FSBM

FACULTE DES SCIENCES BEN M'SICK
UNIVERSITE HASSAN II DE CASABLANCA

Algorithmes d’optimisation

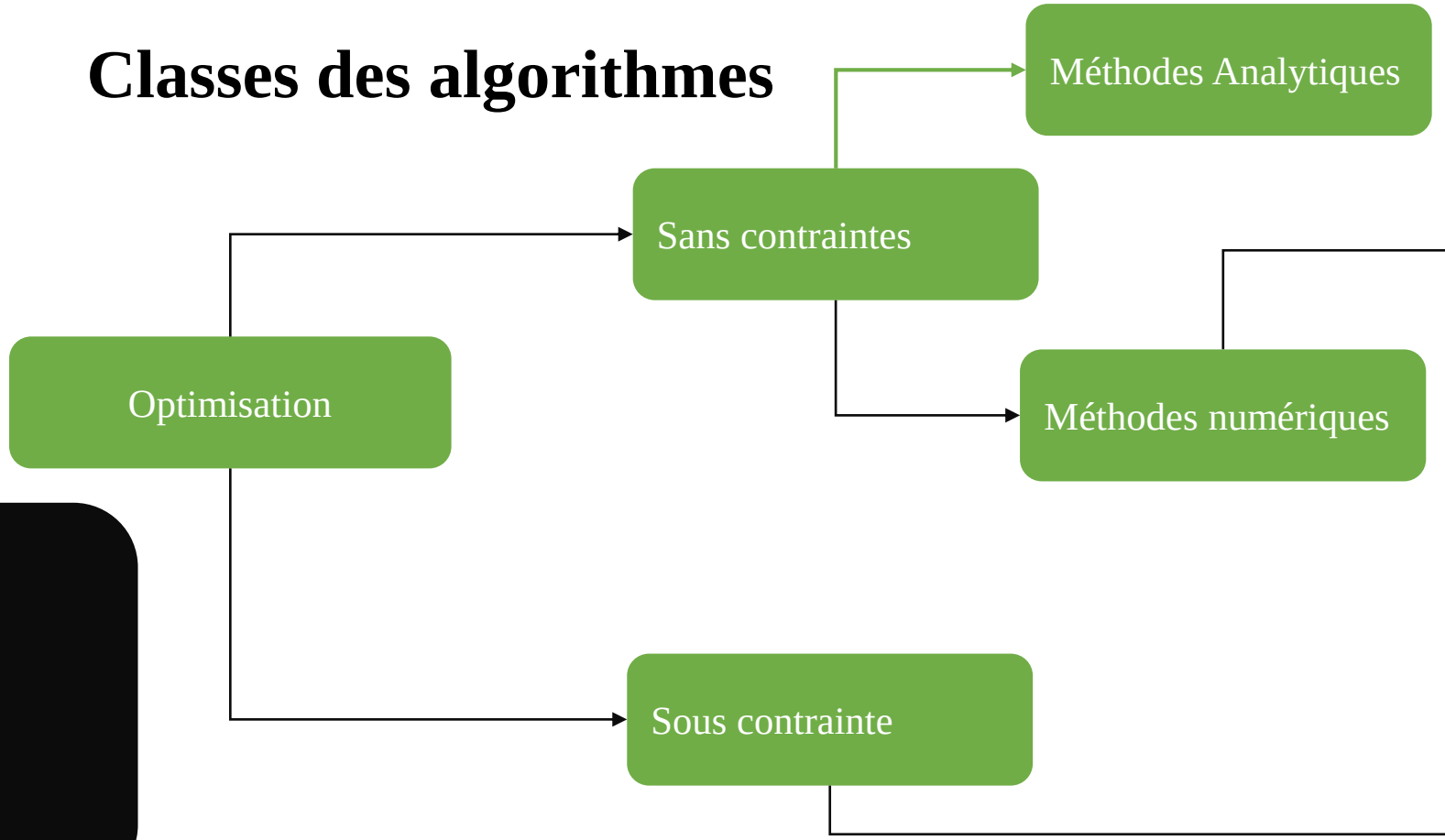
Pr. Faouzia Benabbou (faouzia.benabbou@univh2c.ma)
Département de mathématiques et Informatique
Master Data Science & Big Data
2024-2025

Plan du Module: Algorithmes d’optimisation

1. Introduction
2. Rappels mathématiques
3. Algorithmes d’Optimisation Sans Contraintes
4. Algorithmes d’Optimisation avec Contraintes
5. Programmation Linéaire et non Linéaire
6. Optimisation combinatoire et globale
7. Méthodes heuristiques et métaheuristiques

Les algorithmes d’optimisation

Classes des algorithmes



- Méthode sans gradient
- Méthode de descente de gradient:
  - ✓ Méthode de gradient à pas fixe
  - ✓ Méthode de gradient conjugué
  - ✓ Méthodes du gradient quasi-newton
  - ✓ Méthode de Newton

- Programmation linéaire et non linéaire
- Optimisation combinatoire et globale
- Méthodes heuristiques et métaheuristiques
- Algorithmes multi-objectifs

L’optimisation sans contrainte

- L’**optimisation sans contrainte** est un domaine de l’optimisation où l’on cherche à minimiser ou maximiser une fonction **sans aucune restriction** sur les valeurs que peuvent prendre les variables de décision.
- Le problème peut être formulé de la manière suivante : Soit une fonction f objective : $K \subset \mathbb{R}^n \rightarrow \mathbb{R}$ $n \geq 1$

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

On cherche soit le minimum ou le maximum de la fonction f tel

que : $f(a^*) = \min_{x \in K} f(x)$, $f(a^*) = \max_{x \in K} f(x)$.

