

# IFT 3515

## Programmation non-linéaire

Fabian Bastin  
DIRO  
Université de Montréal

## L'objet d'intérêt

Nous considérons le problème

$$\min_{x \in S} f(x),$$

où  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  et  $S$  est un sous-ensemble de  $\mathbb{R}^n$ .

- $f$ : fonction objectif (ou fonction coût)
- $S$ : ensemble réalisable
- $n$ : dimension du problème

Tout point  $x \in S$  est appelé *solution réalisable* (ou simplement parfois *solution*).

## Optimum global

Idéalement, nous voudrions trouver un **minimum global** de  $f$  dans  $S$ , i.e., un point dans  $S$  où la fonction atteint sa plus petite valeur. Une définition formelle est:

### Définition (Minimiseur global)

*Un point  $x^*$  est un minimiseur global de  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  dans  $S$  si  $f(x^*) \leq f(x)$  pour tout  $x \in S$ .*

$x^*$  est appelé *solution optimale* ou *minimiseur de  $f$* .  $f(x^*)$  est dite *valeur optimale de  $f$* , et nous la noterons parfois  $f^*$ .

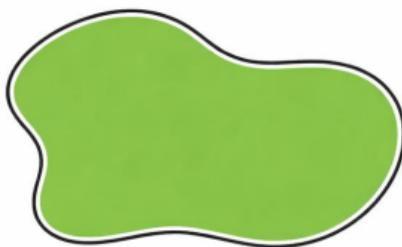
Un problème d'optimisation n'a pas nécessairement une solution unique. Il peut ne pas y avoir de solution ou plus d'une solution optimale. Nous définirons

$$\arg \min f(x) \stackrel{\text{def}}{=} \{x \mid f(x) \leq f(y) \forall y \in S\}.$$

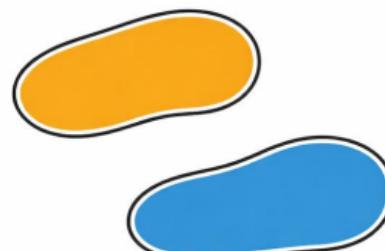
## Optimisation continue

Nous nous limiterons à l'optimisation continue, i.e. les variables de décisions sont continues.

Nous imposerons aussi que l'ensemble réalisable  $S$  est connexe, i.e. quels que soient  $x, y \in S$ , il existe un chemin  $\mathcal{P}$  reliant  $x$  à  $y$  entièrement contenu dans  $S$ .



Connexe



Non-connexe

## Ensemble ouvert, borné, compact

### Définition (ensemble ouvert)

*X est un ensemble ouvert si  $\forall x \in X, \exists \epsilon > 0$  t.q. la boule ouverte centrée en x de rayon  $\epsilon$ ,  $\mathcal{B}(x, \epsilon) = \{z \in \mathbb{R}^n | \|z - x\| < \epsilon\}$ , est incluse dans X.*

(Contre)-exemple: dans  $R^2$  l'ensemble

$$\Gamma = \{[x, y] \in \mathbb{R}^2 | x \in (a, b), y = 0\}$$

n'est pas un ensemble ouvert puisque nous ne pouvons pas modifier la valeur de y tout en restant dans  $\Gamma$ .

### Définition (Ensemble borné)

*Un ensemble X est borné si  $\exists M$  t.q.  $\forall x \in X, \|x\| \leq M$ .*

### Définition (Ensemble compact)

*Dans  $\mathbb{R}^n$ , un ensemble X est compact si et seulement si il est fermé et borné.*

# Voisinage

## Définition (voisinage)

*Un voisinage  $\mathcal{V}(x)$  d'un point  $x$  est un ensemble ouvert contenant  $x$ .*

Nous travaillerons principalement avec la norme 2 (norme euclidienne):

$$\|z - x\|_2 = \sqrt{(z - x)^T(z - x)}$$

Dès lors, quel que soit  $\mathcal{V}(x)$ ,  $\exists \epsilon > 0$  tel que  $\mathcal{B}(x, \epsilon) \subseteq \mathcal{V}$ . Nous ne perdons dès lors aucune généralité en travaillant avec une boule ouverte centrée en  $x$ .

# Optimum local

Un minimum global peut être difficile à obtenir. La plupart des algorithmes cherchent seulement un **minimum local**, qui est un point qui atteint la plus petite valeur de  $f$  dans son voisinage.

