60787887 0.        ]
 [0.         3.60787887]]
Système résolu linéaire : A =
[[ 1.60787887  0.         -0.51834468]
 [ 0.          3.60787887 -3.51419351]
 [ 0.51834468  3.51419351  0.        ]]
Système résolu linéaire : b = [-0.51834468  0.97161299 -2.1545593 ]
Solution: d = [-0.5858302132819211, -0.5266919204845274], λ = -0.817215566925802
Norme de d = 0.7877825955792759, tol = 1e-6

Solution optimale trouvée : x = [-0.32665787  1.23040483], λ = -0.817215566925802
```

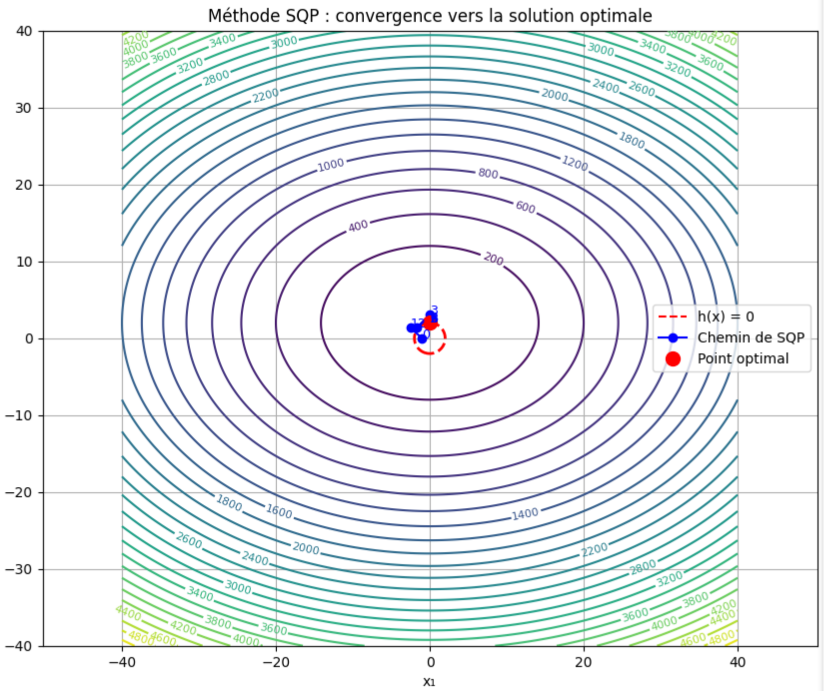


Méthodes d'optimisation Sous contraintes

- **La méthode Sequential Quadratic Programming**

Exemples. Trouver le minimum de la fonction f(x,y)= $x_1^2 + 2(x_2 - 2)^2$ sous contrainte h(x,y)= $x^2 + y^2 - 4$

```
Solution optimale trouvée : x = [-2.04819846e-19  2.00000000e+00], λ = 1.3305515355242805e-19
```




```
=== ITERATION 7 ===
x_k = [-5.43958144e-10  2.00000000e+00]
λ_k = 3.7653604352534426e-10
∇f(x_k) = [-1.08791629e-09  4.52015314e-10]
∇h(x_k) = [-1.08791629e-09  4.00000000e+00]
H_f(x_k) =
[[2. 0.]
 [0. 4.]]
H_h(x_k) =
[[2. 0.]
 [0. 2.]]
H (modifiée) H_f + λ * H_h =
[[2. 0.]
 [0. 4.]]
Système résolu linéaire : A =
[[ 2.00000000e+00  0.00000000e+00  1.08791629e-09]
 [ 0.00000000e+00  4.00000000e+00 -4.00000000e+00]
 [-1.08791629e-09  4.00000000e+00  0.00000000e+00]]
Système résolu linéaire : b = [ 1.08791629e-09 -4.52015314e-10 -4.52015314e-10]
Solution: d = [5.439581435449709e-10, -1.1300382836253647e-10], λ = 1.3305515355242805e-19
Norme de d = 5.555720719703977e-10, tol = 1e-6

