Programmation Mathématiques

A. Bazeniar

Enseignant chercheur
Centre universitaire Abdelhafid Boussouf
Mila, Algérie

Se reporter à des manuels de base et à certaines livres de spécialités

Janvier 2024

Chapitre 1

Introduction et Motivation

1.1 introduction

Prérequis : Algèbre linéaire, le calcul matriciel et la géométrie affine euclidienne.

Note : Dans ce cours, on se focalise sur les techniques d'optimisations (critère d'optimalité, démarches et Algorithmes) plutôt que la modélisation qui est aussi une étape importante dans la chaîne des problèmes d'optimisation.

L'optimisation vise à résoudre des problèmes dont on cherche à trouver une solution satisfaisant avec ou sans contraintes linéaires et non linéaires (égalités et inégalités). Elle cherche à minimiser ou à maximiser une fonction d'une caractéristique numérique. Souvent la théorie et les méthodes de résolution de ces problèmes d'extremum partent d'un principe de solutions réalisables S, ensuite à base d'une stratégie bien tracée on détermine une valeur exacte ou approchée de la solution optimale.

La programmation mathématique est un ensemble de méthodes ou processus mathématiques qui visent la résolution de problèmes d'optimisation. L'objet principal est de determiner les valeurs de decisions qui rendent la valeur de la fonction objectif optimale. Elle est utilisée dans divers domaines tels que,

- Transport et livraisons.
- Fabrication et production.
- Agriculture et génie civil.
- Finance, vente et marketing.
- Gestion de stock.
- Recherche et gestion des bases de données.

N.B La *Programmation* signifier la prise de décision et non pas *codage informatique*.

1.1.1 Types de Programmation Mathématique

— **Programmation Linéaire (PL)** : La fonction objectif et les contraintes sont toutes linéaires.

— P**rogrammation Quadratique (PQ)** : La fonction objectif est quadratique, mais les contraintes peuvent être linéaires ou non linéaires.
— **Programmation Convexe** : La fonction objectif et les contraintes sont convexes.
— **Programmation Non Linéaire (PNL**) : La fonction objectif et (ouà les contraintes sont non linéaires.
— **Programmation Entière (PE)** : Les variables de décision prennent des valeurs entières.
— **Programmation Stochastique** : les paramètres du problème sont aléatoires.
— **Programmation Dynamique** : pour les problèmes décisionnels séquentiels.

1.1.2 Méthodes de preuve (pour la rigueur mathématique)

Dans ce cours, la plupart des énoncés ont la forme

$$A \Rightarrow B.$$

Pour prouver une telle implication, on peut choisir l'une des trois techniques suivantes :
— **Preuve directe :** on part de A et, par un raisonnement analytique, on arrive à B.
— **Preuve par contraposition :** on suppose $\neg B$ et on montre $\neg A$.
— **Preuve par contradiction :** on suppose A et $\neg B$ et on aboutit à une contradiction.

1.2 Optimisation unidimensionnelle

Le problème d'optimisation le plus simple, sans contrainte, est la recherche des maxima ou minima d'une fonction d'une variable réelle $f(x)$, avec $x \in \mathbb{R}$. On l'écrit :

$$\min_{x\in\mathbb{R}} f(x) \quad \text{ou} \quad \max_{x\in\mathbb{R}} f(x).$$

Remarque 1.2.1. *La notation argmin ou argmax est utilisée dans certains ouvrage. Ainsi, l'optimisation ci-dessus peut être écrite sous la forme suivante :*

$$\arg\min_{x\in\mathbb{R}} f(x) \quad \textit{ou} \quad \arg\max_{x\in\mathbb{R}} f(x).$$

L'objectif est de trouver le (ou les) point(s) x dans le domaine de f qui minimise(nt) ou maximise(nt) la fonction.