L’optimisation sans contrainte

- Il existe plusieurs approches pour résoudre un problème d’optimisation sans contrainte, on peut peut les classer en deux approches:
  - Les **méthodes analytiques** reposent sur le **calcul différentiel** pour identifier les points critiques de la fonction objectif et analyser leur nature.
  - Les **méthodes numériques** en optimisation sont des techniques algorithmiques utilisées pour approximer les solutions de problèmes d'optimisation lorsque les solutions analytiques sont difficiles, voire impossibles à obtenir.

Les méthodes analytiques

- On calcule les dérivées partielles de la fonction et on résout le système d'équations obtenu en étudiant l'équation $\nabla f(X_0) = 0$.
- On utilise ensuite $\nabla^2 f(x_0)$ (matrice hessienne) pour déterminer la nature des points critiques (minimum, maximum, point de selle).
- Cette approche est adaptée aux problèmes simples et aux fonctions régulières.
- Elles ont l'avantage de fournir des solutions exactes (si possible) et de permette une analyse théorique du problème.
- L'inconvénients des méthodes analytiques réside dans la difficulté à les appliquer aux fonctions complexes qui peuvent être non linéaire ou non différentiables.

Les méthodes numériques

- Résoudre l'équation $\nabla^2 f(x^*)=0$, n'est pas toujours possible, dans les cas suivants :
  - Les équations sont **non linéaires** ou trop complexes pour être résolues à la main.
  - Les données sont discrètes ou **bruitées**.
  - Les problèmes impliquent un **grand nombre** de variables.
- C'est pourquoi on est amenée à chercher une valeur approchée de x*.
- Pour construire cette approximation, nous allons utiliser des algorithmes itératifs.
- Les méthodes d'optimisation itératives construisent une suite (x_n) $n \in \mathbb{N}$ qui converge vers $x*$.

Les méthodes numériques

- **Définition.** Un algorithme itératif est défini par une application vectorielle f : $\mathbb{R}^n \rightarrow \mathbb{R}^n$ qui génère une suite de vecteurs $(x_n)n \in \mathbb{N}$, à l'aide d'une construction de la forme:

$$\begin{cases} \text{Choisir } x_0 \in \mathbb{R}^n \text{ (phase d'initialisation de l'algorithme)} \\ \text{Calculer } x_{n+1} = f(x_n) \text{ (nième itération)} \end{cases}$$

- Ce que l'on espère, c'est que la suite $(x_n)n \in \mathbb{N}$, converge vers l'optimum cherché.

Les méthodes numériques

- **Convergence.** La vitesse de convergence d'un algorithme itératif est la vitesse de convergence de la suite (x_n) vers $x*$.
- Cette vitesse peut-être très différente de la vitesse réelle de l'algorithme, qui prend aussi en compte le temps que met l'ordinateur à évaluer la fonction f par exemple.
- Cependant, le temps physique de l'algorithme est évidemment proportionnel au nombre d'itérations nécessaires pour obtenir une approximation à ϵ près (ϵ fixé à l'avance) de l'optimum x*.

Les méthodes numériques

- **Définiton convergence.** Supposons connue une suite $(x_n)_{n\in N}$, obtenue à l'aide d'un algorithme itératif, et telle que $\lim_{n\to+\infty} x_n = x^*$. sachant que ||.|| désigne une norme.
  - On dira que la vitesse de convergence de l'algorithme est linéaire si $\exists\ C \in [0, 1[$ tel que : $\forall n \in N\ \|\boldsymbol{x}_{n+1} - \boldsymbol{x}*\| \leq C\|\boldsymbol{x}_n - \boldsymbol{x}*\|$; La constante C détermine la vitesse de convergence, plus C est proche de 0, plus la convergence est rapide.
  - Une suite $(\mathbf{x}_n)_{n\in N}$ converge superlinéairement vers une limite x* si :

$$\lim_{n\to\infty}\left(\frac{||x_{n+1} - x*||}{||x_n - x*||}\right) = 0$$

  - Une suite $(\mathbf{x}_n)_{n\in N}$ converge quadratiquement vers une limite x* si :

$$\forall n \in N\ \ ||x_{n+1} - x*|| \leq C||x_n - x*||^2$$

Les méthodes numériques

- La convergence peut être aussi globale ou locale.
- Dans convergence globale l'algorithme converge quel que soit le point de départ x_0, alors que dans la convergence locale l'algorithme ne converge que si on démarre suffisamment près de $x*$.

Type de convergence	vitesse
convergence linéaire	Relativement lente.
converge superlinéairement	Plus rapide que la convergence linéaire
Convergence quadratique	Très rapide

Les méthodes numériques

- **Complexité.** La complexité calculatoire, mesure le coût des opérations nécessaires pour obtenir une itération.
  - Le coût global est donné par le coût d'une itération multiplie par le nombre d'itérations nécessaires pour obtenir la solution escomptée avec une certaine précision ϵ.
- **Critère d'arrêt.** Le critère d'arrêt peut se baser aussi sur :
  - La précision ϵ est associée à un critère d'arrêt, permettant à l'algorithme de s'arrêter et de fournir une valeur approchée x_n de l'optimum, que l'on jugera acceptable.
  - Le nombre maximal d'itérations dans un algorithme d'optimisation dépend de plusieurs facteurs, notamment de la complexité du problème et de la vitesse de convergence.
  - La variation de la fonction objectif $||f(x_{k+1}) - f(x_k)||$
  - La variation des points $||x_{k+1} - x_k||$
  - La norme du gradient $||\nabla f(x_k)||$

Méthodes d’optimisation directe (sans gradient)

Méthodes univariable: Méthode de dichotomie

- La méthode de dichotomie appelée aussi méthode de **bissection** est une méthode de résolution des équations non linéaires.
- Cette méthode est basée sur le théorème des valeurs intermédiaires.