 Convergence atteinte !
```

Méthodes d’optimisation Sous contraintes

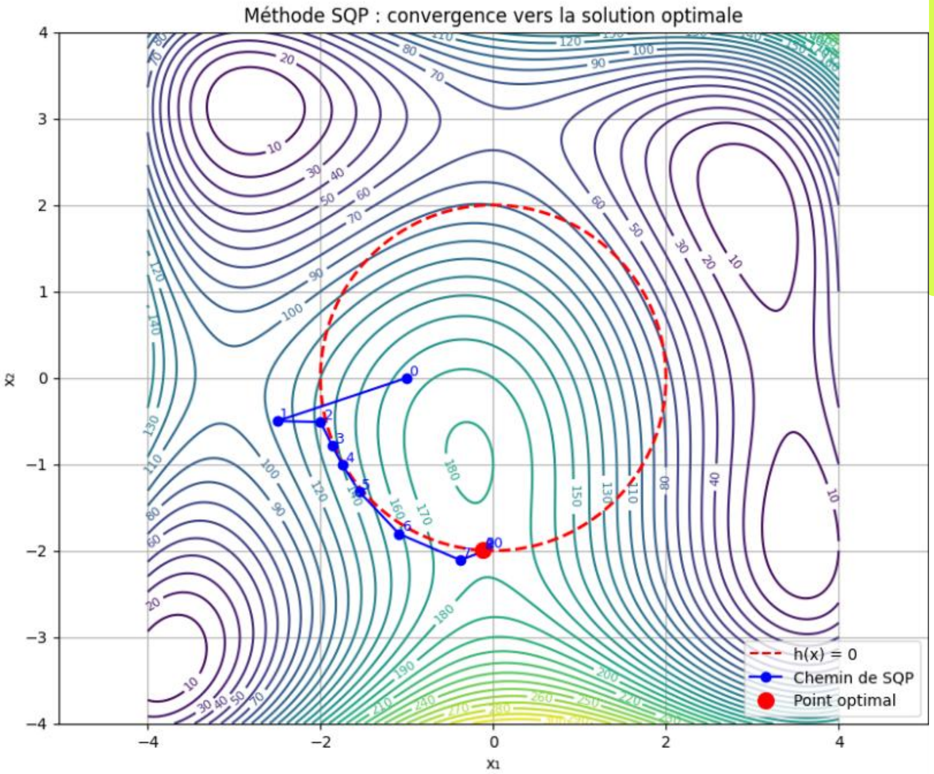
- **La méthode Sequential Quadratic Programming**

Exemples. Trouver le minimum de la fonction f(x,y)= $(x^2 + y - 11)^2 + (x + y^2 - 7)^2$ sous contrainte h(x,y)= $x^2 + y^2 - 4$

x0 = [-1, 0], λ0 = 1, tol=1e-6, max_iter=10

```
=== ITERATION 9 ===
x_k = [-0.11938612 -1.99653736]
λ_k = 0.2291509558149751
∇f(x_k) = [-0.06683101 -0.94216807]
∇h(x_k) = [-0.23877225 -3.99307473]
H_f(x_k) =
[[-49.8151129   -8.46369394]
 [ -8.46369394  21.35639282]]
H_h(x_k) =
[[2. 0.]
 [0. 2.]]
H (modifiée) H_f + λ * H_h =
[[-49.35681099  -8.46369394]
 [ -8.46369394  21.81469473]]
Système résolu linéaire : A =
[[-49.35681099  -8.46369394   0.23877225]
 [ -8.46369394  21.81469473   3.99307473]
 [ -0.23877225  -3.99307473   0.        ]]
Système résolu linéaire : b = [ 6.68310068e-02  9.42168067e-01 -4.14488520e-04]
Solution: d = [-0.00023839506442932662, 0.00011805705564O5311], λ = 0.234800260486828
Norme de d = 0.00026602570389113573, tol = 1e-6

Solution optimale trouvée : x = [-0.11962452 -1.99641931], λ = 0.234800260486828
```



Méthodes d’optimisation Sous contraintes

- **La méthode Sequential Quadratic Programming**
  - **Avantages**
    - ✓ La méthode SQP est très efficace pour traiter les problèmes d'optimisation avec des contraintes non linéaires.
    - ✓ Elle est adaptée pour résoudre des problèmes d'optimisation à grande échelle avec de nombreuses variables et contraintes.
    - ✓ la méthode SQP peut approcher la solution par des points réalisables et non réalisables, ce qui est avantageux lorsqu'il est difficile de trouver un point de départ réalisable.
    - ✓ Convergence rapide (quadratique) proche de l’optimum
  - **Limites**
    - ✓ Chaque itération de la méthode SQP implique la résolution d'un sous-problème de programmation quadratique (PQ), ce qui peut être coûteux en termes de calcul, surtout pour les problèmes à grande échelle.
    - ✓ Les sous-problèmes de programmation quadratique générés au sein de l'algorithme SQP peuvent devenir non réalisables ce qui aboutit à l’échec de la méthode.