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'extrémum partent d'un principe de solutions réalisables S, en suite 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 principale 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 prendre des valeurs entières.

1.1.2 Méthodes de preuve

Dans ce cours la plupart des énoncés de problèmes ont la forme $A \Rightarrow B$. Pour prouver cette implication, on peut choisir l'une des trois techniques suivante :

- **La preuve directe :** consiste à commencer par le A et avec un certains procédés analytique (suivant le cas du problème) on arrive à verifier le B.
- **La preuve de contraposition :** consiste à commencer la preuve par le non B et avec un certains procédés analytique (suivant le cas du problème) on arrive à verifier le non A en conclusion.
- **La preuve de contradiction :** consiste à commencer la preuve par le A et non B et avec un certains procédés analytique on arrive à une contradiction.

1.2 Optimisation unidimensionnelle

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

$$\text{maximiser ou minimiser } f(x), \quad x \in \mathbb{R}. \tag{1.1}$$

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 :*

$$argmin_{x\in\mathbb{R}}\ f(x), \quad argmax_{x\in\mathbb{R}}\ f(x).$$

La tâche d'optimisation est de trouver un point de l'argument x (ou des points) dans le domaine de $f(x)$ qui maximise (ou minimise) les valeurs de 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,*

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

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

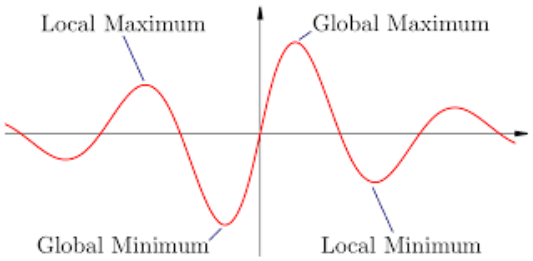


FIGURE 1.1 – Les extremaux d'une fonction

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

$$f(x^*) \leq f(x), \forall x \in B_\epsilon(x^*) \subseteq S. \tag{1.3}$$

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

$$B_\epsilon(x^*) = \{x \in \mathbb{R}^n, \parallel x - x^* \parallel < \epsilon\}. \tag{1.4}$$

Théorème 1.2.1. ***(Condition d'optimalité)***
Soit $f : \mathbb{R} \to \mathbb{R}$ une fonction de C^2. x^* est un minimum (resp. maximum) local ssi,

$$\begin{cases} f'(x^*) = 0, \\ f''(x^*) > 0, \quad \textit{(resp <0).} \end{cases}$$

Remarque 1.2.3. *1. Si $f'(x^*) = 0$ et $f''(x^*) = 0$, il s'agit d'un point d'inflexion.*
*2. La première condition de (**??**), nous donne les points stationnaires.*

Exemple 1.2.1. *Soit $f(x) = x^3 - 3x^2$.*
- $f'(x) = 0 \Rightarrow 3x^2 - 6x = 0 \Rightarrow x = 0 \ \wedge \ x = 2.$
- $f''(x) = 6x - 6 \Rightarrow f''(0) = -6 < 0 \ \wedge \ f''(2) = 6 > 0$

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

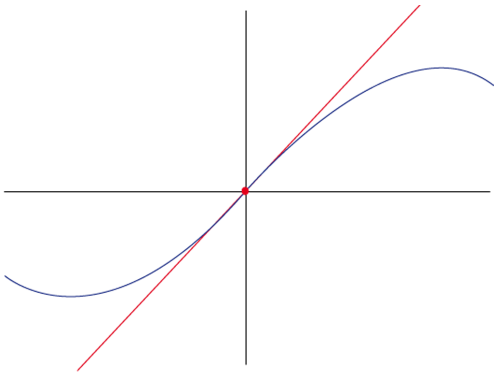


FIGURE 1.2 – Point d 'inflexion

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.

Exemple 1.2.2. *Pour* $S = \mathbb{R}$ *la fonction* $f(x) = e^x$

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 :** La méthode de Bissection, La méthode de section dorée.
- **Méthodes avec dérivée :** La méthode de Sécante, La méthode de 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 La méthode de Bissection

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 de bissection

1. Poser $x_a = a,\ x_b = b$ et $x_m = \frac{(a+b)}{2}$.
2. Calculer $f_a = f(x_a),\ f_b = f(x_b),\ f_m = f(xm)$.
3. **Repeat**
   poser $x_l = \frac{(x_a+x_m)}{2},\ x_r = \frac{(x_m+x_b)}{2}$.
   Calculer $f_l = f(x_l)$ et $f_r = f(x_r)$.
   Soit $f_{min} = min\{f_a, f_b, f_m, f_l, f_r\}$.
   SI $f_{min} = f_a$ ou f_l alors soit $x_b = x_m,\ x_m = x_l,\ f_b = f_m,\ f_m = f_l$.
   Sinon
      SI $f_{min} = f_m$ alors soit $x_a = x_l,\ x_b = x_r,\ f_a = f_l,\ f_b = f_r$.
      Sinon
         Si $f_{min} = f_r$ ou f_b alors soit $x_a = x_m,\ x_m = x_r,\ f_a = f_m,\ f_m = f_r$.
4. **until** $|x_b - x_a|$ soit suffisamment petit.

