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

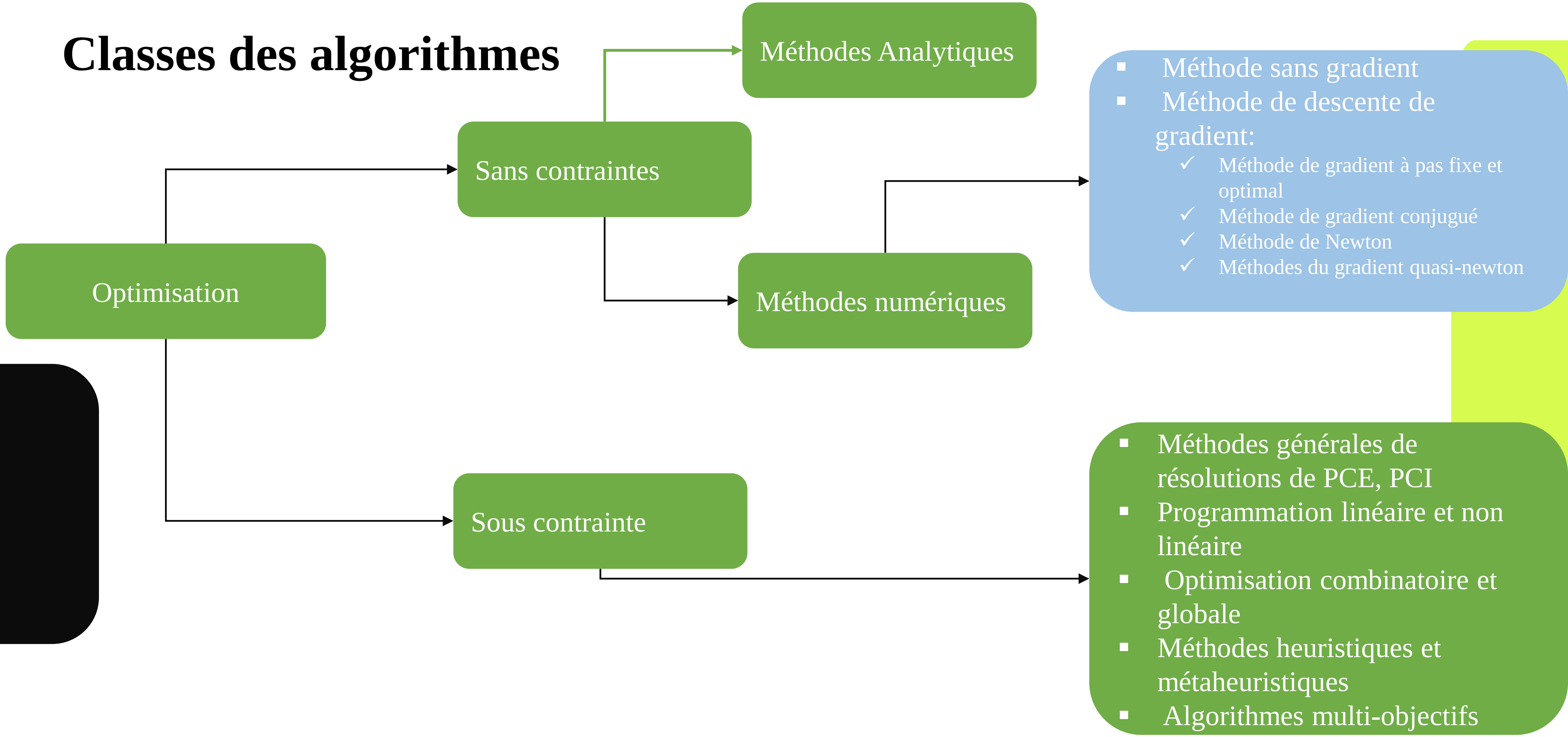
05 Programmation Linéaire et non Linéaire

06 Optimisation combinatoire et globale

07 Méthodes heuristiques et métaheuristiques

Les algorithmes d’optimisation

Classes des algorithmes



Méthodes d'optimisation de descente gradient

- **Rappel : Méthode de Newton**

Algorithme 6 : Méthode de Newton

1. **Initialisation** $x_0, \varepsilon, \mathrm{n} = 0, \max\_\mathrm{iter}$
2. **Répéter jusqu'à convergence** :
   a) Calculer le **gradient** $\nabla \mathrm{f}(x_n)$.
   b) Calculer la **matrice Hessienne** $\nabla^2 f(x_n)$
   c) Résoudre le système linéaire :$p_n = -\frac{\nabla \mathrm{f}(x_n).}{\nabla^2 f(x_n)}$
   d) Mettre à jour la solution : $x_{n+1} = x_n + p_n$
   e) n++

3. Vérifier la condition d'arrêt : $\|\nabla \mathrm{f}(x_n)\| < \varepsilon$ ou n>max_iter

4. Retourner la solution x_n

Méthodes d'optimisation de descente gradient

- **Méthode de Newton**
  - Il existe d'autres versions de l'algorithme de Newton : Newton-Raphson, Newton tronquée, Gauss-Newton, .., et quasi-Newton
- **Avantages :**
  - Très performante lorsque la fonction est deux fois différentiable et que la hessienne est définie positive.
  - Atteint des solutions précises en peu d'itérations.
  - **Convergence Super-linéaire voire quadratique** : Si le point de départ est proche de la solution, la convergence est très rapide.
- **Inconvénients :**
  - Nécessite le calcul du gradient et de la hessienne, ce qui est coûteux pour des fonctions complexes ou de grande dimension.
  - Un mauvais choix peut entraîner divergence ou convergence vers un minimum local.
  - L'algorithme échoue si la matrice hessienne est singulière.
  - L'inversion de la hessienne devient impraticable pour de très grandes dimensions.