Définition 1.2.1. *Une solution $x^* \in S \subseteq \mathbb{R}$ est un minimum global de f (fonction continue) sur S si,*

$$f(x^*) \leq f(x), \forall x \in S. \tag{1.1}$$

Remarque 1.2.2. *Le minimum global x^* n'est pas unique mais la valeur $f(x^*)$ l'est.*

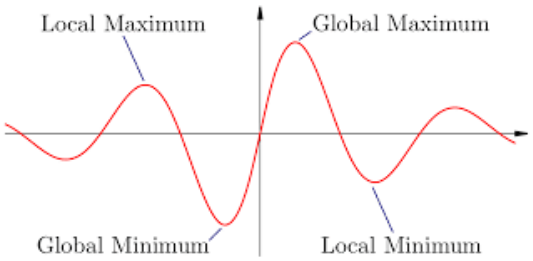


FIGURE 1.1 – Les extremaux d'une fonction

Définition 1.2.2. *Une solution $x^* \in S$ est un minimum local de la fonction f sur $S \subseteq \mathbb{R}$ si,*

$$f(x^*) \leq f(x), \forall x \in B_\epsilon(x^*) \subseteq S. \tag{1.2}$$

Tel que : $B_\epsilon(x^*)$ est appelée une boule de rayon ϵ centrée en x^* avec,

$$B_\epsilon(x^*) = \{x \in \mathbb{R}^n, \parallel x - x^* \parallel < \epsilon\}. \tag{1.3}$$

Définition 1.2.3. *Une fonction $f : [a, b] \to \mathbb{R}$ est dite* unimodale *si elle possède un seul extremum local (minimum ou maximum) sur l'intervalle.*

Théorème 1.2.1. ***(Condition nécessaire du premier ordre)***
Soit $f : \mathbb{R} \to \mathbb{R}$ une fonction dérivable en x^*. Si x^* est un extremum local de f, alors

$$f'(x^*) = 0.$$

Un tel point est appelé point stationnaire *ou* point critique.

Théorème 1.2.2. ***(Condition suffisante du second ordre)***
Soit $f \in \mathcal{C}^2(\mathbb{R})$. Si $f'(x^*) = 0$ et :

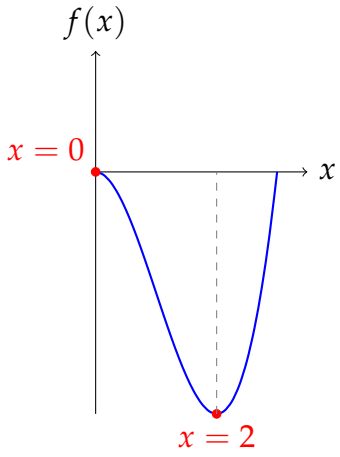
- *si $f''(x^*) > 0$, alors x^* est un minimum local strict ;*
- *si $f''(x^*) < 0$, alors x^* est un maximum local strict ;*
- *si $f''(x^*) = 0$, le test est inconclusif (point d'inflexion possible).*

Remarque 1.2.3. *Si $f'(x^*) = 0$ et $f''(x^*) = 0$, il peut s'agir d'un point d'inflexion ou d'un extremum dégénéré. Une analyse d'ordre supérieur est alors nécessaire.*

Exemple 1.2.1. *Soit $f(x) = x^3 - 3x^2$.*

- $f'(x) = 0 \Rightarrow 3x^2 - 6x = 0 \Rightarrow x = 0 \ \wedge \ x = 2.$
- $f''(x) = 6x - 6 \Rightarrow f''(0) = -6 < 0 \ \wedge \ f''(2) = 6 > 0$

Ainsi, $x = 0$ est un maximum local et $x = 2$ est un minimum local.



Une grande classe des fonctions de ce problème n'est pas si simple. D'une part, C'est le cas des situations réelles, il se peut que la résolution de l'équation $f'(x) = 0$ soit très compliquée. D'autre part, dans les problèmes pratiques, on ne sait pas souvent si la fonction $f(x)$ est dérivable. Bref, si ces cas se présentent on fait recours aux méthodes itératives.