  Soit $f : [a, b] \rightarrow \mathbb{R}$ une fonction **continue**, alors f prend toutes les valeurs intermédiaires entre f(a) et f(b); En particulier, si f est telle que $f(a)f(b) < 0$, alors il existe $\boldsymbol{\alpha} \in]a, b[$ tel que $\mathbf{f(\alpha) = 0.}$

- Nous utilisons la méthode de dichotomie pour résoudre l’équation f(a)=0.

Méthodes d'optimisation directe (sans gradient)

- **Méthodes univariable: Méthode de dichotomie**

Algorithme 1 : Dichotomie

1.On pose n=0,a_n = a, b_n = b, ϵ =1e-6 (10^{-6}) ϵ est la tolérance, max_iter : nombre max d'iterations.

2. Répéter

a) $x_n = \frac{a_n + b_n}{2}$.

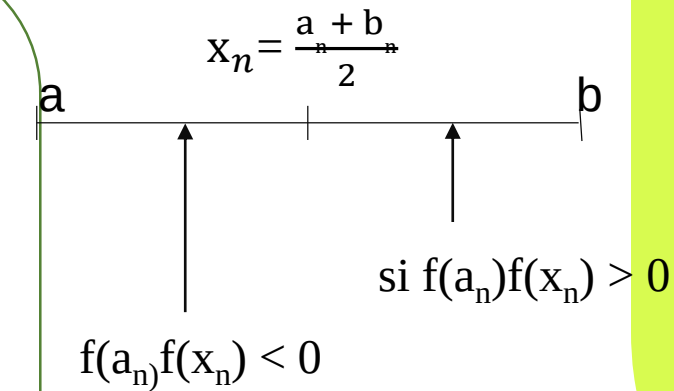
b) si **f(x_n)=0**, alors x_n est la solution

c) si $f(a_n)f(x_n) > 0$ on pose $a_{n+1} = x_n$, $b_{n+1} = b_n$

d) sinon ($f(a_n)f(x_n) < 0$), on pose $a_{n+1} = a_n$, $b_{n+1} = x_n$

e) n++

3. jusqu'à ce que $|b_n - a_n| < \epsilon$ et et n<max_iter.



Méthodes d'optimisation directe (sans gradient)

Méthodes univariable: Méthode de dichotomie

- **Exemple.** Soit $f(x)= x^3 - 4x + 1$, continue sur l'intervalle [1,2].
  - Chercher un x* tel que f(x*)=0.
  - nombre d'itération =100,
  - ϵ=tolérence=1e-6,
  - Tracer la courbe de f.
- La méthode converge après 21 itérations et la solution approchée est $x \approx 1.860805988311767 6$.
- On voit bien que l'équation a 3 solutions, en variant l'intervalle [a,b] on peut les approcher.

Méthodes d’optimisation directe (sans gradient)

Méthodes univariable: Méthode de dichotomie

- **Limites**
  - Convergence lente par rapport à d’autres méthodes
  - Nécessite un intervalle initial contenant une racine : Il faut connaître un intervalle [a,b] tel que $f(a)\cdot f(b)<0$. Si cette information est indisponible, la méthode ne peut pas être appliquée.
  - Ne fonctionne que pour les fonctions continues
  - La méthode ne garanti pas un optimum global.

Méthodes d’optimisation directe (sans gradient)

Méthodes univariable: Méthode du nombre d’or ou la section dorée

- Cette méthode est utile lorsque la fonction est complexe ou coûteuse à évaluer.
- On cherche une solution pour le problème f(x*)=0.
- Elle se base sur le nombre d’or $\varphi = \frac{1+\sqrt{5}}{2} \approx 1.618$
- La méthode du nombre d'or repose sur le principe de la section dorée, qui divise un intervalle en deux parties dans un rapport spécifique lié au nombre d’or.
- Ce rapport permet de réduire l'intervalle de recherche de manière efficace à chaque itération, garantissant une convergence rapide vers le minimum de la fonction sur cet intervalle.

Méthodes d'optimisation directe (sans gradient)