Méthodes d’optimisation de descente gradient

Méthode Quasi-newton

- La méthode Quasi-Newton permet de pallier aux inconvénients de la méthode de Newton, en particulier le calcul (coûteux) de l’inverse de la matrice hessienne $\nabla^2 f(x_k)$ à chaque itération.

- Dans les méthodes quasi-Newton, on cherche à construire une **approximation de l’inverse** de la matrice hessienne **, c’est-à-dire:** $(\nabla^2 f(x_k))^{-1}$.

- On remplace alors et $(\nabla^2 f(x_k))^{-1}$ par une matrice B_k, éventuellement constante, qui est censée l’approcher.

Méthodes d'optimisation de descente gradient

Méthode Quasi-newton

Algorithme 7: Algorithme Quasi- Newton

Initialisation $x_0, \varepsilon, \mathrm{n} = 0, \max\_\mathrm{iter}, \alpha_{0,}\ \mathrm{B}_0$

Initialiser la matrice B_0 (approximation de l'inverse de la Hessienne).

En général, on prend $\mathrm{B}_0 = I$ (identité).

Répéter

1. Calculer le **gradient** $\nabla \mathrm{f}(x_n)$.
2. Calculer la direction de recherche $\mathrm{d}_\mathrm{n} = - \mathrm{B}_\mathrm{n} \nabla \mathrm{f}(x_n)$.
3. Trouver un pas tel que $\alpha_n = \min_{\alpha > 0} \mathrm{f}(x_n + \alpha d_n)$ // calcul optimal du pas
4. Mettre à jour la solution : $x_{n+1} = x_n + \alpha_n \mathrm{d}_n$

Méthodes d'optimisation de descente gradient

Méthode Quasi-newton

Algorithme quasi- Newton

5. Calculer la différence entre les nouveaux et anciens gradients :

   Sn est le déplacement., y_n est la variation du gradient

   Sn =$x_{n+1} - x_n$, y_n =∇f(x_{n+1})- ∇f(x_n)

6. Mettre à jour la matrice B en utilisant la formule de mise à jour de BFGS (Broyden-Fletcher-Goldfarb-Shanno) ou DFP (Davidon-Fletcher-Powell).

Mise à jour avec BFGS : $B_{n+1} = B_n - \frac{B_n S_n s_n^T B_n}{s_n^T B_n s_n} + \frac{y_n\, y_n^T}{y_n^T\, s_n}$

Mise à jour avec DFP : $B_{n+1} = B_n - \frac{B_n y_n y_n^T B_n}{y_n^T B_n y_n} + \frac{s_n\, s_n^T}{s_n^T\, y_n}$

7. n++
8. jusqu'à ce que ‖∇f(x_n)‖<ε ou n>max_iter
9. **retourner $\boldsymbol{x_n}$**

Méthodes d'optimisation de descente gradient

Méthode Quasi-newton

Exemple. f(x,y)= x^2 - 5xy + y^4 - 25x - 8y, x0 = [0.0, 0.0], epsilon=1e-6, max_iter=100

```python
##methode quasi-newton ex1 avec rotation
import numpy as np
import sympy as sp
import matplotlib.pyplot as plt

# Définition des variables symboliques
x, y = sp.symbols('x y')

# Définition de la fonction objectif
f_symbolic = x**2 - 5*x*y + y**4 - 25*x - 8*y

# Calcul des dérivées partielles (gradient) symboliques
grad_f_symbolic = [sp.diff(f_symbolic, var) for var in (x, y)]

# Conversion en fonctions numériques
grad_f_lambdified = [sp.lambdify((x, y), grad, 'numpy') for grad in grad_f_symbolic]

# Fonction pour calculer le gradient numériquement
def grad_f(x_val, y_val):
    return np.array([grad(x_val, y_val) for grad in grad_f_lambdified])

# Recherche linéaire pour trouver le pas optimal
def line_search(f, grad_f, x, p, alpha=1e-4, beta=0.7, t=1.0, max_iter=100):
    for _ in range(max_iter):
        if f(x[0] + t * p[0], x[1] + t * p[1]) <= f(x[0], x[1]) + alpha * t * np.dot(grad_f(x[0], x[1]), p):
            return t
        t *= beta
    return t
```

Méthodes d’optimisation de descente gradient

Méthode Quasi-newton

Exemple. $f(x,y)= x^2 - 5xy + y^4 - 25x - 8y$, $x0 = [0.0, 0.0]$, epsilon=1e-6, max_iter=100

```python
# Algorithme quasi-Newton DFP
def quasi_newton_dfp(f, grad_f, x0,  epsilon=1e-6, max_iter=100):
    x = np.array(x0)
    grad = grad_f(x[0], x[1])
    n = len(x)
    B = np.eye(n)  # Initialisation de B0 comme matrice identité
    trajectory = [x]  # Trajectoire des points
    # Itérations
    num_iterations = 0
    for k in range(max_iter):
        p = -np.dot(B, grad)  # Direction de recherche
        t = line_search(f, grad_f, x, p)  # Recherche du pas optimal
        x_new = x + t * p  # Mise à jour du point
        grad_new = grad_f(x_new[0], x_new[1])

        # Critère d'arrêt
        if np.linalg.norm(grad_new) < epsilon:
            print("risque Hessienne singuulière")
            break
        s = x_new - x
        y = grad_new - grad
        # mise a jour de B par la méthode DFP
        Bs = np.dot(B, y)
        B = B - np.dot(Bs, Bs.T) / np.dot(y, Bs) + np.outer(s, s) / np.dot(y, s)
        x = x_new
        grad = grad_new
        trajectory.append(x)
        # Compter les itérations
        num_iterations += 1
    return x, f(x[0], x[1]), trajectory, num_iterations
```

