Optimisation Non linèaire

Par

Professeur Abdellatif El Afia

Chapitre2

Optimisation sans contrainte

- Fonctions à une seule variable
- Fonctions à plusieurs variables

Fonctions à une seule variable

1. Rappel de certains résultats de calcul
2. Algorithmes d'optimisation

Rappel de certains résultats de calcul

- Line search: $\min_{\alpha \in \mathbb{R}} \varphi(\alpha)$
- $f: \mathbb{R}^n \to \mathbb{R}$
- $\min_{x \in \mathbb{R}^n} f(x)$
- $\{x^k \in \mathbb{R}^n\}$
  - x^0 point initiale
  - Iitération k: x^k
  - Calcule de d^k : Fonctions à plusieurs variables
  - $\varphi: \mathbb{R} \to \mathbb{R}$ $telque\ \varphi(\alpha) = f(x^k + \alpha d^k)$
  - $\alpha_k = arg\min_{\alpha > 0} \varphi(\alpha)$: Fonctions à une seule variable
  - $x^{k+1} = x^k + \alpha_k d^k$

Rappel de certains résultats de calcul

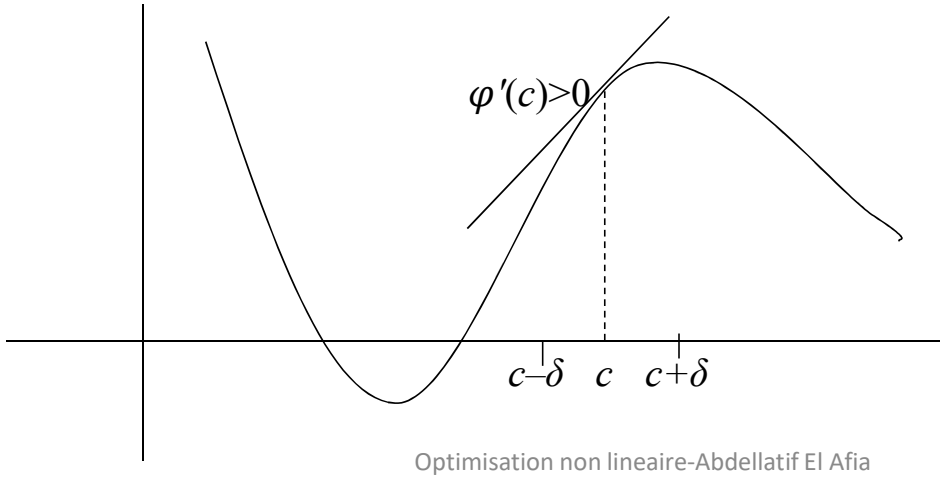
Résultat 1: Soit $c \in \mathbb{R}^1$ tel que: Si $\varphi'(c) = \frac{d\varphi(c)}{d\alpha} = \lim_{\varphi \to 0} \frac{\varphi(\alpha)-\varphi(c)}{\alpha-c} \neq 0$ alors c est ni un minimum local ni un maximum local de φ.

Résultat 2: Soit $c \in \mathbb{R}$

si $\varphi'(c) > 0$, il existe un intervalle $(c-\delta, c+\delta)$ avec $\delta > 0$ tel que:

$$\varphi(\alpha) < \varphi(c) \quad \forall \alpha \in (c-\delta, c)$$
$$\varphi(c) < \varphi(\alpha) \quad \forall \alpha \in (c, c+\delta)$$

et la fonction φ est croissante au point au point c



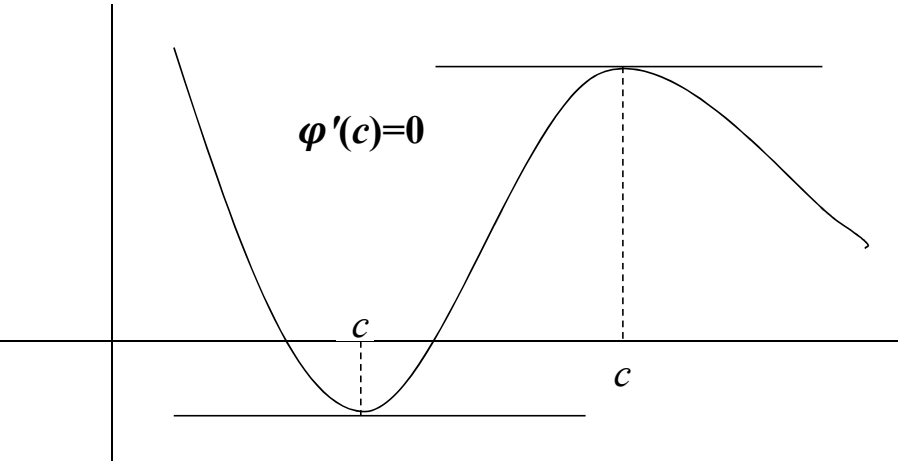
Rappel de certains résultats de calcul

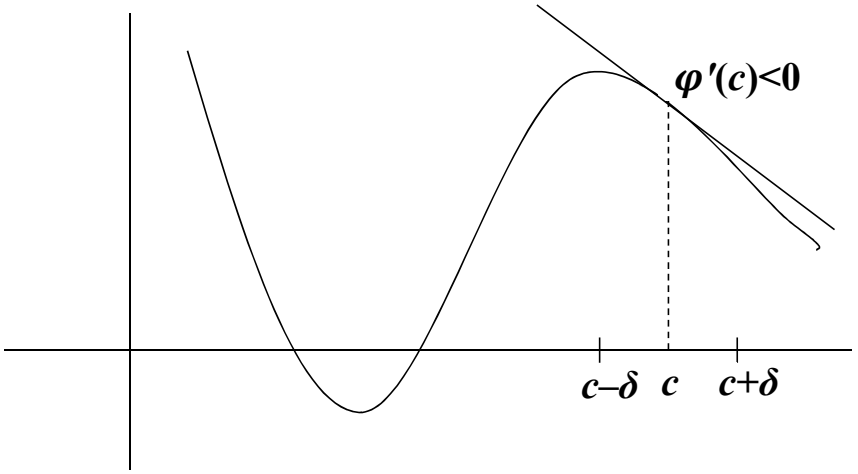
Résultat 4: Soit $c \in \mathbb{R}$
si $\varphi'(c) = 0$, alors c est un minimum ou un maximum local (ou un point d'inflexion) de φ

Résultat 3: Soit $c \in \mathbb{R}$
si $\varphi'(c) < 0, \exists \delta > 0$ tel que:

$$\varphi(\alpha) > \varphi(c) \quad \forall \alpha \in (c-\delta, c)$$
$$\varphi(c) > \varphi(\alpha) \quad \forall \alpha \in (c, c+\delta)$$

et la fonction φ est décroissante au point au point c





Rappel de certains résultats de calcul

Résultat 5: (Test de la dérivé seconde)

Soit $c \in \mathbb{R}$tel que $\varphi'(c) = 0$. Supposons également que la dérivée $\varphi'(\alpha)$ existe $\forall \alpha \in B_\epsilon(c)$

1) Si $\varphi''(c) < 0$, alors c est un maximum local de φ.
2) Si $\varphi''(c) > 0$, alors c est un minimum local de φ.

Note: $\varphi''(c)$ dénote la dérivée seconde de f à c:

$$\varphi''(c) = \frac{d\varphi'}{d\alpha}(c) = \lim_{x \to c} \frac{\varphi'(\alpha) - \varphi'(c)}{\alpha - c} = \frac{d^2\varphi}{d\alpha}(c).$$

Justification intuitive de 2) :

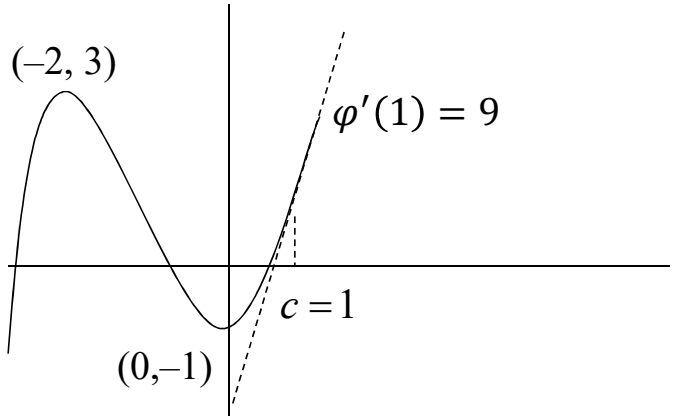
- Si $\varphi''(c) > 0$, alors φ' est une fonction croissante à c et par conséquent c

est un minimum local.

Rappel de certains résultats de calcul

Exemple:

- $\varphi(\alpha) = \alpha^3 + 3\alpha^2 - 1 \Longrightarrow \varphi'(\alpha) = 3\alpha^2 + 6\alpha = 3\alpha(\alpha + 2)$
- $\varphi'(\alpha) = 0 \Longrightarrow \alpha = 0$ ou $\alpha = -2$
- $\varphi''(\alpha) = 6(\alpha + 1) \Longrightarrow \varphi''(0) = 6 > 0$ et $\varphi''(-2) = -6 < 0$



Si $c = 1, \varphi'(1) = 3 + 6 = 9 > 0,$
d'après résultat 1
soit $\delta = 1$, Considérons l'intervalle $(1 - 1,1 + 1) = (0,2)$.