Méthodes univariable: Méthode du nombre d'or ou la section dorée

Algorithme 2 : du nombre d'or

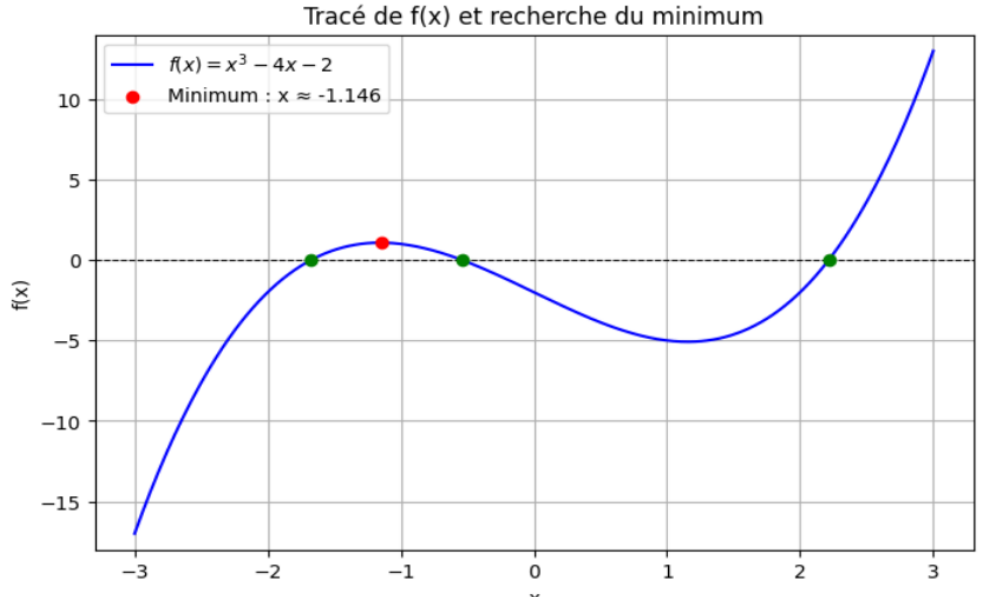
1. Soit f une fonction, [a,b] un intervalle.
   ϵ =1e-6 (10^{-6}) ϵ est la tolérance
2. Répéter
   a) Calculer : $\boldsymbol{x_1 = b - \frac{(b-a)}{\varphi^2}}$, $\boldsymbol{x_2 = a + \frac{(b-a)}{\varphi^2}}$,
   b) Évaluer la fonction aux points de division : f($\boldsymbol{x_1}$) et f($\boldsymbol{x_2}$).
   c) Si f(x_1) < f(x_2), alors le minimum se trouve dans l'intervalle [a, x_2].
      Mettre à jour l'intervalle : $b = x_2$
   d) Sinon, le minimum se trouve dans l'intervalle [x_n, b_n].
      Mettre à jour l'intervalle : $a = x_1$.
3. Jusqu'à (b – a < ϵ) .

Le minimum approximatif est le point médian de l'intervalle final : (a + b) / 2.

Méthodes d’optimisation directe (sans gradient)

Méthodes univariable: Méthode du nombre d’or ou la section dorée

- **Exemple.** Soit f(x)= x^3 - 4x - 2, ϵ= 1e-6, intervalle [-3,0]. La méthode converge vers x≈ 1.154700.
- La courbe montre les points où la fonction et le point en rouge le minimum trouvé par cette méthode.

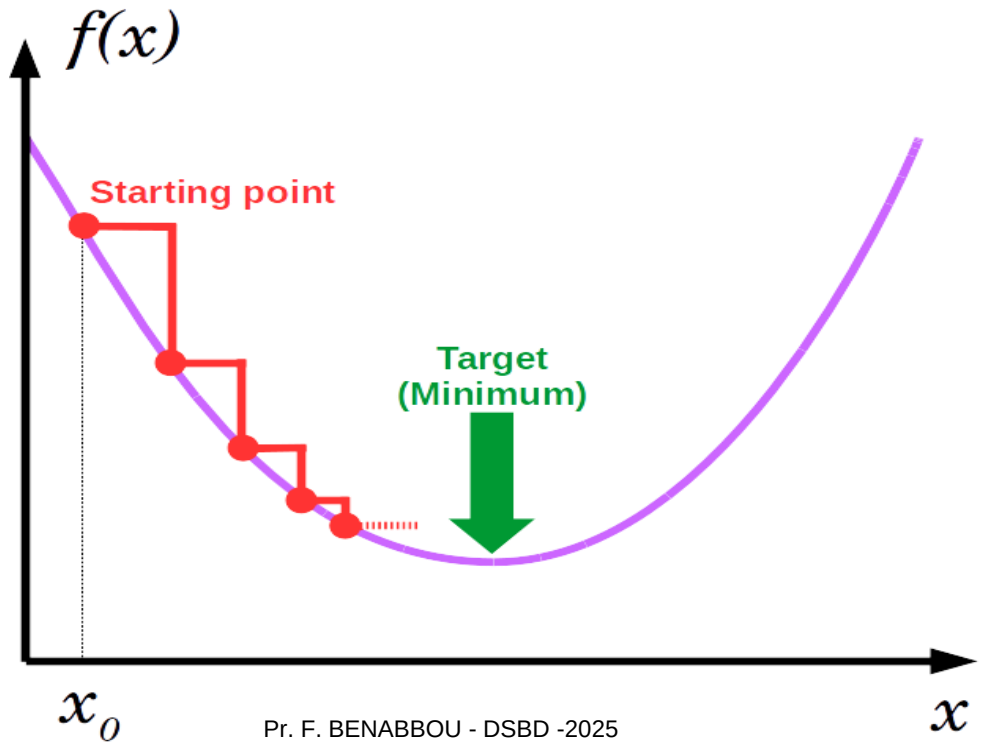


Méthodes d’optimisation de descente gradient

- Les algorithmes de descente constituent une famille de méthodes d'optimisation itératives, conçues pour minimiser des fonctions réelles différentiables.
- Vous êtes perdu dans la montagne !