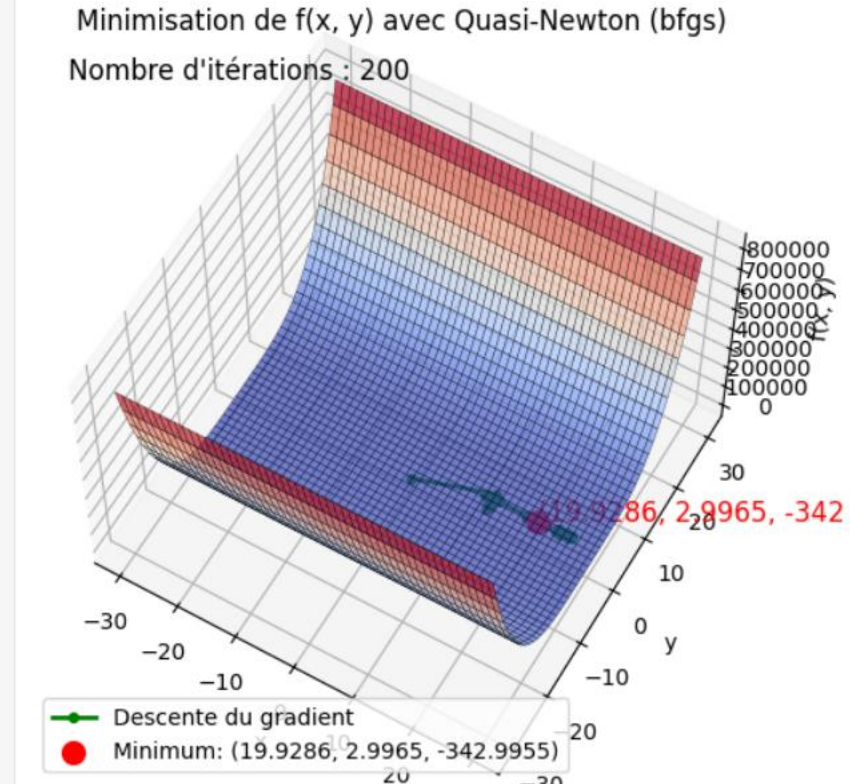
Méthodes d’optimisation de descente gradient

Méthode Quasi-newton

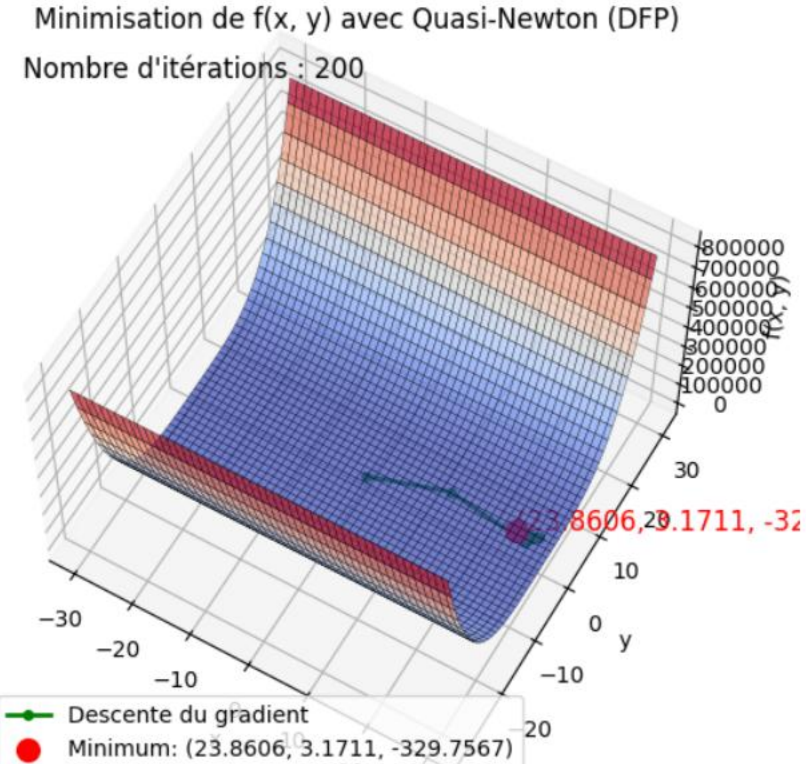
Exemple. $f(x,y)= x^2 - 5xy + y^4 - 25x - 8y$, x0 = [0.0, 0.0], epsilon=1e-6 max_iter=100

```
# Rotation de la vue à 30 degrés verticalement
ax.view_init(elev=60, azim=-60)
```

```
Le minimum trouvé est à x = 19.9286, y = 2.9965, f(x, y) = -342.9955
Nombre d'itérations : 200
```