Remarque 1.3.1. *En présence de plusieurs minimaux locaux, la méthode de la bissection ne garantit pas nécessairement la convergence vers un minimum global.*

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

On pose $x_a = 0,\ x_b = 3,\ x_m = 1.5$.
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* $[x_m, x_b] = [1.5, 3.0]$.
re-calcule x_l, x_r *on aura*

$$x_a = 1.5;\ x_l = 1.875;\ x_m = 2.25;\ x_r = 2.625;\ x_b = 3.$$

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* $[x_a, x_m] = [1.5, 2.25]$.
re-calcule x_l, x_r *on aura*

$$x_a = 1.5;\ x_l = 1.6875;\ x_m = 1.875;\ x_r = 2.0625;\ x_b = 2.25.$$

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* $x = 2$.

Dans des situations pratique, 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. Alors l'encadrement du minimum se détermine par l'algorithme suivant.

Algorithme d'encadrement du minimum

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

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.*

$$N \approx \frac{log(b-a)/\epsilon)}{log(2)}, \quad \epsilon \text{ est la tolérance d'erreur.} \tag{1.5}$$

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 estimer 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

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

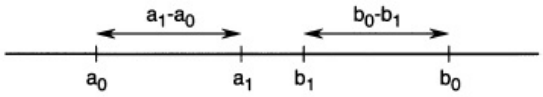
1.3.2 La méthode de Section dorée

Supposons que la fonction f est unimodals sur l'intervalle $[a, b]$. L'idée est d'évaluer f en deux points de l'intervalle $[a, b]$, voir la figure au dessous, dans le sens où,

$$a_1 - a = b_1 - b = \rho(b - a), \quad \rho < \frac{1}{2}.$$

Ce qui donne,

$$a_1 = a + \rho(b - a) \text{ et } b_1 = a + (1 - \rho)(b - a).$$



Maintenant, on évalue $f(a_1)$ et $f(b_1)$. Si $f(a_1) < f(b_1)$, alors le minimum se trouve dans $[a, b_1]$. Sinon le minimum est dans $[a_1, b]$.

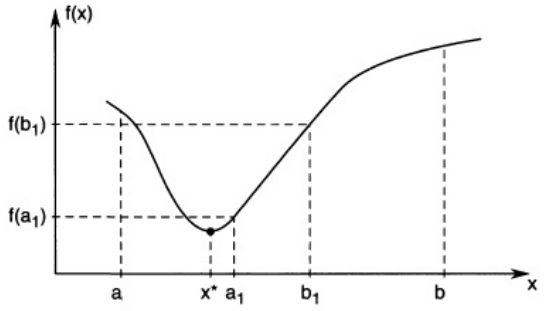
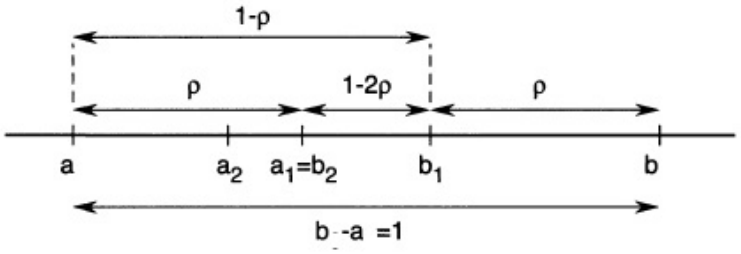


FIGURE 1.3 – Le minimum est dans $[a, b_1]$.

Pour trouver la valeur de ρ qui soit acceptable on cherche d'autres deux point dans $[a, b_1]$. Et comme elle montre la figure au dessous, $b_1 - a = 1 - \rho$ et $b_1 - b_2 = 1 - 2\rho$,
Alors,la résolution de l'équation $\rho(1 - \rho) = 1 - 2\rho$, donne

$$\rho_1 = \frac{3 + \sqrt{5}}{2} \text{ et } \rho_2 = \frac{3 - \sqrt{5}}{2}$$

Et puisque $\rho < \frac{1}{2}$, alors on prend pour toutes les itération $\rho = \frac{3-\sqrt{5}}{2} = 0.382$.



Code Python : la fonction est $f(x) = x^3 - 3x^2$.

```
import math as m
def f(x) :
    return x**3 - 3*x**2
ips=float(input("Donner l'erreur du clcule "))
a=float(input("Donner l'extremité de l'intervalle a= "))
b=float(input("Donner l'extremité de l'intervalle b= "))
i,ro= 0, 0.382
while m.fabs(b-a)>ips :
    na = a + ro*(b - a)
    nb = a + (1 - ro)*(b - a)
    if f(na)<f(nb) :
        a, b = a, nb
        print("Le nouveau intervalle est", [a, nb])
    else :
        a,b=na,b
        print("Le nouveau intervalle est", [na, b])
```

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

1.3.3 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,

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 $f''(x^k) = 0$,

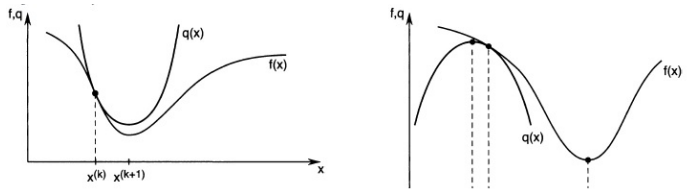


FIGURE 1.4 – Approximation de Newton dans les cas $f''(x) < 0$ et $f''(x) > 0$ resp.

1.3.4 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.