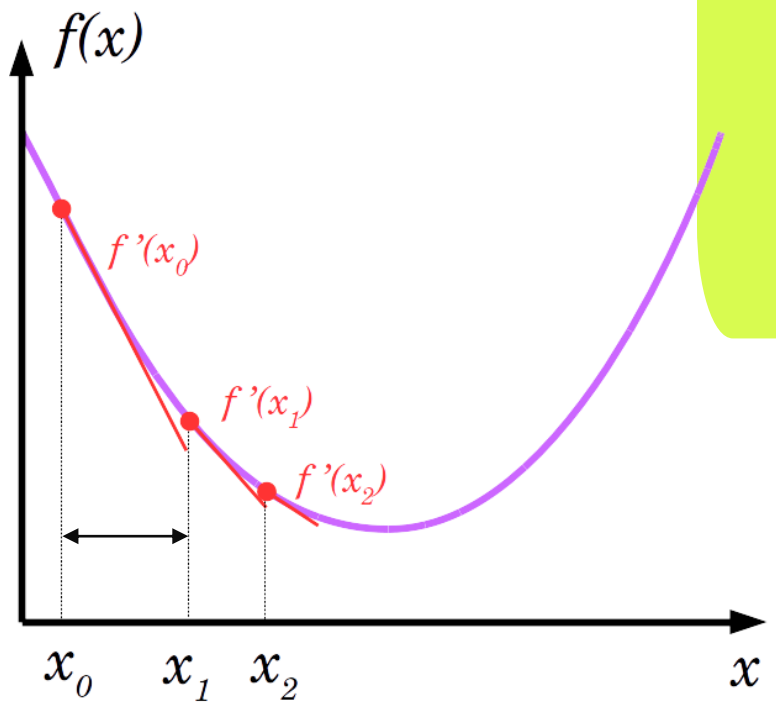
Méthodes d’optimisation de descente gradient

- Considérons la fonction dérivable f(x) que l'on souhaite minimiser.
- L'algorithme de descente de gradient démarre à une coordonnée initiale arbitraire et converge vers le minimum de façon itérative.



Méthodes d'optimisation de descente gradient

- Soit f: $\mathbb{R} \rightarrow \mathbb{R}$, et x_0 le point de départ de l'algorithme.
- Pour déterminer le point suivant x_1, la descente de gradient calcule la dérivée **$f'(x_0)$.**
- La dérivée étant la pente de la tangente en ce point, elle détermine une direction de descente, qui est une **direction** dans laquelle la fonction objective **diminue**.
- Un déplacement est effectué le long de cette direction du gradient selon la **longueur du pas**.
- Le point suivant est alors calculé grâce à la formule suivante : $x_{k+1} = x_k + \alpha \frac{df(x_k)}{dx}$
  - $\frac{df(x_k)}{dx}$ est la direction de descente
  - α est le pas



Méthodes d'optimisation de descente gradient

Généralisation

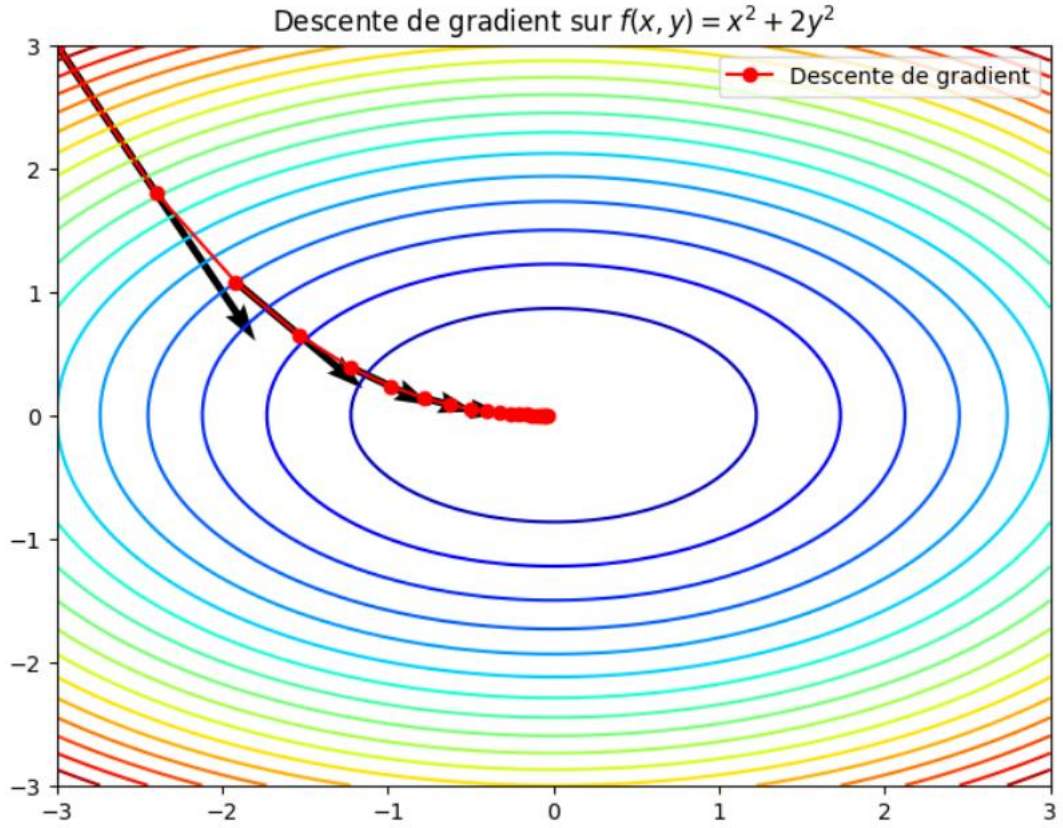
- Soit f: f : $\mathbb{R}^n \rightarrow \mathbb{R}$, et x_0 le point de départ de l'algorithme.
- La dérivée est alors remplacé par le gradient de la fonction.
- C'est la raison pour laquelle cet algorithme s'appelle la descente du gradient.
- La généralisation de l'algorithme de descente de gradient est donnée par l'équation suivante:

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

- Le choix de la **direction de descente** et la détermination du **pas** sont des éléments cruciaux pour l'efficacité et la rapidité de convergence des algorithmes de descente.