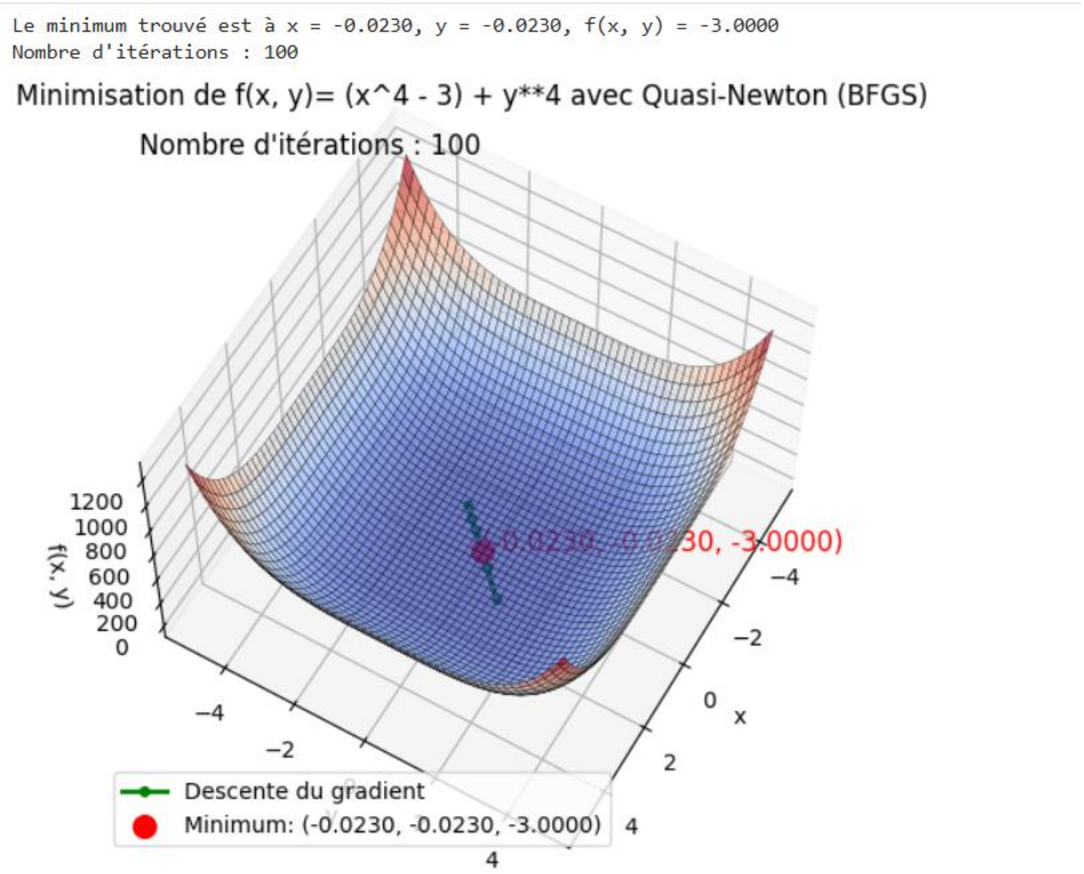

```
Le minimum trouvé est à x = 23.8606, y = 3.1711, f(x, y) = -329.7567
Nombre d'itérations : 200
```



Méthodes d’optimisation de descente gradient

Méthode Quasi-newton

Exemple. f(x,y)= $(x^4 - 3) + y^4$, x0 =x0 = [-1.0, -1.0], epsilon=1e-6, max_iter=100



Méthodes d’optimisation de descente gradient

Méthode Quasi-newton

- La méthode Quasi-Newton permet de pallier aux inconvénients de la méthode de Newton, elle:
  - est efficace pour les problèmes de grande dimension, car il ne nécessite pas le calcul explicite de la matrice hessienne.
  - est connu pour sa convergence **superlinéaire**, ce qui signifie qu'il converge rapidement vers le minimum local.
  - fonctionne bien pour une large gamme de fonctions objectifs comme les fonctions non linéaires et non quadratiques
  - nécessitent généralement une recherche linéaire à chaque itération, ce qui peut être coûteux en termes de temps de calcul,
  - Dans de rare cas, la matrice hessienne peut perdre sa qualité de définie positivité, ce qui peut affecter la stabilité de la convergence.

Comparaison de la convergence des méthodes sans contraintes

Algorithme	Type de convergence	Desciprion
Dichotomie	Convergence linéaire	La longueur de l'intervalle de recherche est divisée par deux à chaque itération. Le nombre d'itérations nécessaires pour atteindre une certaine précision est proportionnel au logarithme de la précision souhaitée.
Section dorée	Convergence linéaire	La longueur de l'intervalle de recherche est réduite d'un facteur constant (le nombre d'or) à chaque itération
Gradient à pas fixe	Convergence linéaire	La méthode de descente de gradient à pas fixe a généralement une convergence linéaire La vitesse de convergence dépend fortement du choix de la longueur de pas
Gradient à pas optimal	Convergence linéaire (dans le pire des cas)	Ajustement du pas à chaque itération, elle peut souvent converger plus rapidement que la méthode à pas fixe. La convergence dépend de la fonction objectif et de la précision de la recherche linéaire.
Méthode de Newton	Convergence quadratique	l'erreur diminue au carré à chaque itération, ce qui conduit à une convergence très rapide près du minimum. Elle nécessite le calcul de la matrice hessienne, ce qui peut être coûteux en termes de temps de calcul et de mémoire. De plus la méthode de newton demande une initialisation proche de l'optimum locale, et peut diverger si cette initialisation est mal choisi.
Méthodes quasi-Newton (BFGS, DFP)	Convergence superlinéaire	Elles offrent un bon compromis entre vitesse de convergence et coût de calcul, car elles n'ont pas besoin de calculer la matrice hessienne complète. La méthode BFGS est considérée comme plus stable et plus efficace que la méthode DFP.