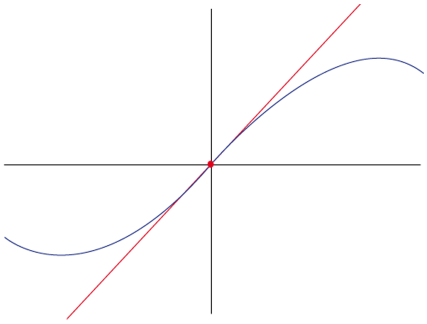


FIGURE 1.2 – Point d 'inflexion

Exemple 1.2.2. *Pour $S = \mathbb{R}$, la fonction $f(x) = e^x$ n'admet pas de minimum global (elle tend vers 0 lorsque $x \to -\infty$ mais ne l'atteint pas). Elle n'a pas non plus d'extremum local, car*

$$f'(x) = e^x > 0 \quad \textit{pour tout } x \in \mathbb{R}.$$

1.3 Méthodes itératives

Certaines méthodes itératives sont appelées techniques de *recherche directe* et sont basées sur des comparaisons de valeurs de fonctions à des points limites. D'autres, appelées *méthodes à dérivée*, utilisent les dérivées de la fonction objective et peuvent être considérées comme des algorithmes de résolution de l'équation non linéaire $f'(x) = 0$. Les méthodes de gradient tendent à converger plus rapidement que les méthodes de recherche directe. Dans cette section on va étudier les algorithmes suivants :

- **Méthodes sans dérivée :** basées sur l'évaluation de f en plusieurs points (ex : bissection adaptée, section dorée).

■ **Méthodes avec dérivée :** utilisent f' et évetuellement f'' (ex : Sécante, Newton).

Les *méthodes sans dérivées* sont basées sur le principe de l'évaluation de la fonction f en différents point sur l'intervalle $[a, b]$. L'objectif est de réduire progressivement l'intervalle jusqu'à ce que le minimum soit encadré avec une précision suffisante.

1.3.1 Méthode de recherche par intervalle (Bissection pour les minima)

Le principe de la méthode consiste à estimer la plus petite valeur de f (qui soit unimodale) dans un intervalle $[a, b]$, cela fait en calculant la fonction en de nombreux points de l'intervalle. Puis de choisir celui qui a la valeur la plus faible.

Algorithme : recherche par intervalle

1. Choisir un intervalle $[a, b]$ contenant un minimum et un réel $\varepsilon > 0$.
2. Tant que $b - a > \varepsilon$:
   — Calculer
   $$m = \frac{a+b}{2}, \qquad x_l = \frac{a+m}{2}, \qquad x_r = \frac{m+b}{2}.$$
   — Évaluer
   $$f(a),\ f(m),\ f(b),\ f(x_l),\ f(x_r).$$
   — Déterminer le point $x_{\min}$ donnant la plus petite valeur de f.
   — Remplacer l'intervalle $[a, b]$ par le sous-intervalle contenant $x_{\min}$.
3. Retourner $\frac{a+b}{2}$ comme approximation du minimum.

Voici une implémentation en Python de l'algorithme de recherche par intervalle

```
import math
def bissection(f, a, b, epsilon=1e-3):
    i=0
    while (b - a) > epsilon:
        m = (a + b) / 2
        xl = (a + m) / 2
        xr = (m + b) / 2
        i+=1
        # Évaluation de la fonction aux points
        f_values = { 'a': f(a),  'xl': f(xl), 'm': f(m),
        'xr': f(xr), 'b': f(b)  }
```

```
        # Détermination du point donnant la plus petite valeur
        xmin = min(f_values, key=f_values.get)

        # Mise à jour de l'intervalle
        if xmin in ['a', 'xl', 'm']:
            b = m
        else:
            a = m
    # Retourner la moyenne de l'intervalle comme approximation du minimum
    return i,(a + b) / 2
# Exemple d'utilisation
def f(x):
    return x**3 - 3*x**2

a, b = 0, 3
nbr_it,xmin_approx = bissection(f, a, b, epsilon=1e-3)
print("Approximation du minimum :", xmin_approx)
print("Valeur minimale :", f(xmin_approx))
print("Le nombre d'itérations :", nbr_it)
```