- Si $\alpha \in (0,1)$, alors $\varphi(\alpha) < \varphi(1)$
- Si $\alpha \in (1,2)$, alors $\varphi(\alpha) > \varphi(1)$

d'après les résultats 2, 3, 4, 5

- $\alpha = -2$ est un maximum local et $\varphi'(-2) = 0, \varphi''(-2) = -6$
- $\alpha = 0$ est un minimum local et $\varphi'(0) = 0, \varphi''(0) = 6$

Algorithme d'optimisation

- **Algorithme de bissection**
- **Algorithme de Newton-Raphson**
- **Algorithme de fausse position**

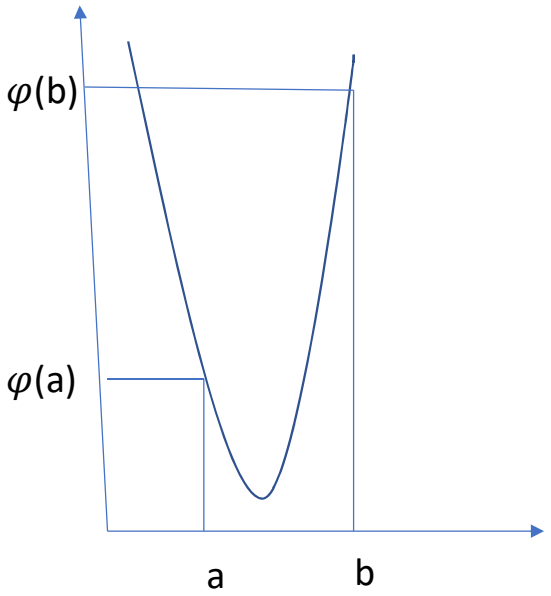
Algorithme de bissection

Principe de la méthode:

à chaque itération, réduire la longueur de l'intervalle contenant c en la divisant en deux.

Line search : $\min_{\alpha \in \mathbb{R}} \varphi(\alpha)$

- Input :
  - **H1: φ est une Fonction unimodale**
  - **H2: $[a, b]$ tel que $\varphi'(a)\varphi'(b) < 0$**
  - $\delta(= 10^{-5})$ tolérance
- While $|a - b| > \delta$
  - $c = \frac{a+b}{2}$
  - If $\varphi'(c) = 0$ et $\varphi''(\mathrm{c}) > 0$ stop
  - Else Mise à jour de a $(\varphi'(c) \leq 0)$ et b
- Output:
  - $[\bar{a}, \bar{b}]$ ou c tel que $\varphi'(\mathrm{c}) = 0$

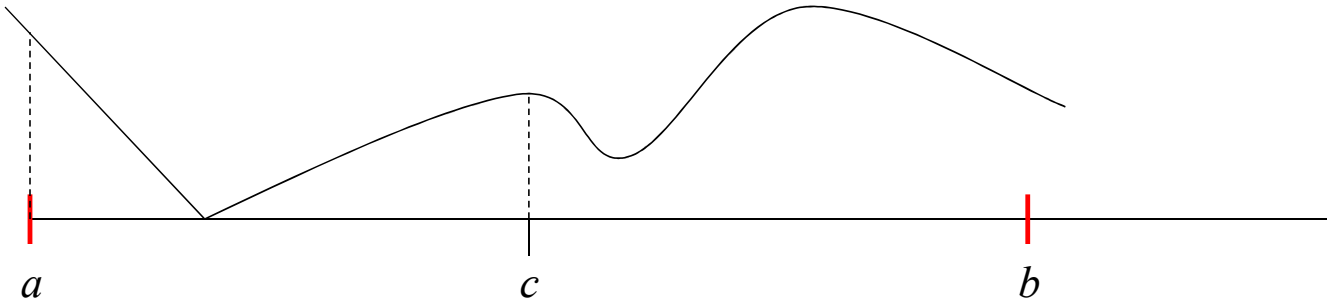


Algorithme de bissection: Exemple

- Input:
  - $\varphi(\alpha) = \alpha^3 + 3\alpha^2 - 1$
  - $\varphi'(\alpha) = 3\alpha^2 + 6\alpha = 3\alpha(\alpha + 2) \Longrightarrow \varphi''(\alpha) = 6\alpha + 6$
  - $[a = -1, b = 1] \Longrightarrow \varphi'(-1) = -3 < 0$ et $\varphi'(1) = 9 > 0$
  - $\delta = 0{,}001$
- Boucle: $|a - b| = 2 > \delta$
  - $c = \frac{a+b}{2} = \frac{-1+1}{2} = 0$
  - $\varphi'(c) = \varphi'(0) = 0$ et $\varphi''(0) = 6 > 0$ stop
- $c = 0 = \underset{\alpha \in [-1,1]}{arg\min}\, \varphi(\alpha)$

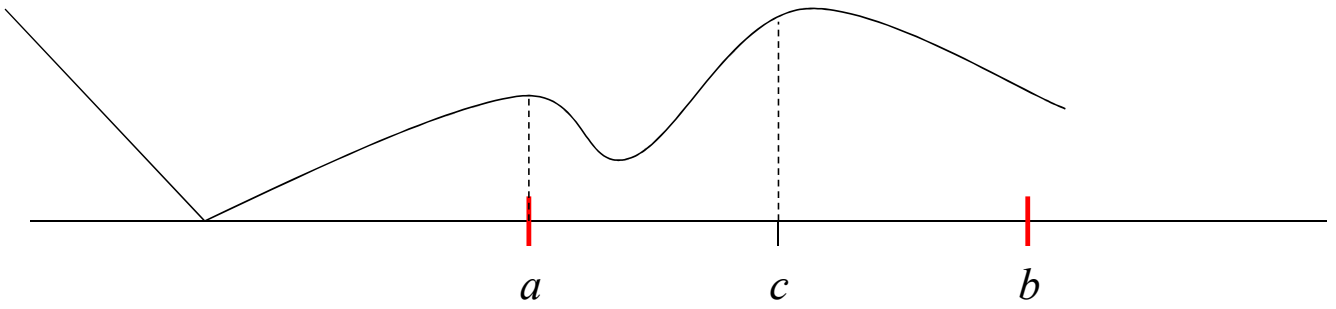
Algorithme de bissection (importance d'hypothèse)

- H2: Sur l'intervalle $[a, b]$, la fonction $\boldsymbol{\varphi'(\alpha)}$ **est continue et telle que** $\boldsymbol{\varphi'(a)\varphi'(b) < 0}$ (i.e., il existe $\bar{\alpha} \in [a, b]$ où $\varphi'(\alpha_k) = 0$)
- Hypothèse $\varphi'(a)\ \varphi'(b)$<0 est essentielle si $\varphi'(a)\ \varphi'(b) > 0$
- $\forall \alpha :\ \varphi'(\alpha) \geq 0$



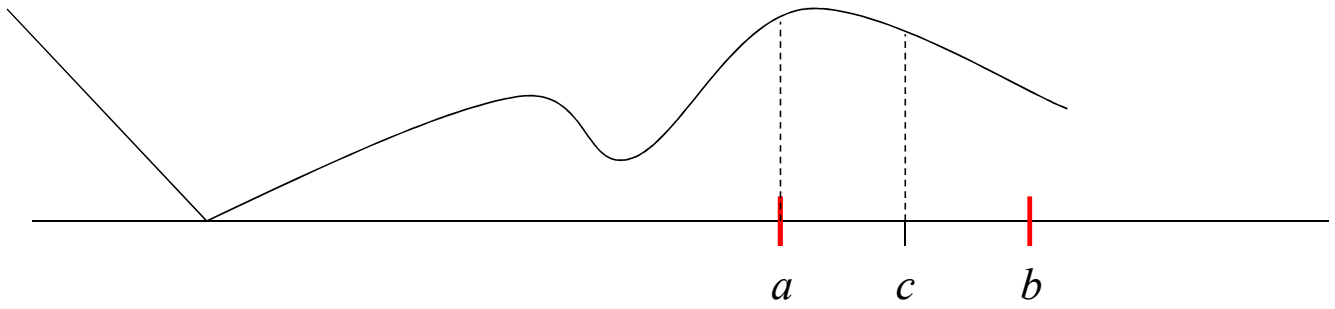
Algorithme de bissection (importance d'hypothèse)

- Hypothèse $\varphi'(a)\ \varphi'(b)<0$ est essentielle
- $\forall \alpha :\ \varphi'(\alpha) \geq 0$