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

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

- Dans cette partie, on s’intéresse à des problèmes d’optimisation sous contraintes.
- Les problèmes d'optimisation sous contraintes d'égalité sont des problèmes mathématiques où l'on cherche à minimiser ou maximiser une fonction objective tout en respectant des contraintes qui doivent être satisfaites.
- Soit f une fonction objective, qu’on cherche à minimiser, typiquement non linéaire, définie par :

  $\mathrm{K} \quad \mathbb{R}^n \rightarrow \mathbb{R} \quad n \geq 1$

  $$x = (x1, \ldots, xn) \rightarrow f(x)$$

  On cherche un point tel que : $f(a^*) = \min_{x \in \mathrm{K}} f(x)$.
- Les problèmes sous contrainte sont définis sous forme de :
  - Problème avec contraintes d’égalité, (PCE) : $h(x)=0$
  - Problème avec contraintes d’inégalité (PCI): $g(x) \leq 0$
  - où les fonctions f, g et h sont différentiables au moins une fois.

Méthodes d'optimisation Sous contraintes

- Un problème est dit problème d'optimisation avec contraintes d'égalité s'il est défini de la forme (PCE):

$$\begin{cases} \min\limits_{x \in \mathbb{R}^n} f(\mathrm{x}), f \text{ une fonction objective}. \\ \mathrm{h(x)} = 0, \mathrm{h} \text{ fonction de } \mathbb{R}^n \to \mathbb{R}^p \text{ est une fonction } \textbf{contrainte d'égalité} . \end{cases}$$

p est un entier ≥ 1 qui représente le nombre de contraintes $h_1 h_2, .., h_p$.

- Un problème est dit problème d'optimisation avec contraintes d'inégalité s'il est défini de la forme (PCI):

$$\begin{cases} \min\limits_{x \in \mathbb{R}^n} f(\mathrm{x}), f \text{ une fonction objective}. \\ \mathrm{g(x)} \leq 0, \mathrm{g} \text{ fonction de } \mathbb{R}^n \to \mathbb{R}^m \text{ est une fonction } \textbf{contrainte d'inégalité} . \end{cases}$$

m est un entier ≥ 1 qui représente le nombre de contraintes $g_1 g_2, .., g_m$

- Le cas général peut englober un problème avec des contraintes d'égalités et d'inégalités.

Méthodes d'optimisation Sous contraintes

- **Définition.** La **matrice jacobienne** de la fonction de contraintes h $=\begin{pmatrix} h_1 \\ h_2 \\ \cdot \\ h_p \end{pmatrix}$ définie de $\mathbb{R}^n \rightarrow \mathbb{R}^p$ est donnée par :

$$J_h(x)=\nabla h(x) = \begin{bmatrix} \frac{\partial h_1}{\partial x_1} & \frac{\partial h_1}{\partial x_2} & \cdots & \frac{\partial h_1}{\partial x_n} \\ \frac{\partial h_2}{\partial x_1} & \frac{\partial h_2}{\partial x_2} & \cdots & \frac{\partial h_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial h_p}{\partial x_1} & \frac{\partial h_p}{\partial x_2} & \cdots & \frac{\partial h_p}{\partial x_n} \end{bmatrix}$$

Méthodes d'optimisation Sous contraintes

- **Définition.** Tout vecteur $x \in \mathbb{R}^n$ vérifiant les contraintes est appelée **solution admissible**.

- On définit l'ensemble S={$x \in \mathbb{R}^n$ / x est admissible}

- **Définition. (Direction admissible)** . On dit que d $\in \mathbb{R}^n$ est une **direction admissible** en $x \in S$ s'il existe $\alpha > 0$ tel que $x + td \in S$, $\forall t \in [-\alpha, \alpha]$

Méthodes d’optimisation Sous contraintes

- **Théorème. condition nécessaire d’amissibilité**
  - une **condition nécessaire** pour qu'une direction « d » soit considérée comme admissible dans le contexte de l'optimisation sous contraintes d'égalité est $\nabla h(x)^T d = 0$; où $\nabla h(x)$ est la matrice jacobienne des contraintes évaluées au point x.
  - Cette condition garantit que le mouvement dans la direction d reste tangent à l'espace des contraintes.
  - Elle est basée sur l'idée que pour rester sur la surface de contrainte, tout mouvement doit être tangent à cette surface

Problèmes d’optimisation sous contrainte d’égalité

- Dans les problèmes d’optimisation sous contrainte d’égalité, les contraintes peuvent être de différents types, amenant à des sous types de problèmes d’optimisation différents.
- On distingue plusieurs sous-types de problèmes d’optimisation sous contrainte, notamment :
  - Problèmes linéaires avec contraintes d'égalité
  - Problèmes non linéaires avec contraintes d'égalité
  - Problèmes quadratiques avec contraintes d'égalité
  - Problèmes dynamiques avec contraintes d'égalité

Problèmes d’optimisation sous contrainte d’égalité

- **Optimisation linéaire sous contrainte d'égalité**

Un problème d’optimisation avec contraintes d’égalité linéaires prend la forme suivante :

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

où A est une matrice p × n avec p < n et b $\in \mathbb{R}^p$.

On notera S = $\{\mathrm{x} \in \mathbb{R}^n \,, \mathrm{Ax} - \mathrm{b} = 0\}$.

A est une matrice de contraintes, et b un vecteur constant.

Problèmes d’optimisation sous contrainte d’égalité

- **Optimisation linéaire sous contrainte d'égalité**
- **Proposition.** Soit x une solution réalisable du problème d’optimisation avec contraintes d’égalité linéaires. Un vecteur d $\in \mathbb{R}^n$ est alors une direction admissible en x si et seulement si

  Ad = 0.