Remarque 1.3.1. *Cette méthode ne garantit pas la convergence vers un minimum global si f a plusieurs minima locaux.*

Exemple 1.3.1. *Reprenant l'exemple precedent $f(x) = x^3 - 3x^2$ sur l'intervalle $[0, 3]$.*

On pose $a = 0,\ b = 3,\ m = 1.5$.
on calcule $x_l = 0.75,\ x_r = 2.25$ alors, on aura

$$f_a = 0;\ f_l = -1.266;\ f_m = -3.375;\ f_r = -3.797;\ f_b = 0.$$

La valeur minimal f_{min} se situe à $x_r = 2.25$ et donc l'interval devient $[m, b] = [1.5, 3.0]$.
recalculer x_l, x_r on aura

$$a = 1.5;\ x_l = 1.875;\ m = 2.25;\ x_r = 2.625;\ b = 3.$$

et

$$f_a = -3.375;\ f_l = -3.955;\ f_m = -3.797;\ f_r = -2.584;\ f_b = 0.$$

La valeur minimal f_{min} se situe à $x_l = 1.875$ et donc l'interval devient $[a, m] = [1.5, 2.25]$.
re-calcule x_l, x_r on aura

$$a = 1.5;\ x_l = 1.6875;\ m = 1.875;\ x_r = 2.0625;\ b = 2.25.$$

et

$$f_a = -3.375;\ f_l = -3.737;\ f_m = -3.955;\ f_r = -3.988;\ f_b = -3.797.$$

Ces valeurs impliquent que le minimum se situe dans $[x_l, x_r] = [1,875, 2,25]$. *Après quelques étapes supplémentaires, on a une approximation acceptable de la vraie solution (déjà calculer en dessus) à* $x = 2$.

Courbe de convergence : *Enfin, pour visualiser la convergence, on peut tracer l'erreur* $|x_k - x^*|$ *en fonction du nombre d'itérations k :*

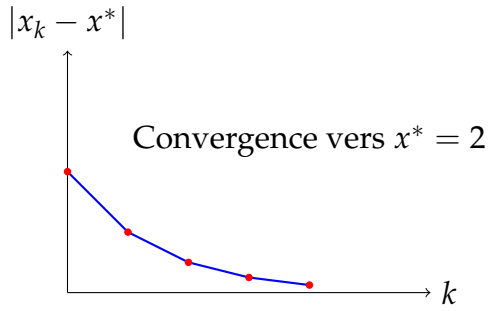


FIGURE 1.3 – visualisation de la convergence.

Dans le cas de la minimisation d'une fonction avec la méthode de la bissection. Une estimation grossière du nombre d'itérations N peut être donnée par la proposition suivante.

Proposition 1.3.1. *Soit* $[a, b]$ *un intervalle.*

$$N \approx \frac{log(b-a)/\epsilon)}{log(2)}, \quad \epsilon \text{ est la tolérance d'erreur.} \tag{1.4}$$

Démonstration. La longueur de l'intervalle contenant la solution est divisée par deux à chaque itération. Ainsi, après N itérations, la longueur de l'intervalle est de

$$\frac{(b-a)}{2^N}.$$

Et comme le critère d'arrêt est estimée à ϵ alors,

$$\frac{(b-a)}{2^N} \leq \epsilon.$$

Après l'introduction du log sur les deux cotés, on aura

$$N \geq \frac{log((b-a)/\epsilon)}{log(2)}$$

□

Par exemple le nombre d'itération pour l'exemple precedent est estimé à

$$N \approx \frac{log(3/0.01)}{log(2)} = 8.228..., \quad \text{On prend} \quad N = 9.$$