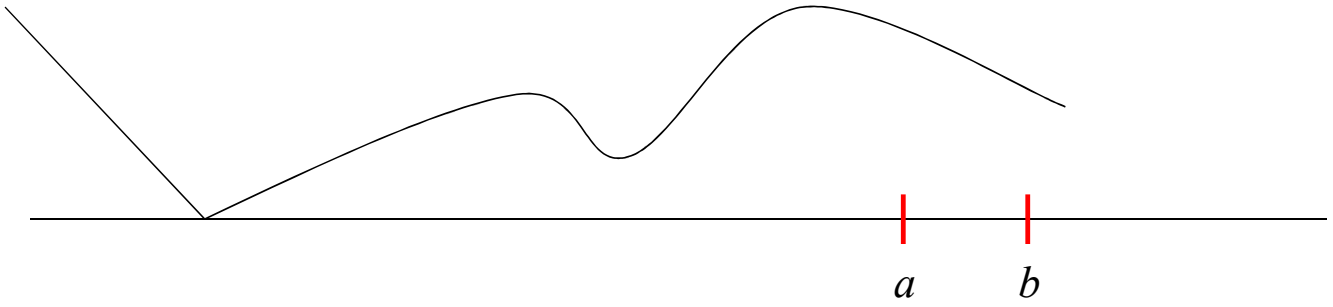
Algorithme de bissection (importance d'hypothèse)

- Hypothèse $\varphi'(a)\ \varphi'(b)<0$ est essentielle
- $\forall \alpha :\ \varphi'(\alpha) \geq 0$



Algorithme de bissection (importance d'hypothèse)

- Hypothèse $\varphi'(a)\ \varphi'(b)<0$ est essentielle
- $\forall \alpha :\ \varphi'(\alpha) \geq 0$
- Uniquement l'intervalle initial contient la racine.



Algorithme de bissection (complexité)

Nombre d'itération:

$L = |b - a|$: Longueur d'intervalle initial

- $L = |b - a| \rightarrow c = \frac{b+a}{2} \rightarrow L = |c - a| = \left|\frac{b+a}{2} - a\right| = \frac{|b-a|}{2} = \frac{L}{2} \rightarrow \cdots \rightarrow \frac{L}{2^n}$

$\left\{L, \frac{L}{2}, \frac{L}{4}, \frac{L}{8}, \ldots, \frac{L}{2^n}, \ldots\right\}$ est une suite générées par l'algorithme de bissection

Le nombre n d'itérations requises pour atteindre une longueur inférieure ou égale à δ :

$$\frac{L}{2^n} \leq \delta \Leftrightarrow 2^n \geq \frac{L}{\delta} \Leftrightarrow n \geq \log_2\left(\frac{L}{\delta}\right) \Rightarrow n = f(L, \delta) = \left[\log_2\left(\frac{L}{\delta}\right)\right]$$

Ordre de convergence:

La suite des longueurs des intervalles $\frac{L}{2^n}$ converge vers 0: $\lim_{n\to\infty} \frac{L}{2^n} \rightarrow 0$

La convergence est linéaire avec un rapport de convergence de $\beta = \frac{1}{2}$:

$$\lim_{k\to+\infty} \frac{\frac{L}{2^{k+1}} - 0}{\frac{L}{2^k} - 0} = \lim_{k\to\infty} \frac{L}{2^{k+1}} \frac{2^k}{L} = \frac{1}{2}$$

Algorithme de Newton-Raphson

1. **Principe**
2. **Algorithme**
3. **Convergence**

Algorithme de Newton : Principe

Hypothèses: Aux points d'évaluation α_k, il est possible d'évaluer $\varphi(\alpha_k), \varphi'(\alpha_k), \varphi''(\alpha_k)$ et de plus $\varphi''(\alpha_k) \neq 0$, pour assure que α^* est un maximum ou un minimum de φ.

Rappel: Développement de Taylor d'ordre n

- Il existe un point z entre x et x^k tel que:

$$\varphi(\alpha) = \varphi(\alpha_k) + \varphi'(\alpha_k)(\alpha - \alpha_k) + \frac{\varphi''(\alpha_k)}{2!}(\alpha - \alpha_k)^2 + \cdots + \frac{\varphi^{(n-1)}(\alpha_k)}{(n-1)!}(\alpha - \alpha_k)^{(n-1)} + \frac{\varphi^{(n)}(z)}{n!}(\alpha - \alpha_k)^n$$

Considérons l'approximation quadratique suivant de φ à x^k:

- $\varphi(\alpha) = \varphi(\alpha_k) + \varphi'(\alpha_k)(\alpha - \alpha_k) + \frac{\varphi''(z)}{2!}(\alpha - \alpha_k)^2$
- $q_k(\alpha) = \varphi(\alpha_k) + \varphi'(\alpha_k)(\alpha - \alpha_k) + \frac{\varphi''(\alpha_k)}{2!}(\alpha - \alpha_k)^2 \Rightarrow \boldsymbol{q_k(\alpha_k) = \varphi(\alpha_k)}$
- $\Rightarrow q_k'(\alpha) = \varphi'(\alpha_k) + 2\frac{\varphi''(\alpha_k)}{2!}(\alpha - \alpha_k) \Rightarrow \boldsymbol{q_k'(\alpha_k) = \varphi'(\alpha_k)}$
- $\Longrightarrow \boldsymbol{q_k''(\alpha) = \varphi''(\alpha_k)}$

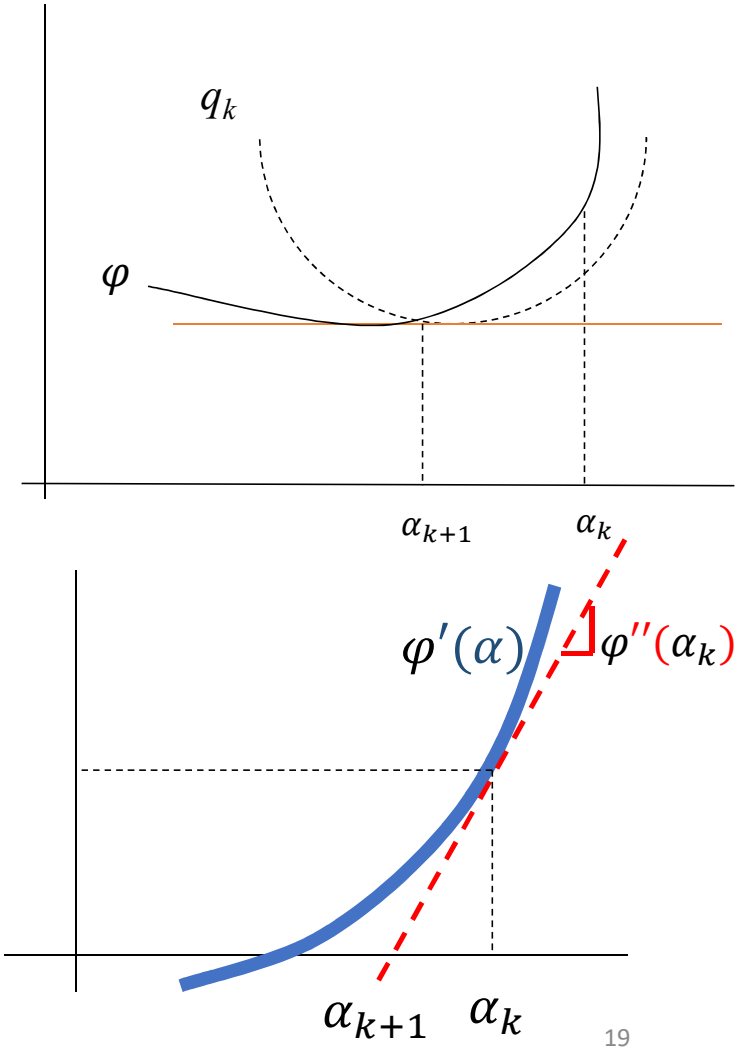
Algorithme de Newton : Principe

la méthode itérative de newton détermine le prochain point d'évaluation α_{k+1}en remplaçant φ par q_k et en annulant $q_k'(\alpha_{k+1})$

- $q_k'(\alpha_{k+1}) = \varphi'(\alpha_k) + \varphi''(\alpha_k)(\alpha_{k+1} - \alpha_k) = 0$
- $\Longrightarrow \alpha_{k+1} - \alpha_k = -\dfrac{\varphi'(\alpha_k)}{\varphi''(\alpha_k)}$
- $\Longrightarrow \boldsymbol{\alpha_{k+1} = \alpha_k - \dfrac{\varphi'(\alpha_k)}{\varphi''(\alpha_k)}}$

Cette méthode utilisée également pour déterminer un point où la fonction s'annule. Il suffit de considérer la fonction $\varphi'(\alpha)$.

α_{k+1} est choisi de telle sorte que de la droite passant par les points $(\alpha_k, \varphi'(\alpha_k))$ et $(\alpha_{k+1}, 0)$ a pour pente $\varphi''(\alpha_k)$, celle de φ' au point α_k.



Algorithme de Newton (algorithme)

- Entrée :
  - α_0: Réel
  - δ : Réel(la tolérance)
  - φ: Fonction (de classe C^2)
- Sortie :
  - α_*: Réel (racine de la fonction)
- Début :

$$k = 0, \alpha_k = \alpha_0$$

DO {

- Calculer les dérivé de fonction $\varphi'(\alpha_k)$ et $\varphi''(\alpha_k) \neq 0$
- Calculer la direction de Newton $d_k = -\frac{\varphi'(\alpha_k)}{\varphi''(\alpha_k)}$
- Calculer l'itération $x^{k+1} = x^k + d_k$
- $k = k + 1$

} **WHILE**($|\alpha_k - \alpha_{k-1}| > \delta$ or $|\varphi'(\alpha_k)| > \delta$)

- $\alpha_* = \alpha_k$
- **RETURN** α_*
- Fin.