  En effet, on a :

  Soit x $\in$ S, A(x + td) − b = Ax-b+tAd = tAd

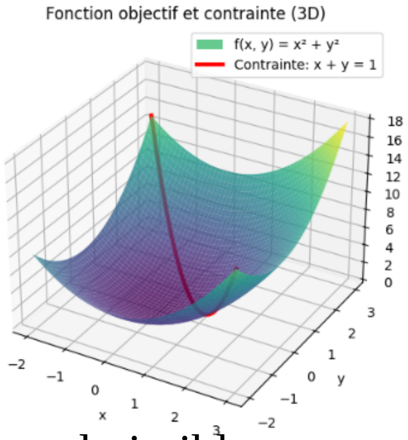
  Si si Ad=0, alors tAd =0, 0 ∀t, et donc les directions admissibles d sont caractérisées par **Ad = 0**.

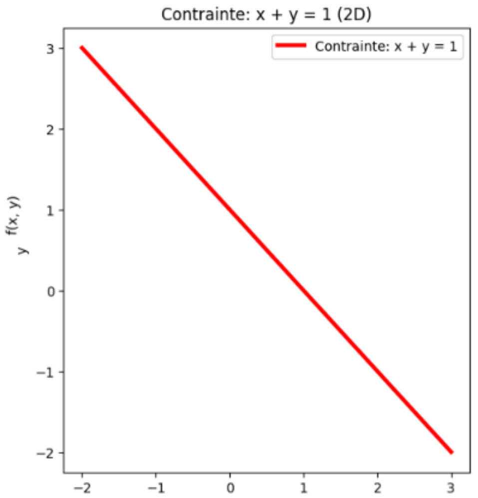
Problèmes d’optimisation sous contrainte d’égalité

- **Optimisation linéaire sous contrainte d'égalité**
- **Exemple.** soir $\begin{cases} f(x,y)=x^2+3y^2 \ (x,y\ ) \\ x+2y=4 \end{cases}$

A=$\begin{pmatrix} 1 & 2 \\ 0 & 0 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix}$- $\begin{pmatrix} 4 \\ 0 \end{pmatrix}$, la contrainte est linéaire et représente une droite.

Le problème n’est pas linéaire





Les points sur la courbe représentent les solutions admissibles du problème d'optimisation.

Problèmes d’optimisation sous contrainte d’égalité

- **Optimisation non linéaire sous contrainte d'égalité**

Un problème d’optimisation avec contraintes d’égalité non linéaire prend la forme suivante :

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

où h : $\mathbb{R}^n \rightarrow \mathbb{R}^p$est différentiable, S = {x ∈ $\mathbb{R}^n$, h(x) = 0}

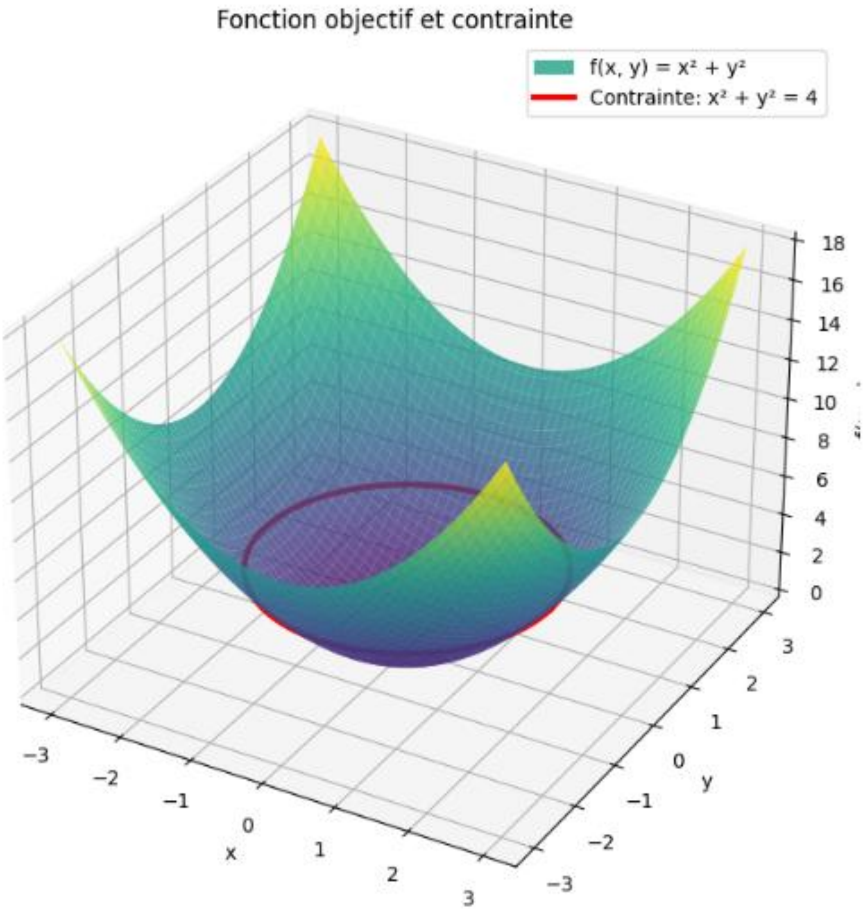
La fonction objectif et/ou les contraintes sont **non linéaires**.

Problèmes d’optimisation sous contrainte d’égalité

- **Optimisation non linéaire sous contrainte d'égalité**

Exemple.

$$\begin{cases} \min\limits_{x \in \mathbb{R}^n} f(\mathrm{x}, \mathrm{y}) = x^2 + y^2, f \; une \; fonction \; objectiv \\ \mathrm{h}(\mathrm{x}) = x^2 + y^2 - \mathbf{4}\,. \end{cases}$$



Les points sur le cercle représentent les solutions admissibles du problème d'optimisation.