1.3.2 La méthode de la Section dorée

Objectif principal : La méthode de la section dorée (ou "golden section search") est un algorithme d'optimisation unidimensionnelle visant à trouver le minimum (ou maximum) d'une fonction unimodale sur un intervalle donné sans utiliser la notion dérivée. Elle a les caractéristiques :

— À chaque itération, l'intervalle de recherche est réduit d'un facteur constant ρ.
— Une seule nouvelle évaluation de fonction est nécessaire par itération.
— Méthode déterministe et convergente pour une fonction unimodale.

Principe de la méthode : Supposons que la fonction f est unimodale sur l'intervalle $[a, b]$. L'idée est d'évaluer f en deux points intérieurs x_1 et x_2 de l'intervalle, comme elle montre la figure ci-dessous.

$$x_1 = a + \rho(b - a), \quad x_2 = b - \rho(b - a),$$

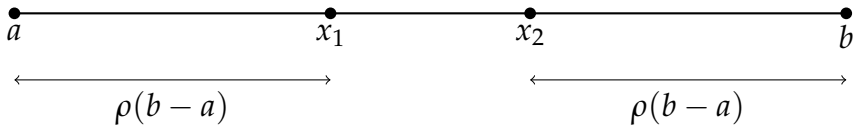


FIGURE 1.4 – Points internes x_1 et x_2 dans la méthode de la section dorée

On évalue ensuite $f(x_1)$ et $f(x_2)$:
— Si $f(x_1) < f(x_2)$, le minimum se trouve dans l'intervalle $[a, x_2]$.
— Sinon, le minimum se trouve dans $[x_1, b]$.

Supposons par exemple que $f(x_1) > f(x_2)$. Alors, le minimum de la fonction se trouve dans l'intervalle $[x_1, b]$.Ensuite, afin de rendre la méthode efficace, on impose que le nouvel intervalle $[x_1, b]$ possède la même structure géométrique que l'intervalle initial $[a, b]$.

Sur l'intervalle initial :
— longueur totale : $b - a$,
— premier point interne : x_2,
— distance correspondante : $x_2 - a$.

Sur le nouvel intervalle $[x_1, b]$:
— longueur totale : $x_2 - a$,
— premier point interne : x_1,
— distance correspondante : $x_1 - a$.

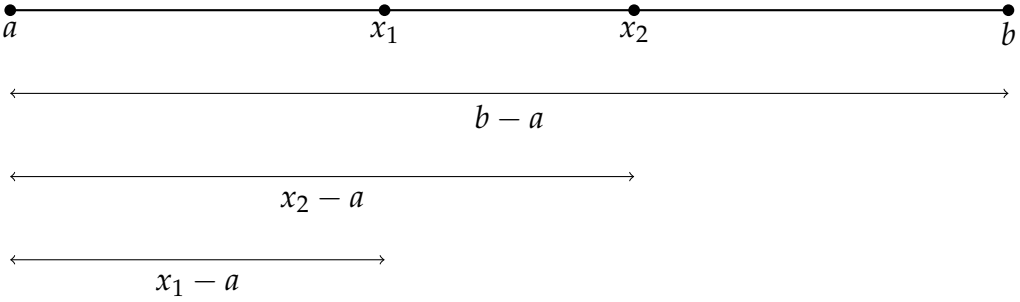


FIGURE 1.5 – Disposition des points internes x_1 et x_2 dans l'intervalle $[a, b]$