Méthodes d'optimisation de descente gradient

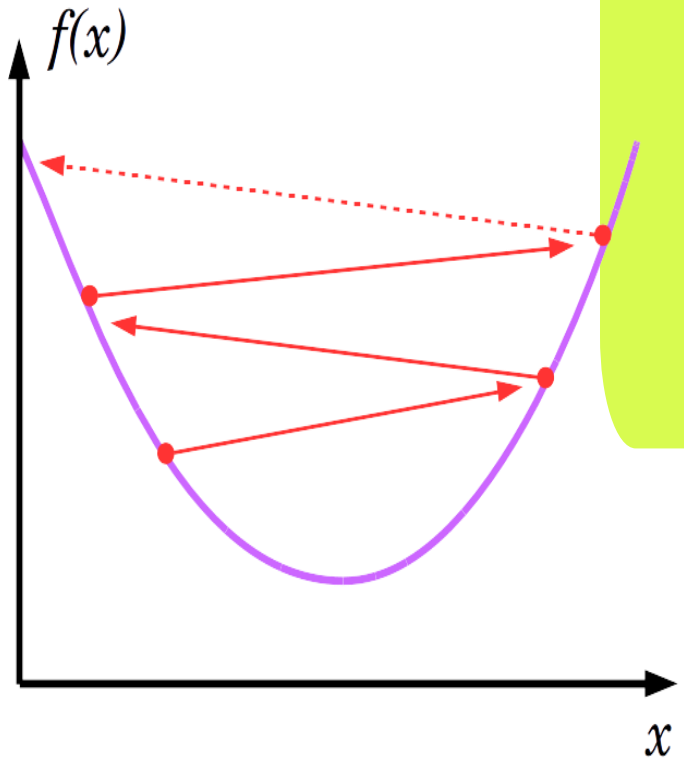
- **Exemple.** Soit f: f : $(x, y) \rightarrow x^2 + 2y^2$,



α = 0.1 , Taux d'apprentissage
itérations = 20 , Nombre d'itérations
x0, y0 = -3.0, 3.0 , Point de départ

Méthodes d’optimisation de descente gradient

- **Pas de l'algorithme.** Le pas de l'algorithme α_k est un paramètre qui doit être choisi soigneusement.
- pour un pas élevé ce paramètre, il se peut que l’algorithme commence à osciller autour du minimum sans jamais l’atteindre.
- De petites valeurs assurent plus de stabilité à l'algorithme mais la méthode risque de prendre un temps important avant de converger.
- Sur l'illustration ci-dessous, un pas trop grand empêche l'algorithme de converger vers le minimum.



Méthodes d’optimisation de descente gradient

- Pas de formule magique pour trouver α_k , il faut tâtonner
- La majorité des réseaux de neurones artificiels s'appuie sur l'algorithme de descente de gradient.
- α est le facteur d'apprentissage (η), learning rate en anglais.
- La descente de gradient, peut converger vers des minimas locaux
- Il existe plusieurs variantes de cet méthode
  - Méthode de gradient à pas fixe, à pas optimal
  - Méthode de gradient conjugué
  - Méthode de Newton
  - Méthodes de gradient quasi-newton

Méthodes d'optimisation de descente gradient

Méthode de gradient à pas fixe

Dans le cas de l'algorithme de descente à pas constant , le pas est constant, c'est-à-dire $\alpha_k = \alpha \ \forall \ k \in N$

Algorithme 3 : descente de gradient à pas fixe

1. Initialisation : $x_0 \in$ Rn, f une fonction objective de classe 1, $\alpha \in$R+*.
ε : critère d'arrêt (tolérance sur le gradient), max_iter :nombre maximal d'itérations
2. Répéter
    1.Calculer la direction de la descente $d_k = - \nabla f(x_k)$.
    2.Mettre à jour la solution : $x_{k+1} = x_k + \alpha \ d_k$
    3.k++
3. Jusqu'à Vérifier la condition d'arrêt : $\|\nabla f(x_n)\| > \varepsilon$ et k<max_iter

Méthodes d'optimisation de descente gradient

Méthode de gradient à pas fixe

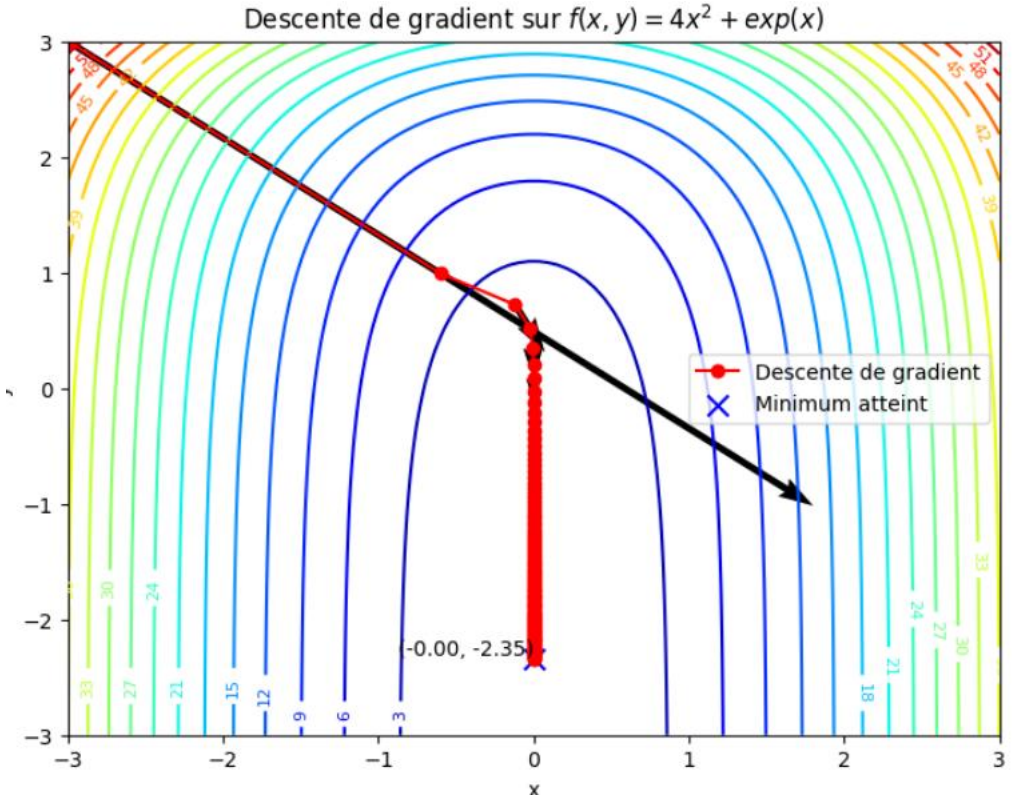
- Les critères d'arrêt sont très important dans un algorithme de descente, voici des exemples :