Problèmes d’optimisation sous contrainte d’égalité

- **Optimisation quadratique sous contrainte d'égalité (QP)**

Un problème d’optimisation avec contraintes d’égalité quadratique prend la forme suivante :

$$\begin{cases} \min\limits_{x \in \mathbb{R}^n} f(\mathrm{x}) \text{ où } \mathrm{f(x)} = \frac{1}{2}\mathrm{x}^{\mathrm{T}}\mathrm{Qx} + \mathrm{c}^{\mathrm{T}}\mathrm{x} \\ \mathrm{Ax} = \mathrm{b} \end{cases}$$

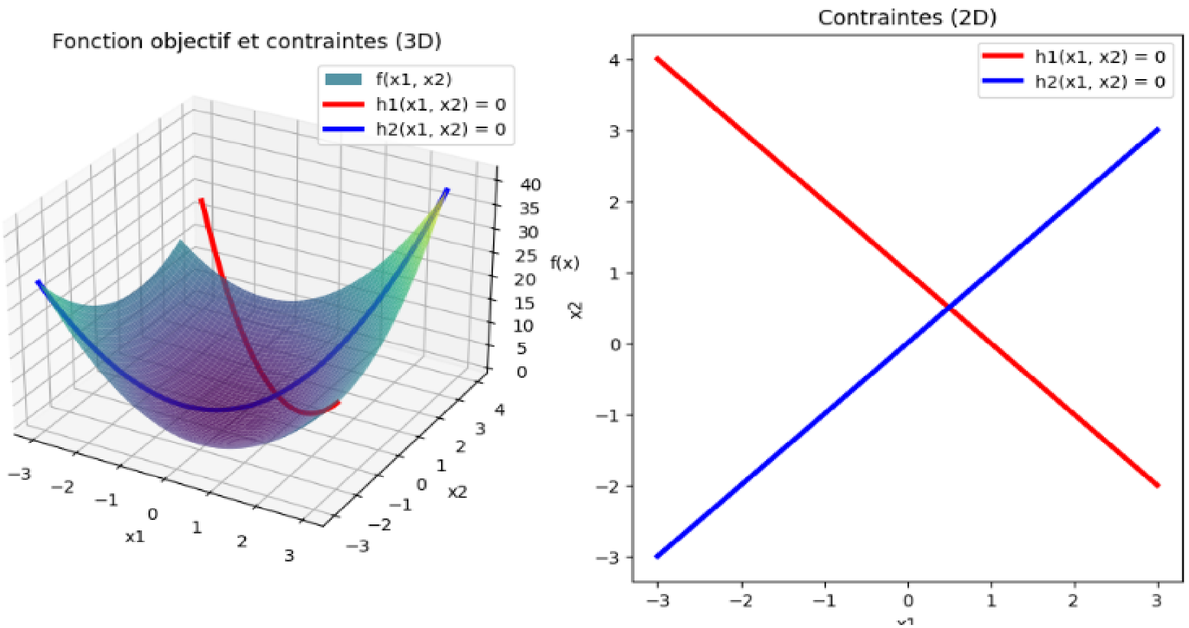
où :

- Q est une matrice$\mathbb{R}^{nxn}$ **symétrique définie positive,** représentant la matrice quadratique de la fonction objective
- c$\in\mathbb{R}^n$ est un vecteur de coefficients linéaires.
- A $\in$ $\mathbb{R}^{nxp}$est la matrice des coefficients des contraintes d'égalité (avec ppp le nombre de contraintes).
- b$\in$ $\mathbb{R}^p$ est le vecteur des termes constants dans les contraintes d'égalité.

Problèmes d'optimisation sous contrainte d'égalité

- **Optimisation quadratique sous contrainte d'égalité (QP)**

Exemple. $\begin{cases} \min\limits_{x\in\mathbb{R}^n} f(\mathrm{x}) \text{ où } \mathrm{f}(\mathrm{x}) = \frac{1}{2}1(4\mathrm{x}_1^2 + 4\mathrm{x}_1\mathrm{x}_2 + 2\mathrm{x}_2^2) + \mathrm{x}_1 + \mathrm{x}_2 \\ sc.\ \mathrm{x}_1 + \mathrm{x}_2 - 1 = 0 \text{ et } \mathrm{x}_1 - \mathrm{x}_2 = 0 \end{cases}$



Problèmes d'optimisation sous contrainte d'égalité

- **Optimisation dynamique sous contrainte d'égalité**
  - L'optimisation dynamique sous contrainte d'égalité est une extension du problème d'optimisation dynamique où on cherche à optimiser un objectif au cours du temps, tout en respectant des contraintes d'égalité qui lient les variables de décision à chaque instant.
  - L'optimisation dynamique est généralement formulée en termes d'un **problème de contrôle optimal**, où l'objectif est de minimiser une fonction de coût au fil du temps tout en respectant des contraintes dynamiques et des contraintes d'égalité.

Problèmes d'optimisation sous contrainte d'égalité

- **Optimisation dynamique sous contrainte d'égalité**
- Le problème classique d'optimisation dynamique sous contrainte d'égalité peut être formulé comme suit : $\min_{\mathbf{u(t)}} \mathrm{J} \int_{t_0}^{t_f} \mathrm{L}(\mathrm{x}(\mathrm{t}), \mathrm{u}(\mathrm{t}), \mathrm{t})\, \mathrm{dt}$.
  - Sous les contraintes :

1. **Équations d'état** (dynamique) : $\dot{\mathbf{x}}(t)=f(x(t),u(t),t)$, $x(t_0)=x_0$.