Afin que les rapports de longueurs soient identiques, on impose :

$$\frac{\text{ancien intervalle}}{\text{nouvel intervalle}} = \frac{\text{nouvel intervalle}}{\text{sous-intervalle interne}},$$

c'est-à-dire

$$\frac{b-a}{x_2-a} = \frac{x_2-a}{x_1-a}.$$

Or,

$$x_2 - a = (1-\rho)(b-a), \qquad x_1 - a = \rho(b-a).$$

On en déduit alors,

$$\frac{1}{1-\rho} = \frac{1-\rho}{\rho}.$$

La solution est

$$\rho = \frac{3-\sqrt{5}}{2} \approx 0.381966.$$

Algorithme : section dorée

1. **Initialisation :** choisir un intervalle $[a, b]$, un réel $\varepsilon > 0$ et

$$\rho = 1 - \varphi \approx 0.382,$$

où $\varphi = \frac{\sqrt{5}-1}{2}$ est le nombre d'or.

2. **Tant que** $b - a > \varepsilon$:
   — Calculer
   $$x_1 = a + \rho(b-a), \qquad x_2 = a + (1-\rho)(b-a).$$
   — Si $f(x_1) < f(x_2)$, alors
   $$b \leftarrow x_2.$$
   — Sinon,
   $$a \leftarrow x_1.$$

3. **Retourner**
   $$\frac{a+b}{2}$$
   comme approximation du minimum.

```
import math

def f(x):
    return x**3 - 3*x**2

eps = float(input("Précision souhaitée ? "))
a = float(input("Borne a ? "))
b = float(input("Borne b ? "))
rho = (3 - math.sqrt(5)) / 2  # environ 0.382
while abs(b - a) > eps:
    x1 = a + rho * (b - a)
    x2 = a + (1 - rho) * (b - a)
    if f(x1) < f(x2):
        b = x2
    else:
        a = x1
    print(f"Nouvel intervalle : [{a:.6f}, {b:.6f}]")

print(f"Solution approchée : {(a+b)/2:.6f}")
```

Le code génère l'intervalle final $[1.9969315472632534, 2.0062422606047927]$ avec une précision à l'ordre de 0.01.

1.3.3 Encadrement préliminaire

Dans des situations pratiques, on peut se retrouver avec un point initial x^0 sans avoir un intervalle bien définit. On cite au-dessous un algorithme qui peut remédier se problème. On décrit une méthode d'encadrement préliminaire nécessaire avant d'appliquer une méthode d'optimisation unidimensionnelle qui parte d'un intervalle initial comme information.

Algorithme d'encadrement du minimum

1. Initialisation : Choisir le point initial x^0 et le pas $\alpha > 0$.
2. Poser $\delta = -\alpha \times sign(f'(x^0))$.
3. **Repeat**
$$x^{k+1} = x^k + \delta.$$
4. **Until** $f(x^{k+1}) > f(x^k)$
5. Si $k = 0$ alors soit $a = x^0$ et $b = x^1$.
6. Si $k > 0$ alors soit $a = x^{k-1}$ et $b = x^{k+1}$.

cet algorithme Il transforme ainsi un problème d'optimisation à partir d'un seul point x^0 en un problème d'optimisation sur intervalle.

Exemple 1.3.2. *Soient l'exemple suivant.*

— ***Point initial*** : $x^0 = 0$
— ***Pas*** : $\alpha = 1$
— ***Fonction*** : $f(x) = x^2 - 4x + 5$
— ***Dérivée*** : $f'(x) = 2x - 4$
Calcul de δ

$$\begin{aligned} f'(x^0) = f'(0) = 2(0) - 4 = -4 \\ \text{sign}(f'(x^0)) = \text{sign}(-4) = -1 \\ \delta = -\alpha \times \text{sign}(f'(x^0)) = -1 \times (-1) = 1 \end{aligned}$$

Comme $f'(0) < 0$, la fonction décroît à droite de x^0, donc on se déplace vers la droite ($\delta = +1$).
Itération 1 ($k = 0$***)*** : $x^1 = x^0 + \delta = 0 + 1 = 1$

$$\begin{aligned} f(x^0) = f(0) = (0)^2 - 4(0) + 5 = 5 \\ f(x^1) = f(1) = (1)^2 - 4(1) + 5 = 1 - 4 + 5 = 2 \end{aligned}$$