Algorithme de Newton: Exemple

Input:

- $\varphi(\alpha) = \alpha^3 + 3\alpha^2 - 1 \Longrightarrow \varphi'(\alpha) = 3\alpha^2 + 6\alpha = 3\alpha(\alpha + 2) \Longrightarrow \varphi''(\alpha) = 6\alpha + 6$
- $\alpha_0 = 2 \Longrightarrow \varphi'(\alpha_0) = 24,\ \varphi''(\alpha_0) = 18$
- $\delta = 0{,}001 \Longrightarrow \alpha_1 = \alpha_0 - \frac{\varphi'(\alpha_0)}{\varphi''(\alpha_0)} = 2 - \frac{24}{18} = \frac{36-24}{18} = \frac{12}{18} = \frac{2}{3}$
- $|\alpha_1 - \alpha_0| = \left|2 - \frac{2}{3}\right| = \frac{4}{3} > \delta$

Boucle:

1. $|\alpha_1 - \alpha_0| = \left|2 - \frac{2}{3}\right| = \frac{4}{3} > \delta$

$$\varphi'(\alpha_1) = \frac{16}{3}, \varphi''(\alpha_1) = 1 \Longrightarrow \alpha_2 = \alpha_1 - \frac{\varphi'(\alpha_1)}{\varphi''(\alpha_1)} = \frac{2}{3} - \frac{16}{30} = \frac{60 - 48}{90} = \frac{2}{15}$$

2. $|\alpha_2 - \alpha_1| = \left|\frac{2}{15} - \frac{2}{3}\right| = \frac{24}{45} = \frac{8}{15} > \delta$

$$\alpha_3 = \alpha_2 - \frac{\varphi'(\alpha_2)}{\varphi''(\alpha_2)}, \varphi'(\alpha_2) = \frac{64}{75}, \varphi''(\alpha_2) = \frac{34}{5} \Longrightarrow \alpha_3 = \frac{2}{15} - \frac{64 \times 5}{75 \times 34} = \frac{2}{255}$$

3. $|\alpha_3 - \alpha_2| = \left|\frac{2}{15} - \frac{2}{255}\right| = \frac{32}{255} = \cdots$

$$c = 0 = \underset{\alpha \in [-1,1]}{arg\min}\, \varphi(\alpha)$$

Algorithme de Newton (Convergence)

Concept de Newton:

- $\alpha_0 \in V(\alpha_*) \rightarrow \alpha_1 \in V(\alpha_0) \rightarrow \alpha_2 \in V(\alpha_1) \rightarrow \cdots \rightarrow \alpha_k \in V(\alpha_{k-1}) \rightarrow \cdots \rightarrow \alpha_* \in V(\alpha_{*-1})$

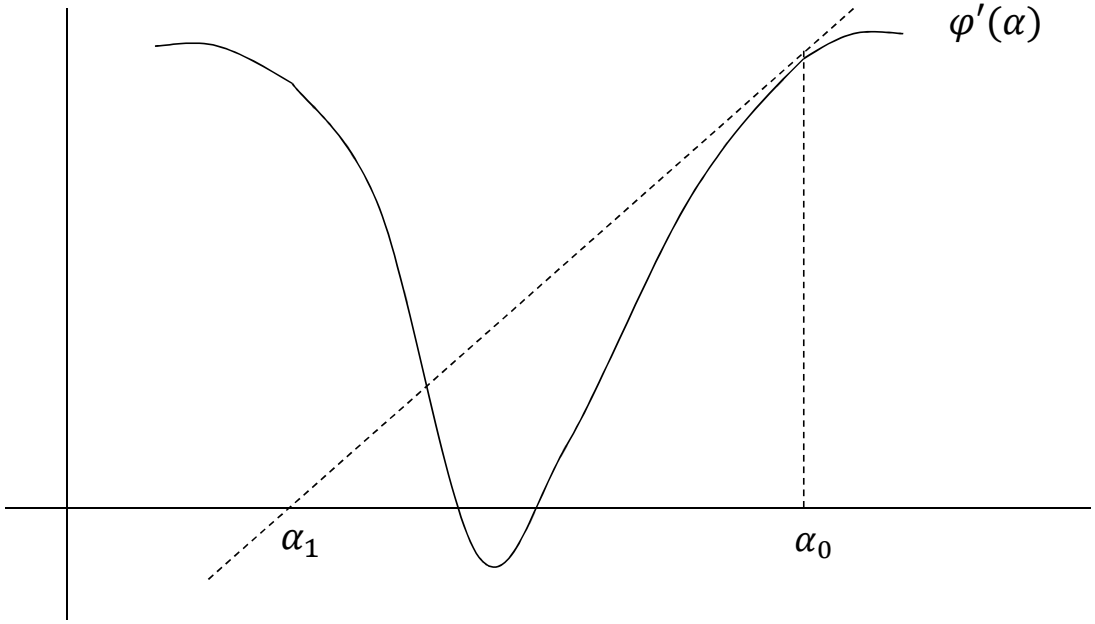
Théorème: soit φ une fonction possédant des dérivées continues d'ordre 3. Supposons que α_* satisfait les conditions $\varphi'(\alpha_*) = 0$ et $\varphi''(\alpha_*) \neq 0$.
Si α_0 (le point initial) est choisi suffisamment près de α_*alors la suite des points $\{\alpha_k\}$ générés par la méthode de Newton converge vers α_* avec un ordre de convergence d'au moins 2.

Avantage / Inconvénient

- Elle permet d'obtenir la valeur exact du minimum d'une fonction.
- l'intérêt de cette méthode est sa convergence quadratique (d'ordre 2)
- il n'est pas facile à l'utiliser puisque elle exige que la fonction soit de 3ème classe, aussi le point de départ doit être suffisamment proche de minimal (point de convergence) sinon la méthode diverge.