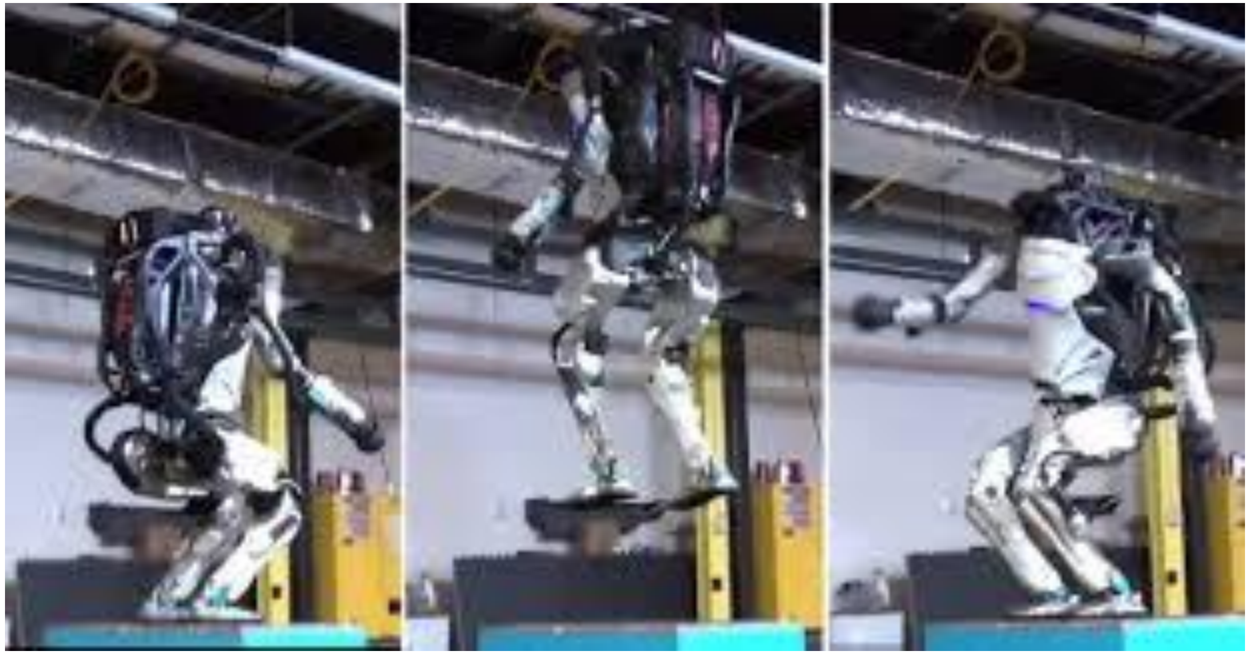
équations différentielles qui régissent l'évolution de l'état x(t) du système au cours du temps. La fonction f(x(t),u(t),t) décrit comment l'état évolue en fonction du vecteur de contrôle u(t)

2. Contrainte d'égalité : $h(x(t),u(t),t)=0, t\in[t_0,t_f]$.

- Où :
  - ✓ J est un critère de performance qui mesure le coût global du système à travers l'intégrale de la fonction de coût instantanée L(x(t),u(t),t)L(x(t), u(t), t)L(x(t),u(t),t)
  - ✓ $x(t)\in \mathbb{R}^n$ est le vecteur d'état du système à l'instant t.
  - ✓ $u(t)\in \mathbb{R}^m$ est le vecteur de contrôle ou de commande à l'instant t.
  - ✓ f(x(t),u(t),t) est la fonction qui décrit la dynamique du système, c.à-d elle décrit comment l'état évolue en fonction du vecteur de contrôle u(t)u(t)u(t) et du temps t.
  - ✓ L(x(t),u(t),t) est la fonction de coût instantanée qui mesure le "coût" à chaque instant.
  - ✓ h(x(t),u(t),t)=0, représente une condition physique ou un objectif à atteindre, est la contrainte d'égalité que l'on impose à chaque instant t.
  - ✓ x_0 est l'état initial du système à $t=t_0$ et t_f est l'instant final.

Problèmes d’optimisation sous contrainte d’égalité

- **Optimisation dynamique sous contrainte d'égalité**
- **Exemple.** Minimiser la consommation d'énergie d'un robot en mouvement sous contrainte de trajectoire.



Méthode d’optimisation des PCE

- Les **méthodes adaptées** à chaque type de problème d'optimisation sous contraintes d'égalité sont principalement :
  - **Méthode de substitutions** pour les problèmes simples
  - **Multiplicateurs de Lagrange** pour les problèmes linéaires, non linéaires et quadratiques.
  - **Méthodes du gradient projeté**
  - **Méthodes de Newton-Raphson** et **SQP** pour les problèmes non linéaires.
  - **Méthode du Simplex** pour les problèmes linéaires.

Méthode d'optimisation des PCE

- **Méthode de résolution par substitution**
  - Le principe de cette méthode de résolution consiste à **substituer directement la contrainte** dans la fonction f.
  - On **exprime une variable** en fonction des autres à partir de la contrainte, on remplace cette variable dans la fonction à minimiser, et on résout ensuite le problème d'optimisation sans contrainte**.**
  - **Exemple**. trouver le minimum de $f(x,y)=x^2+3y^2$ sous la contrainte :

$x+2y=4$.

- ✓ On exprime x en fonction de y : $x=4-2y$.
- ✓ On substitue dans f(x,y) : $f(y)=(4-2y)^2+3y^2 = 7y^2-16y+16$
- ✓ On cherche le minimum par les méthodes analytiques classiques.

$y = 8/7$, $x = 12/7$, $f(x, y) = 48/7$

Méthode d'optimisation des PCE

- **Méthode de résolution par substitution**
  - **Exemple**. trouver le minimum de $f(x,y)=x^2+3y^2$ sous la contrainte : $x+2y=4$.