Condition d'arrêt : $f(x^1) = 2 > f(x^0) = 5$ *Faux* $\rightarrow$ *On continue.*

Itération 2 (**$k = 1$**) :

$$x^2 = x^1 + \delta = 1 + 1 = 2$$

$$f(x^2) = f(2) = (2)^2 - 4(2) + 5 = 4 - 8 + 5 = 1$$

***Condition d'arrêt** : $f(x^2) = 1 > f(x^1) = 2$ Faux → On continue.*

Itération 3 (**$k = 2$**) :

$$x^3 = x^2 + \delta = 2 + 1 = 3$$

$$f(x^3) = f(3) = (3)^2 - 4(3) + 5 = 9 - 12 + 5 = 2$$

***Condition d'arrêt** : $f(x^3) = 2 > f(x^2) = 1$ Vrai → On s'arrête.*
***Résultat** : L'intervalle obtenu est $[a, b] = [1, 3]$.*
***Vérification :** on a,*

$$f(1) = 2, \quad f(2) = 1, \quad f(3) = 2$$

La fonction décroît de $x = 1$ à $x = 2$, puis croît de $x = 2$ à $x = 3$, confirmant que l'intervalle $[1, 3]$ contient bien un minimum.

1.3.4 Méthode de Newton

Cette méthode est appelée méthode d'approximation quadratique. On commence par choisir un point initial dans l'espace des variables. Puis pour chaque itération, on calcule la première et la deuxième dérivée de f. Cette méthode approxime localement la fonction autour du point x^k en utilisant une fonction quadratique, souvent définie par la série de Taylor de deuxième ordre.
Développement de Taylor d'ordre 2 : Pour une fonction f deux fois différentiable, on peut écrire au voisinage de x_n :

$$f(x) = f(x_n) + f'(x_n)(x - x_n) + \frac{1}{2}f''(x_n)(x - x_n)^2 + R_2(x)$$

où $R_2(x)$ est le **reste d'ordre 2**, qui devient négligeable lorsque x est proche de x_n.
On définit l'approximation quadratique $q(x)$ par :

$$q(x) = f(x_n) + f'(x_n)(x - x_n) + \frac{1}{2}f''(x_n)(x - x_n)^2$$

La dérivée première de $q(x)$ s'obtient directement :

$$q'(x) = \frac{d}{dx}\left[f(x_n) + f'(x_n)(x - x_n) + \frac{1}{2}f''(x_n)(x - x_n)^2\right]$$

En appliquant les règles de dérivation :

$$q'(x) = f'(x_n) + f''(x_n)(x - x_n)$$

Pour trouver le minimum (ou maximum) de $q(x)$, on cherche x tel que :

$$q'(x) = 0$$

Ce qui donne l'équation :

$$f'(x_n) + f''(x_n)(x - x_n) = 0$$

Cas 1 : Si $f''(x_n) \neq 0$ (cas non-dégénéré)
On peut diviser par $f''(x_n)$:

$$x - x_n = -\frac{f'(x_n)}{f''(x_n)}$$

D'où la solution :

$$\boxed{x = x_n - \frac{f'(x_n)}{f''(x_n)}}$$

Cas 2 : Si $f''(x_n) = 0$ (cas dégénéré)
L'équation devient :

$$f'(x_n) = 0$$

Deux sous-cas se présentent :

1. Si $f'(x_n) = 0$, alors x_n est déjà un point critique
2. Si $f'(x_n) \neq 0$, il n'existe pas de solution (modèle linéaire)

La nature du point obtenu dépend de $f''(x_n)$:

$f''(x_n)$	Nature de $q(x)$	Type de point
> 0	Convexe (minimum)	Minimum local de $q(x)$
< 0	Concave (maximum)	Maximum local de $q(x)$
$= 0$	Droit (dégénéré)	Point d'inflexion ou cas limite

La solution du problème de minimisation de $q(x)$ nous donne la formule d'itération :

$$\boxed{x_{n+1} = x_n - \frac{f'(x_n)}{f''(x_n)}}$$

Algorithme de Newton dans $\mathbb{R}$

1. Initialisation

   $k = 0$: choix de $x^0 \in \mathbb{R}$ dans un voisinage de x^*.