Type de critère	Condition	Avantage	Inconvénient
Critère principal	$\|\nabla f(x_n)\| < \varepsilon$	Bonne indication de la convergence	Peut s'arrêter sur un point de selle
Critère secondaire	$\| f(x_{n+1}) - f(x_n) \| < \eta$	Permet d'arrêter l'algorithme lorsque l'amélioration est négligeable.	Peut être trompeur si la fonction a un plateau où f(x) varie peu
Critère secondaire	$\| x_{n+1} - x_n \| < \alpha$	Évite les itérations inutiles quand le changement est négligeable	Peut poser problème si α est trop petit, car cela pourrait arrêter prématurément l'algorithme.
Sécurité	n ≥ max_iter	Toujours garanti de s'arrêter	Peut être trop strict
Critère mixte	$\|\nabla f(x_n)\| < \varepsilon$ et $\| f(x_{n+1}) - f(x_n) \| < \eta$	Plus robuste que l'un des critères seuls	Nécessite de régler deux seuils ε et η au lieu d'un.

Méthodes d'optimisation de descente gradient

Méthode de gradient (pas fixe, pas adaptatif)

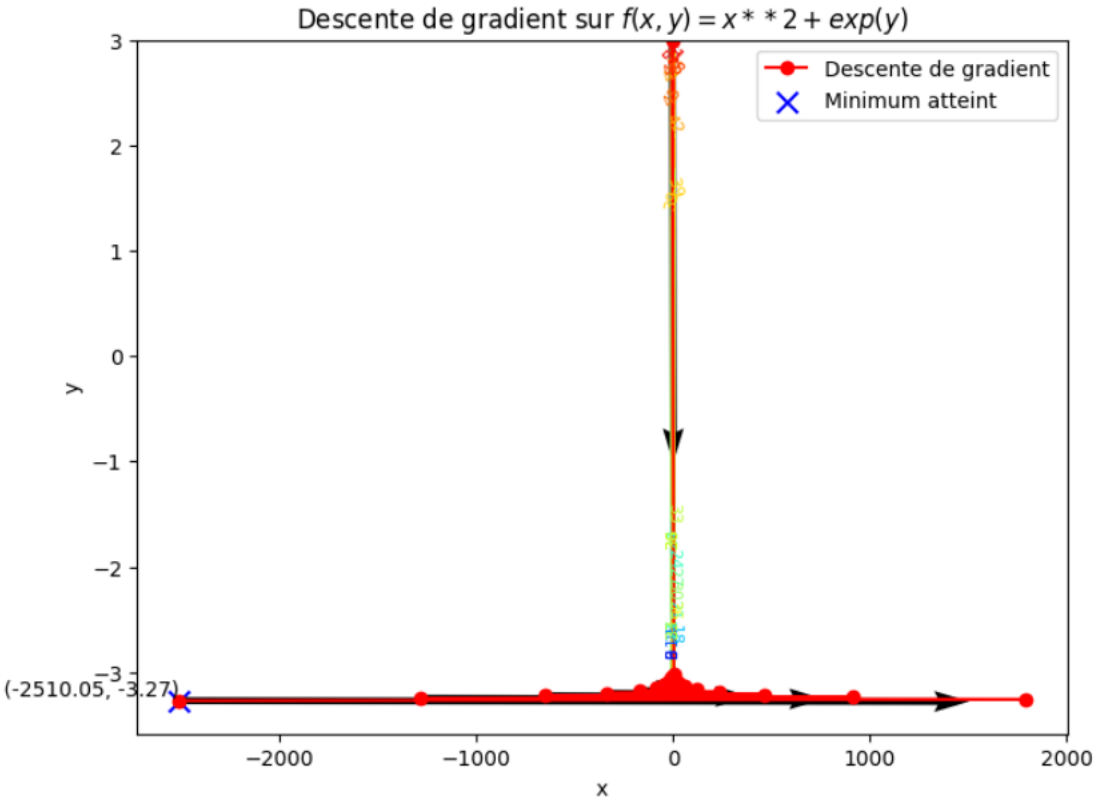
- **Exemple** . Chercher le minimum de la fonction réelle f : $f(x) = 4\,x^2 + e^x$.
  - calcul du gradient de f(x) : $\nabla f(x) = 8x + e^x$.
  - Soit $\alpha = 0.01$, iterations = 20, $x_0, y_0 = -3, 3$.