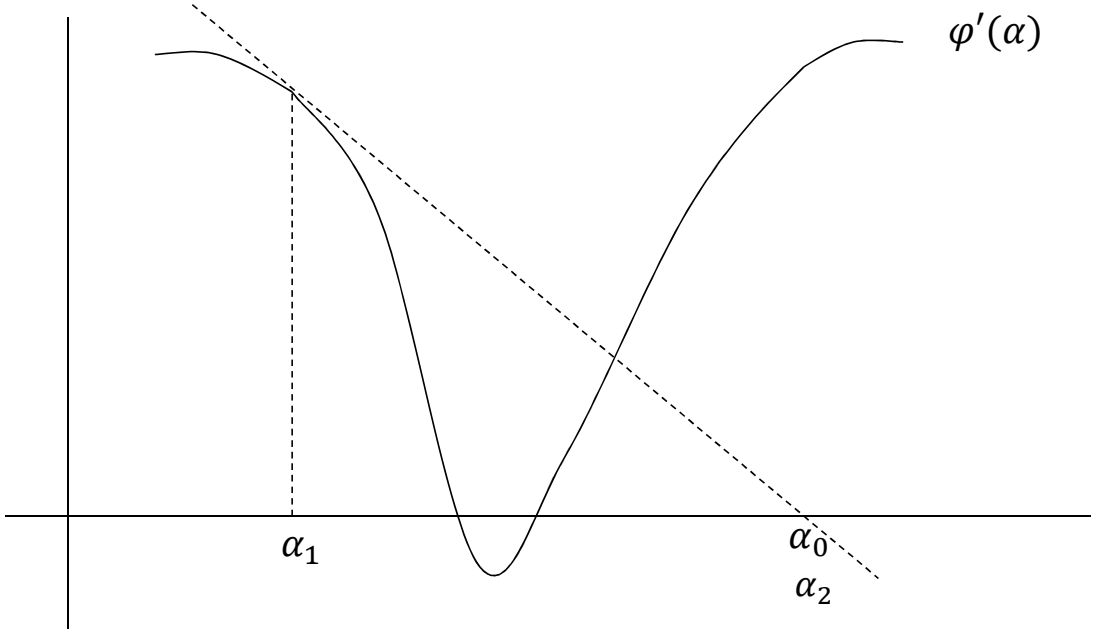
Algorithme de Newton

- Importance de l'hypothèse que α_0 soit suffisamment près de α_*



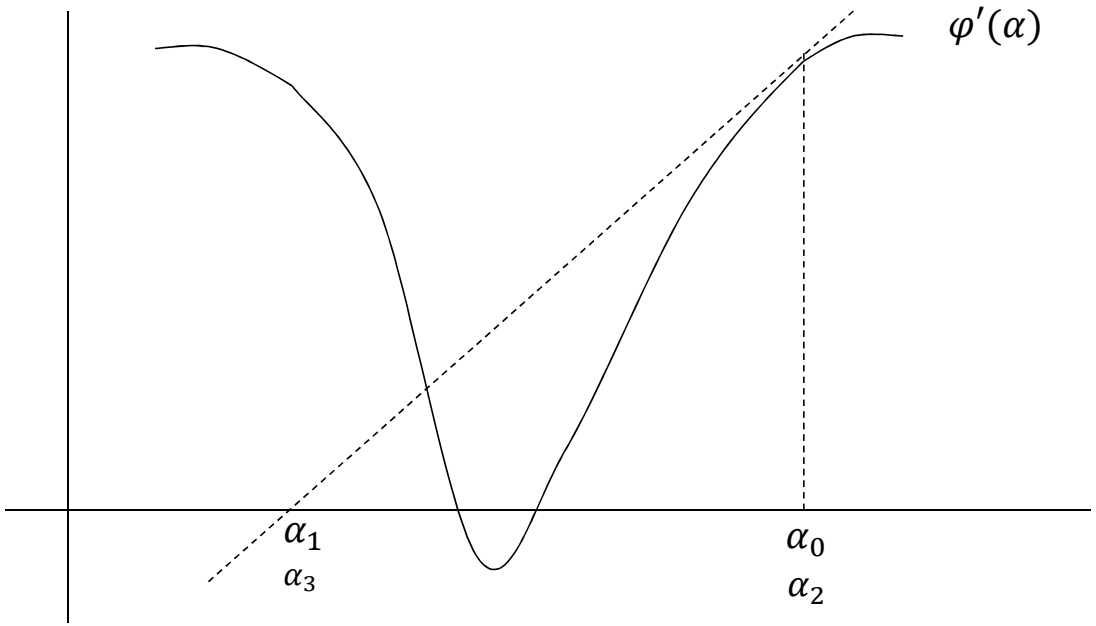
Algorithme de Newton

- Importance de l'hypothèse que α_0 soit suffisamment près de α_*



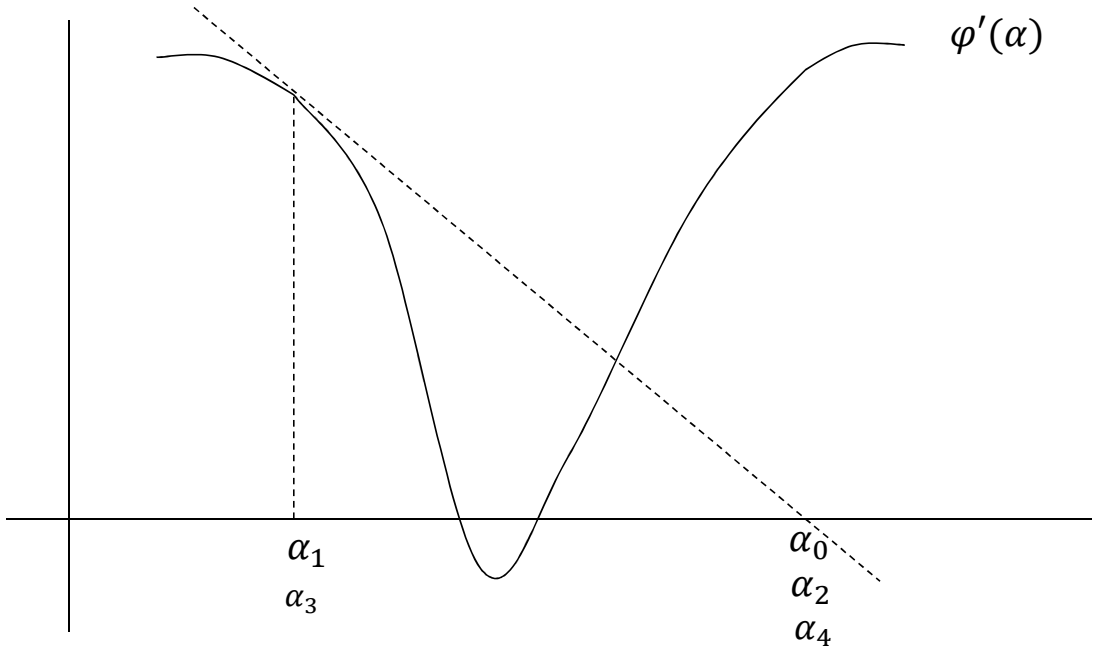
Algorithme de Newton

- Importance de l'hypothèse que α_0 soit suffisamment près de α_*



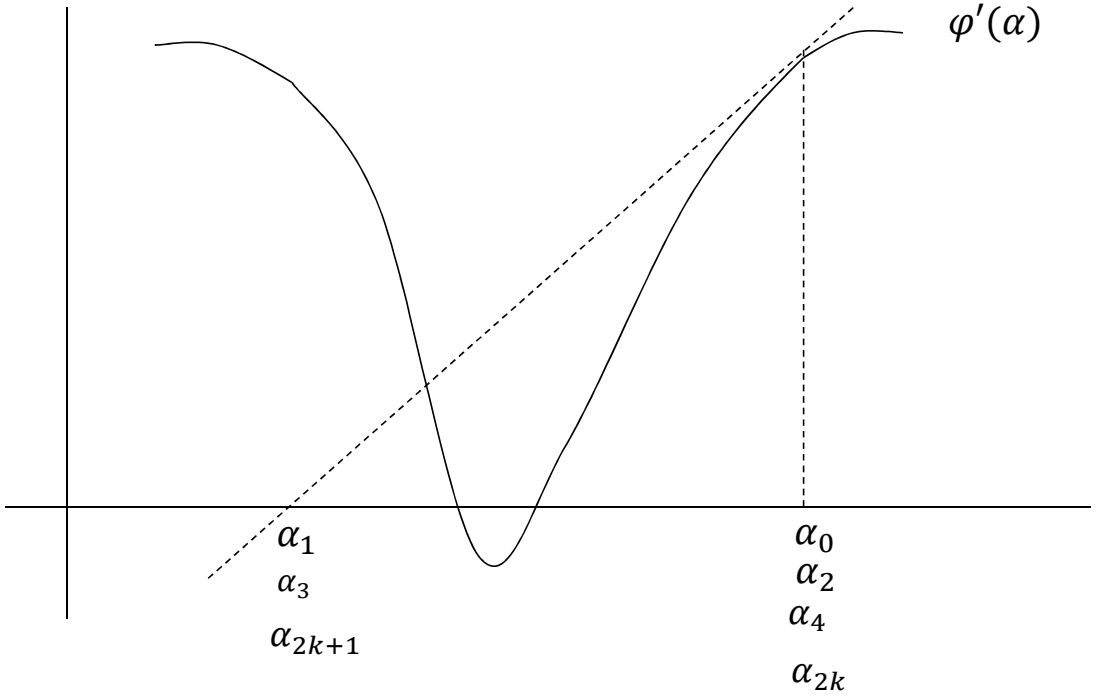
Algorithme de Newton

- Importance de l'hypothèse que α_0 soit suffisamment près de α_*



Algorithme de Newton

- Importance de l'hypothèse que α_0 soit suffisamment près de α_*



Algorithme de Fausse position

1. Principe
2. Algorithme
3. Convergence

Algorithme de la fausse position (Principe)

Iteration k:

$$\varphi''(\alpha_k) = 0$$

- $\alpha_{k-1}, \alpha_k \in V(\alpha_{k-1})$
- $\varphi''(\alpha_k) = \lim_{x \to x^k} \frac{\varphi'(\alpha) - \varphi'(\alpha_k)}{\alpha - \alpha_k} \approx \frac{\varphi'(\alpha_k) - \varphi'(\alpha_{k-1})}{\alpha_k - \alpha_{k-1}}$: fausse position

- $q_k(\alpha) = \varphi(\alpha_k) + \varphi'(\alpha_k)(\alpha - \alpha_k) + \frac{\varphi''(\alpha_k)}{2!}(\alpha - \alpha_k)^2$
- $\bar{q}_k(\alpha) = \varphi(\alpha_k) + \varphi'(\alpha_k)(\alpha - \alpha_k) + \frac{1}{2!}\frac{\varphi'(\alpha_k) - \varphi'(\alpha_{k-1})}{\alpha_k - \alpha_{k-1}}(\alpha - \alpha_k)^2$
- $q_k(\alpha) \approx \bar{q}_k(\alpha)$

Algorithme de la fausse position (Principe)

Hypothèses:
Aux points d'évaluation α_k, il est possible d'évaluer $\varphi(\alpha_k)$, $\varphi'(\alpha_k)$.
Étant donné un point d'évaluation α_k, considérons l'approximation quadratique suivante de φ à α_k ne nécessitant pas la connaissance de $\varphi''(x)$:

- $q_k(\alpha) = \varphi(\alpha_k) + \varphi'(\alpha_k)(\alpha - \alpha_k) + \frac{\varphi''(\alpha_k)}{2!}(\alpha - \alpha_k)^2$
- $\bar{q}_k(\alpha) = \varphi(\alpha_k) + \varphi'(\alpha_k)(\alpha - \alpha_k) + \frac{1}{2}\frac{\varphi'(\alpha_{k-1}) - \varphi'(\alpha_k)}{\alpha_{k-1} - \alpha_k}(\alpha - \alpha_k)^2$