2. Itération k

$$x^{k+1} = x^k - \frac{f'(x^k)}{f''(x^k)}.$$

3. Critère d'arrêt

   Si $|f'(x^k)| < \varepsilon$, Stop

   Sinon, on pose $k = k + 1$ et on retourne à 2.

inconvénients de la méthode :

1. La méthode peut diverger si le point initial est trop éloigné de la solution,
2. La méthode n'est pas définie si $f''(x^k) = 0$,

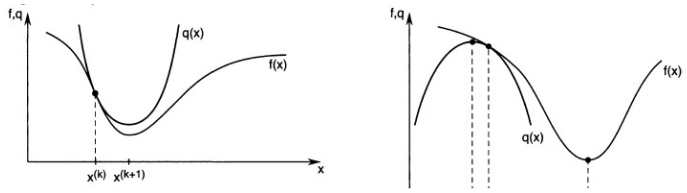


FIGURE 1.6 – Approximation de Newton dans les cas $f''(x) < 0$ et $f''(x) > 0$ resp.

1.3.5 La méthode de la sécante

La méthode de la sécante est une méthode d'optimisation unidimensionnelle, qui permet de trouver le minimum d'une fonction f définie sur un intervalle $[a, b] \subseteq \mathbb{R}$ à une seule variable en utilisant une approximation de la dérivée de la fonction. Elle est basée sur l'interpolation linéaire entre deux points successifs qui encadrent le minimum. Si la deuxième dérivée n"est pas applicable, on remplace $f''(x^k)$ par $\frac{f'(x^k)-f'(x^{k-1})}{x^k x^{k-1}}$ dans les itérations de Newton.

Algorithme La méthode de la sécante dans $\mathbb{R}$

1. Initialisation

   $k = 0$: choix de $x^0 \in \mathbb{R}$ et $x^1 \in \mathbb{R}$ qui encadrent x^*.

2. Itération k

$$x^{k+1} = x^k - f'(x^k)\frac{x^k - x^{k-1}}{f'(x^k) - f'(x^{k-1})}.$$

3. Critère d'arrêt

   Si $|x^{k+1} - x^k| < \varepsilon$, Stop

   Sinon, on pose $k = k + 1$ et on retourne à 2.

Exemple 1.3.3. *Pour $f(x) = x^3 - 3x^2$, avec $x_0 = 1$ et $x_1 = 3$:*

$$f'(x) = 3x^2 - 6x$$

La méthode de la sécante donne :

$$x_2 = 3 - \frac{f(3)}{f(3) - f(1)} \cdot \frac{3-1}{1} = 3 - \frac{-9}{-9-(-3)} \cdot 2 = 2$$

Convergence en une itération pour cet exemple particulier.

1.4 Comparaison des méthodes et critères de choix

Méthode	**Vitesse de convergence**	**Nécessite f' ?**	**Nécessite f'' ?**	**Robustesse**
Bissection	Linéaire (1/2)	Non	Non	Élevée
Section dorée	Linéaire (0.618)	Non	Non	Élevée
Newton	Quadratique	Oui	Oui	Moyenne
Sécante	Superlinéaire (1.618)	Oui	Non	Moyenne

— Utiliser la *section dorée* si f n'est pas différentiable ou si l'on ne dispose pas des dérivées.
— Utiliser la méthode de *Newton* si f'' est disponible et si l'on dispose d'une bonne estimation initiale.
— Utiliser la méthode de la *sécante* comme compromis lorsque f'' est difficile à calculer.