Méthodes d'optimisation de descente gradient

Méthode de gradient (pas fixe, pas adaptatif)

- **Exemple** . Chercher le minimum de la fonction réelle f : $f(x) = 4\,x^2 + e^x$.
  - calcul du gradient de f(x) : $\nabla f(x) = 8x + e^x$.
  - Soit $\alpha = 0.4$, iterations = 100, $\varepsilon = 10^{-6}$, $x_0, y_0 = -3, 3$.



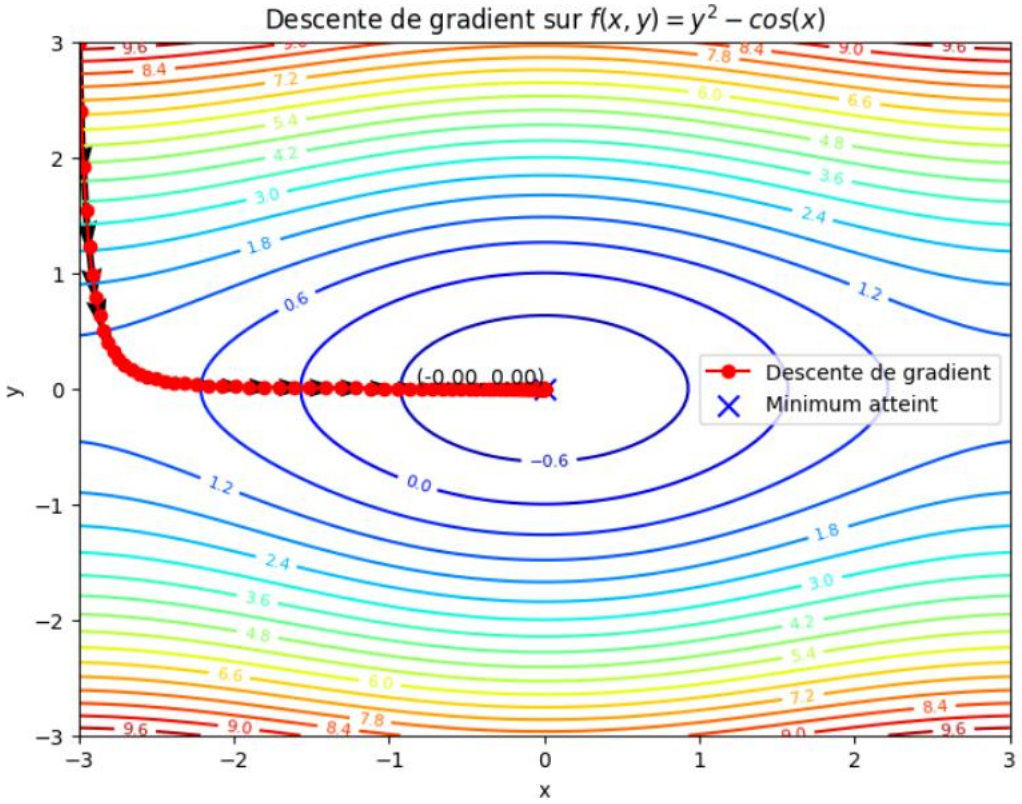
Méthodes d’optimisation de descente gradient

Méthode de gradient (pas fixe, pas adaptatif)

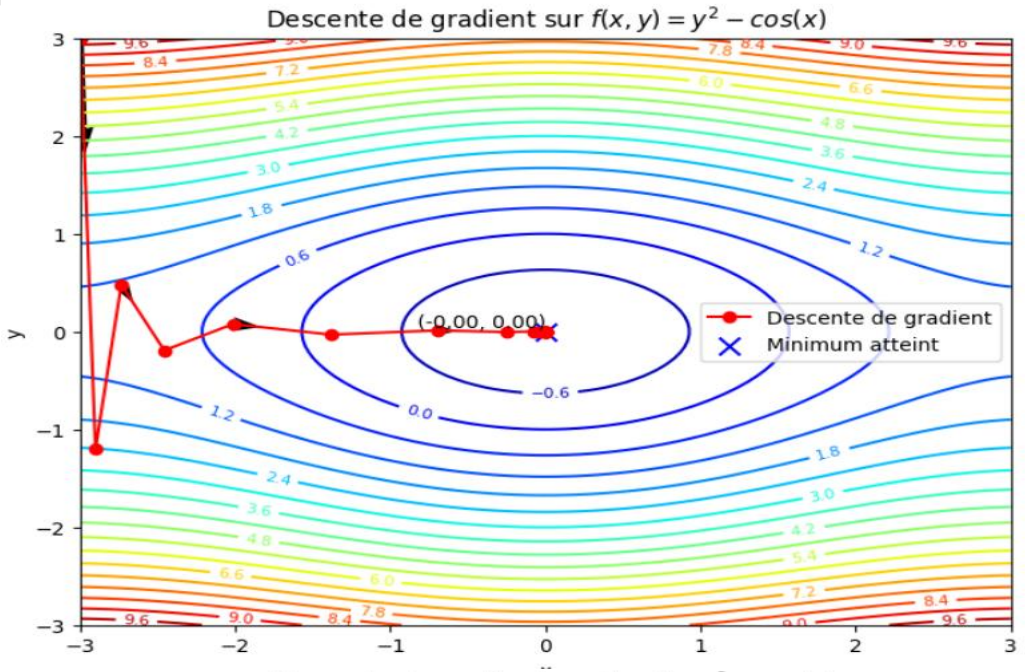
- **Exercice .** Trouver le minimum de la fonction

$$f(x,y) = \cos(2\,x)\sqrt{y^2\ +\ 1},$$

α =0,1, ε = 1e-6 max_iterations = 100 x0, y0 = -3.0, 3.0



α =0,7



α =0,9