La fonction $\bar{q}_k(\alpha)$ a les propriétés suivants :

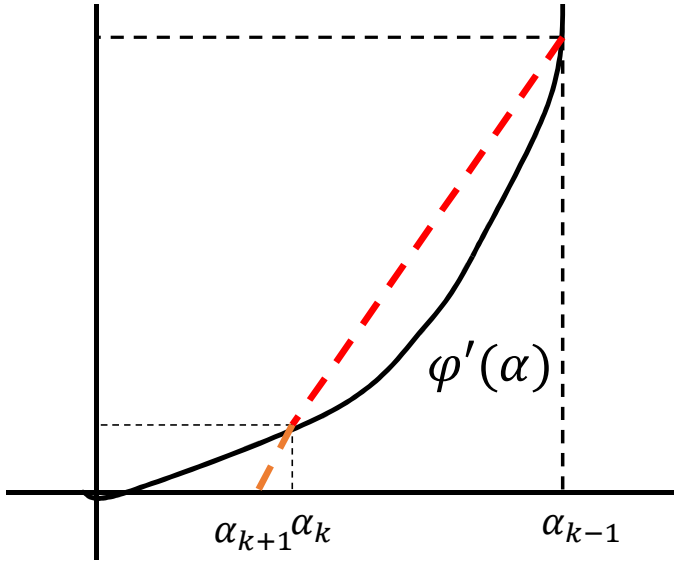
- $\bar{q}_k(\alpha_k) = \varphi(\alpha_k)$
- $\bar{q}_k{}'(\alpha_k) = \varphi'(\alpha_k)$
- $\bar{q}_k{}'(\alpha_{k-1}) = \varphi'(\alpha_{k-1})$

Avec $\bar{q}_k'(\alpha) = \varphi'(\alpha_k) + \frac{\varphi'(\alpha_{k-1}) - \varphi'(\alpha_k)}{\alpha_{k-1} - \alpha_k}(\alpha - \alpha_k)$

Algorithme de la fausse position (Principe)

La méthode itérative de la fausse position détermine le prochain point d'évaluation α_{k+1} en remplaçant φ par $\bar{q}_k$ et en annulant $\bar{q}'_k(\alpha)$:

- $\bar{q}'_k(\alpha_{k+1}) = \varphi'(\alpha_k) + \frac{\varphi'(\alpha_{k-1})-\varphi'(\alpha_k)}{\alpha_{k-1}-\alpha_k}(\alpha_{k+1} - \alpha_k) = 0$
- $\Leftrightarrow \alpha_{k+1} - \alpha_k = -\frac{\varphi'(\alpha_k)(\alpha_{k-1}-\alpha_k)}{\varphi'(\alpha_{k-1})-\varphi'(\alpha_k)}$
- $\Leftrightarrow \alpha_{k+1} = \alpha_k - \frac{\varphi'(\alpha_k)(\alpha_{k-1}-\alpha_k)}{\varphi'(\alpha_{k-1})-\varphi'(\alpha_k)}$



Cette méthode utilisée pour déterminer un point où $\varphi'(\alpha) = 0$.
α_{k+1} est choisi à l'intersection de la droite passant par les points $\left(\alpha_k, \varphi'(\alpha_k)\right)$ et $\left(\alpha_{k-1}, \varphi'(\alpha_{k-1})\right)$ et de l'axe des α.

Algorithme de la fausse position: Algorithme

Entrée : x^0, x^1 : Réels (les extrémités de l'intervalle)

δ : Réel (la tolérance)

φ : Fonctions

Sortie : $\boldsymbol{\alpha_*}$: Réel

Début :

$k = 1$

WHILE($|\alpha_k - \alpha_{k-1}| \geq \delta$ or $|\varphi'(\alpha_k)| \geq \delta$) {

- Calculer les dérivé de fonction $f'(x^k)$ et $f'(x^{k-1})$
- Calculer la direction de fausse position $d_k = -\varphi'(\alpha_k)\frac{\alpha_{k-1}-\alpha_k}{\varphi'(\alpha_{k-1})-\varphi'(\alpha_k)}$
- Calculer l'itération $\alpha_{k+1} = \alpha_k + d_k$

$k = k + 1$

} $\boldsymbol{\alpha_* = \alpha_k}$

RETURN $\boldsymbol{\alpha_*}$

Fin.

Algorithme de la fausse position:Exemple

Rechercher le minimum de $\varphi(\alpha) = \alpha^4 + 2\alpha^2 + 1$ en utilisant 4 itérations de

- la méthode de Newton avec $\alpha^0 = 0.4$
- la méthode des fausses positions avec $\alpha^0 = 0.4$ et $\alpha^1 = -0.2$
- $\boldsymbol{\varphi'(\alpha) = 4\alpha^3 + 4\alpha \rightarrow \varphi''(\alpha) = 12\alpha^2 + 4 > 0}$

1. F.P. $\alpha^{k+1} = \alpha^k - \varphi'(\alpha^k)\left[\frac{\alpha^{k-1}-\alpha^k}{\varphi'(\alpha^{k-1})-\varphi'(\alpha^k)}\right]$
   - $\varphi'(\alpha^0) = 1{,}856 \rightarrow \varphi'(\alpha^1) = -0{,}832$
   - $\alpha^2 = \alpha^1 - \varphi'(\alpha^1)\left[\frac{\alpha^0-\alpha^1}{\varphi'(\alpha^0)-\varphi'(\alpha^1)}\right] = -0{,}2 - (-0{,}832)\left[\frac{0{,}4-(-0{,}2)}{1{,}856-(-0{,}832)}\right] = -0{,}014$
2. Newton $\alpha^{k+1} = \alpha^k - \frac{\varphi'(\alpha^k)}{\varphi''(\alpha^k)} \rightarrow \varphi''(\alpha^0) = 5{,}92$

- $\alpha^1 = \alpha^0 - \frac{\varphi'(\alpha^0)}{\varphi''(\alpha^0)} = 0{,}4 - \frac{1{,}856}{5{,}92} = 0{,}086 \rightarrow \varphi'(\alpha^1) = 0{,}348, \varphi''(\alpha^1) = 4{,}089$
- $\alpha^2 = \alpha^1 - \frac{\varphi'(\alpha^1)}{\varphi''(\alpha^1)} = 0{,}086 - \frac{0{,}348}{4{,}089} = 0{,}0012 \rightarrow \varphi'(\alpha^2) = 0{,}0048$

Algorithme de la fausse position: Convergence

Théorème: Convergence de la méthode de la fausse position

Soit f une fonction possédant des **dérivées continues d'ordre 3**. Supposons que x^* satisfait les conditions $\varphi'(x^*) = 0$ et $\varphi''(x^*) \neq 0$, si x^0 et x^1 (les points initiaux) sont choisis suffisamment près de x^*.

Alors la suite des points $\{x^k\}$ générés par la méthode de la fausse position converge vers x^* avec un ordre de convergence égal à $\tau = 1.618$ (le nombre d'or)

Les méthodes n'utilisant que les valeurs des fonctions

- Algorithme de Fibonacci
- Algorithme de section dorée

Algorithme de Fibonacci: (pseudocode)

Entrée : $\boldsymbol{a}, \boldsymbol{b}$ **:** Réels intervalle initial de largeur $d_1 = |a - b|$

N : Entier (nombre d'itération)

f : Fonction **unimodal**

Sortie : $\boldsymbol{a}, \boldsymbol{b}$: Réels (les extrémités de l'intervalle de sortie)

Début:

Construire F_N la suite de Fibonacci jusqu'à N.

Soit l'intervalle initial $[a_1, b_1]$ donnée et le nombre d'évaluation de fonction est $N - 1$ $(N >= 3)$

FOR $(i = 1, \ldots, N-2)$ {

$$x_1^i = \frac{F_{N-i-1}}{F_{N-i+1}} d_1 + a_i$$

$$x_2^i = \frac{F_{N-i}}{F_{N-i+1}} d_1 + a_i$$

IF $f(x_2^i) > f(x_1^i)$

$$a_{i+1} = a_i$$

$$b_{i+1} = x_2^i$$

IF $f(x_2^i) \leq f(x_1^i)$

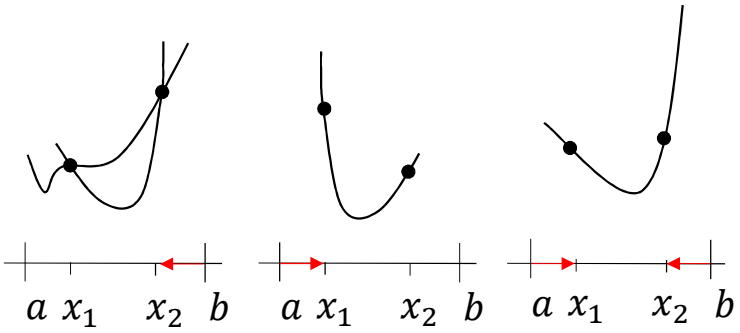
$$a_{i+1} = x_1^i$$

$$b_{i+1} = b_i$$

} $a = x_1^{N-1}, b = x_2^{N-1}$

Return a, b

Fin.



Algorithme de Fibonacci:

Hypothèse : La fonction f est définie sur l'intervalle $[a, b]$ et est uni modal (i.e. f ne possède qu'un seul minimum local dans $[a, b]$.

Notation:

- $d_1 = |b - a|$, la longueur de l'intervalle initial
- $d_k =$ longueur de l'intervalle après avoir utilisé k points d'évaluation
- $\{F_k\}$ la suite des nombres de Fibonacci définie comme suit:
  - $F_0 = F_1 = 1, \quad F_n = F_{n-1} + F_{n-2} \qquad n = 2, 3, \ldots, \implies \{1, 1, 2, 3, 5, 8, 13, 21, \ldots\}$

Stratégie optimale de sélection des points d'évaluation

Supposons que nous décidons au départ d'utiliser N points d'évaluation.

La Procédure se résume comme suit:

- Les deux premiers points sont choisis symétriques à une distance $\left(\frac{F_{N-1}}{F_N}\right) d_1$ de chacune des extrémités de l'intervalle [a, b]. Une partie de l'intervalle est éliminée en se basant sur l'unimodalité de la fonction. Il en résulte un intervalle de longueur $d_2 = \left(\frac{F_{N-1}}{F_N}\right) d_1$.
- Le troisième point est choisi symétriquement par rapport au point déjà dans l'intervalle résultant. Ceci engendre un intervalle de longueur $d_3 = \left(\frac{F_{N-2}}{F_N}\right) d_1$
- En général le point suivant est choisi symétriquement par rapport au point déjà dans l'intervalle résultant.

Algorithme de Fibonacci:Stratégie optimale de sélection des points d'évaluation

Note: Selon (3) , le dernier point N devrait être placé au centre de l'intervalle superposé à celui s'y trouvant déjà. En effet, puisqu'en utilisant cette stratégie de sélection des points d'évaluation, nous avons que:

$$d_k = \frac{F_{N-k+1}}{F_N} d_1 , \qquad k = 2,3, \dots, N$$

- Il s'ensuit que :

$$\left.\begin{aligned} d_{N-1} &= \frac{F_2}{F_N} d_1 = \frac{2}{F_N} d_1 \\ d_N &= \frac{F_{N-N+1}}{F_N} d_1 = \frac{1}{F_N} d_1 \end{aligned}\right\} \Rightarrow d_N = \frac{1}{2} d_{N-1}$$

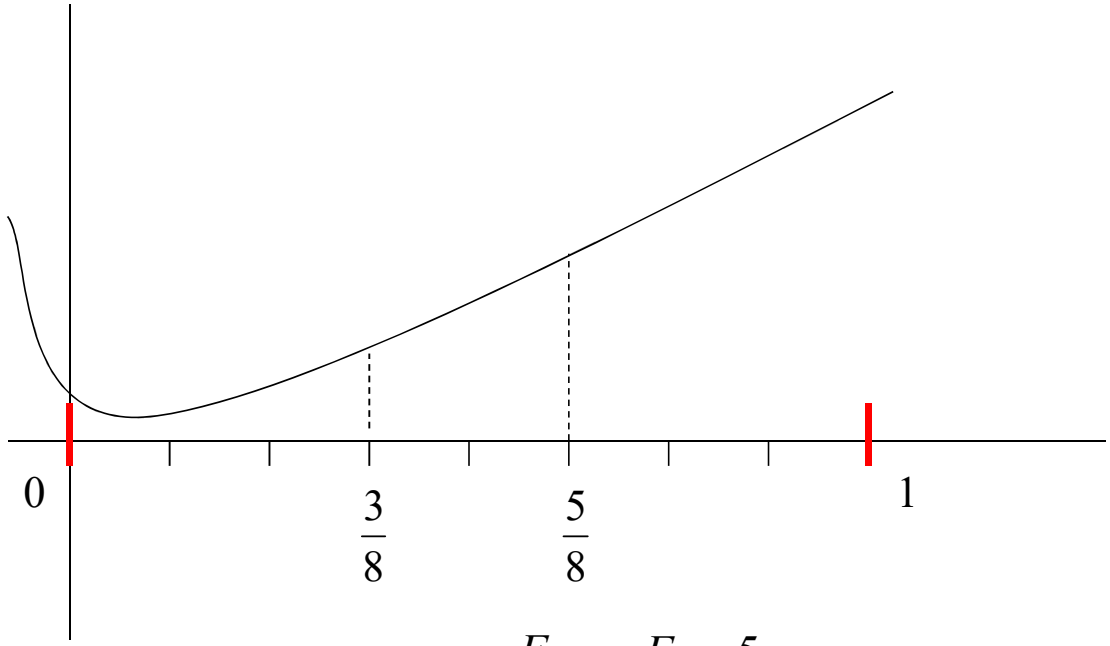
- Pour remédier à cette situation, le dernier point N est plutôt placé à une distance ε (à gauche ou à droite) de celui s'y trouvant déjà.
- En utilisant cette stratégie de sélection des points d'évaluation,

$$d_k = \frac{F_{N-k+1}}{F_N} d_1 , \qquad k = 2,3, \dots, N$$

et il est possible de démontrer que $d_N = \frac{F_{N-N+1}}{F_N} d_1 = \frac{F_1}{F_N} d_1 = \frac{d_1}{F_N}$ est le plus petit intervalle qu'il est possible d'obtenir en utilisant N points d'évaluations. Ainsi, lorsque le nombre de points d'évaluation N devient très grand

- pour tendre vers l'infini, la suite des valeurs $\{d_k\} \to 0$ plus rapidement qu'en utilisant toute autre stratégie.

Algorithme de Fibonacci (Exemple)



$N = 5$

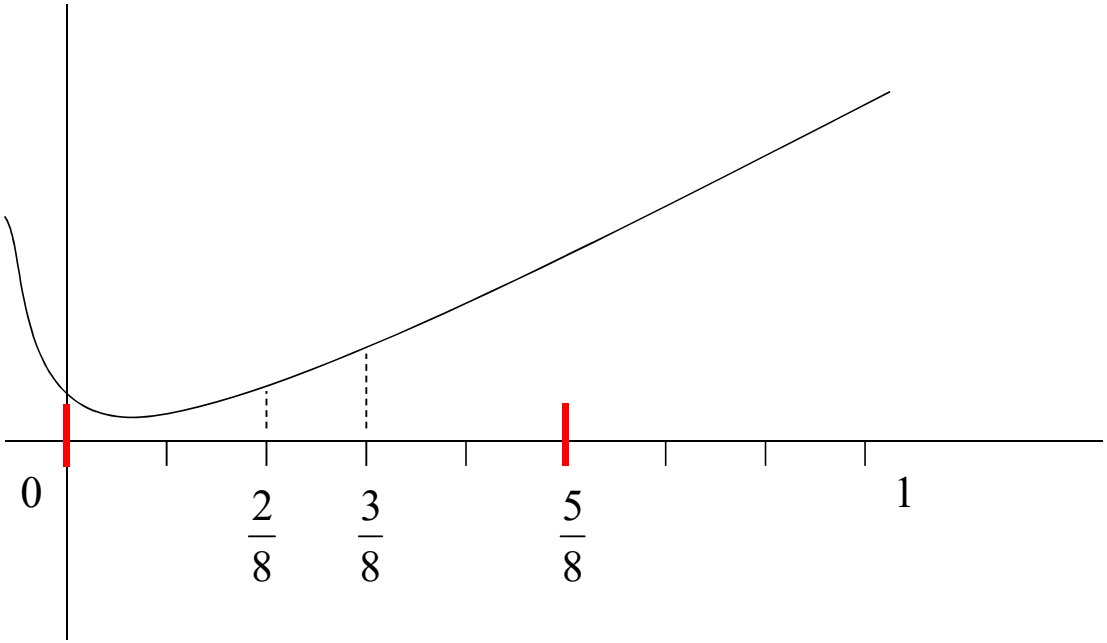
$\{1, 1, 2, 3, 5, 8\}$

$$\frac{F_{N-1}}{F_N} = \frac{F_4}{F_5} = \frac{5}{8}$$

$$d_2 = \frac{F_{5-1}}{F_5} d_1 = \frac{5}{8}$$

$$d_k = \frac{F_{N-k+1}}{F_N} d_1 \qquad k = 2, 3, \cdots, N$$

Algorithme de Fibonacci (Exemple)



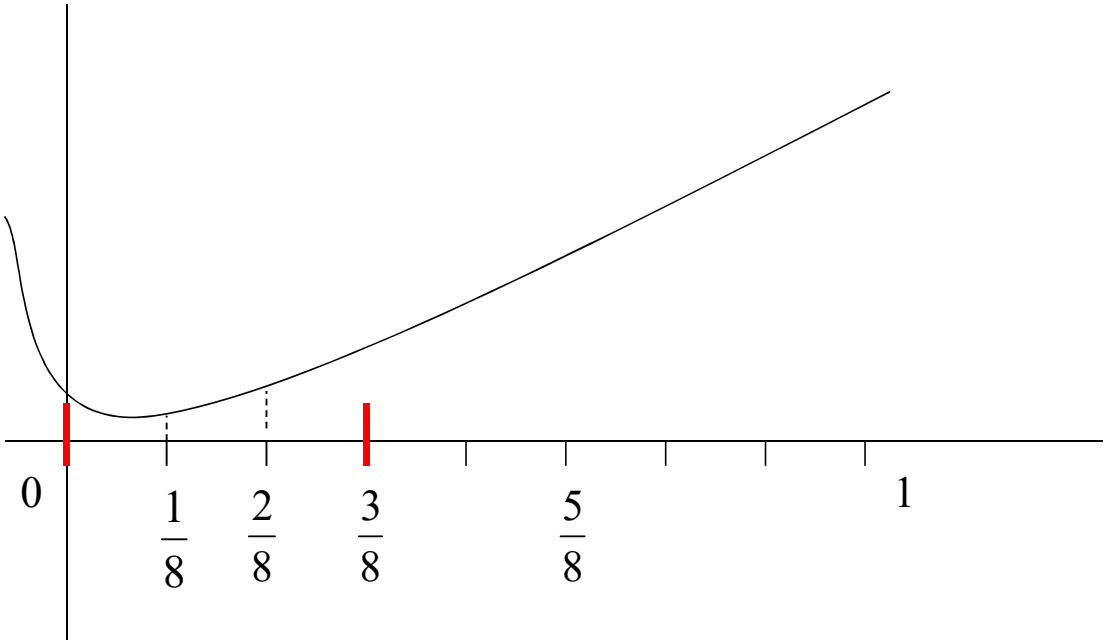
$N = 5$

$\{1, 1, 2, 3, 5, 8\}$

$$d_3 = \frac{F_{5-2}}{F_5} d_1 = \frac{3}{8}$$

$$d_k = \frac{F_{N-k+1}}{F_N} d_1 \qquad k = 2, 3, \cdots, N$$

Algorithme de Fibonacci (Exemple)



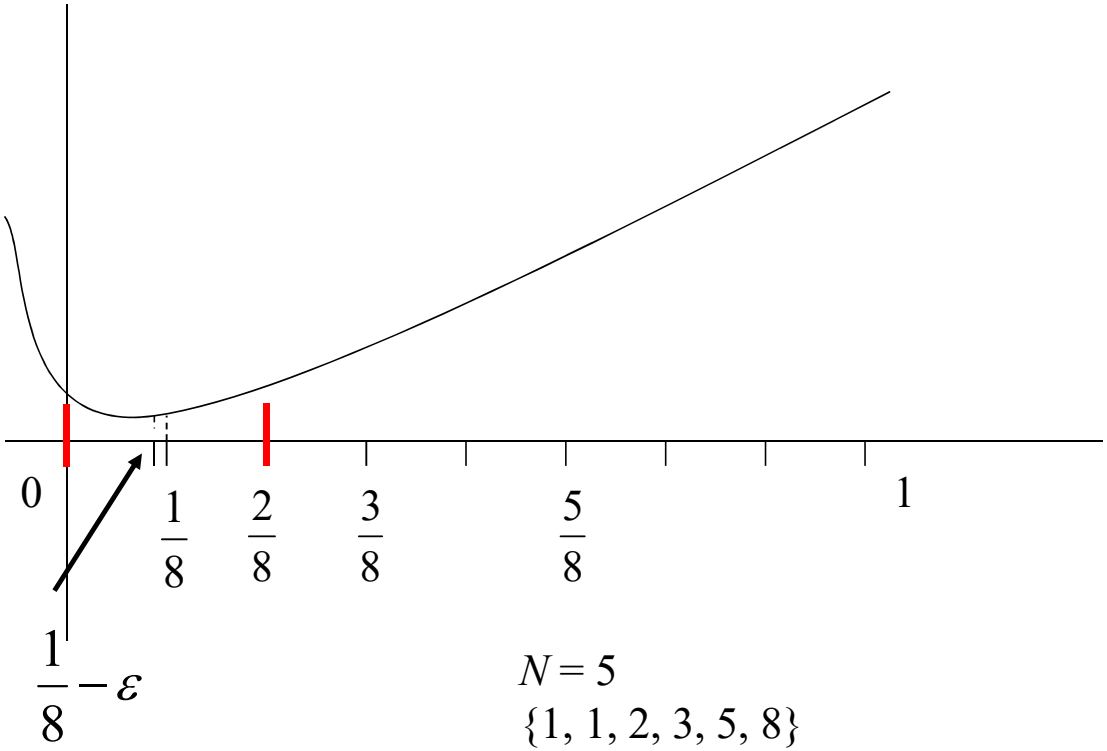
$N = 5$

$\{1, 1, 2, 3, 5, 8\}$

$$d_4 = \frac{F_{5-3}}{F_5} d_1 = \frac{2}{8}$$

$$d_k = \frac{F_{N-k+1}}{F_N} d_1 \qquad k = 2, 3, \cdots, N$$

Algorithme de Fibonacci (Exemple)



$N = 5$

$\{1, 1, 2, 3, 5, 8\}$

$$d_5 = \frac{F_{5-4}}{F_5} d_1 = \frac{1}{8}$$

$$d_k = \frac{F_{N-k+1}}{F_N} d_1 \qquad k = 2, 3, \cdots, N$$

Algorithme de Fibonacci (Exemple)

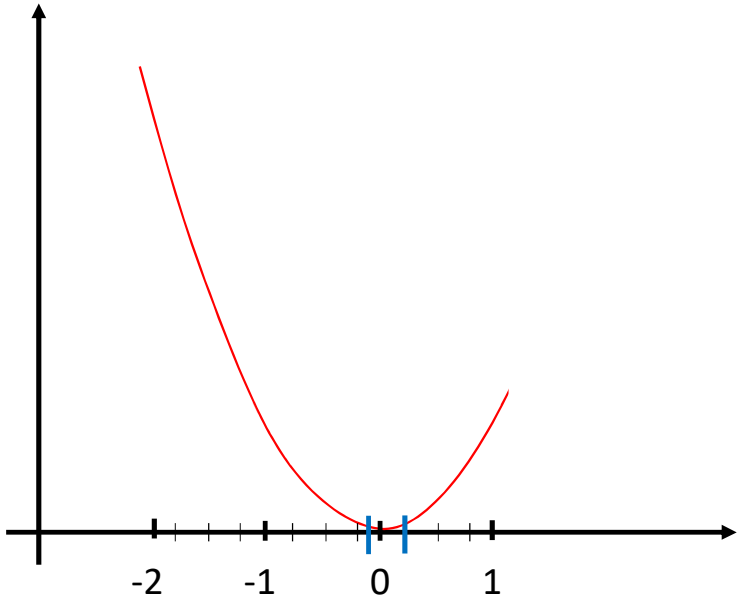
Appliquer 5 itérations d'algorithme de Fibonacci sur la fonction $f(x) = x^2$ avec $x \in]-2,1[$ ($a = -2$ et $b = 1$)

$N = 5$

$\{1, 1, 2, 3, 5, 8\}$

$[a, b] = \left[-\frac{1}{8}, \frac{1}{4}\right]$

$d_k = \frac{F_{N-k+1}}{F_N} d_1 k = 2,3,\cdots,N$



Algorithme de section dorée: (pseudocode)

Entrée :

	$\boldsymbol{a}, \boldsymbol{b}$	**:** Réels	(les extrémités de l'intervalle)
	$\boldsymbol{\delta}$	**: Réel**	**(la tolérance)**
	f	: Fonction **unimodal**	
Sortie :	$\boldsymbol{a}, \boldsymbol{b}$	: Entier	(les extrémités de l'intervalle de sortie)

Constante:

$\boldsymbol{\tau = \frac{1}{2}(\sqrt{5} - 1)}$	:Réel	(nombre d'or)

Début :

$k = 1$

WHILE($|\boldsymbol{b} - \boldsymbol{a}| > \boldsymbol{\delta}$) {

$$x_1^i = \left(\frac{1}{\tau}\right)^{k+1} d_1 + a_i$$

$$x_2^i = \left(\frac{1}{\tau}\right)^{k} d_1 + a_i$$

IF $f(x_2^i) > f(x_1^i)$

$$a_{i+1} = a_i$$
$$b_{i+1} = x_2^i$$

IF $f\left(x_2^i\right) \leq f(x_1^i)$

$$a_{i+1} = x_1^i$$
$$b_{i+1} = b_i$$

$k = k + 1$

}

$$a = x_1^k \quad b = x_2^k$$

RETURN a, b

Fin.

Algorithme de section dorée:

Son nom est dérivé de nombre d'or $\tau = 1.618$

- La méthode de la section dorée utilise la même stratégie que la méthode de Fibonacci pour sélectionner les points d'évaluation, mais le nombre de points d'évaluation n'est pas spécifié au départ.

- Pour spécifier les deux premiers points, **nous procédons comme dans la méthode de Fibonacci en les prenants symétriques** à une distance $\frac{1}{\tau} d_1$ de chaque extrémité en considérant que $N \rightarrow \infty$.