## Définition (Minimiseur local)

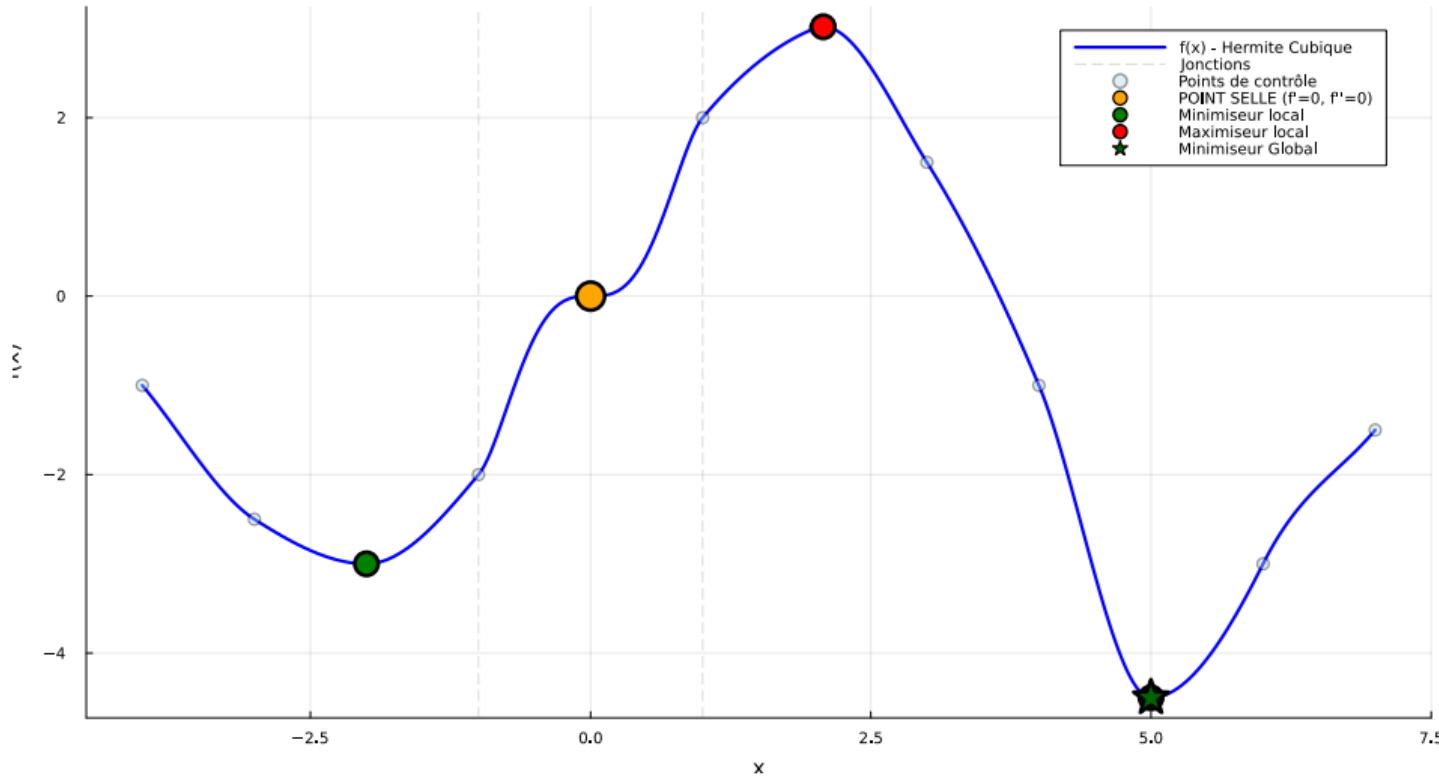
*Un point  $x^*$  de  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  dans  $S$  est un minimiseur local s'il existe un voisinage  $\mathcal{V}$  de  $x^*$  tel que  $f(x^*) \leq f(x)$  pour  $x \in \mathcal{V} \cap S$ .*

Un point qui satisfait cette définition est parfois appelé minimiseur local faible, pour le distinguer d'un minimiseur local strict.

## Définition (Minimiseur local strict)

*Un point  $x^*$  est un minimiseur local strict (aussi appelé minimiseur local fort) s'il existe un voisinage  $\mathcal{V}$  de  $x^*$  tel que  $f(x^*) < f(x)$  pour tout  $x \in \mathcal{V} \cap S$  avec  $x \neq x^*$ .*

# Illustration



## Optimum local isolé

### Définition (Minimiseur local isolé)

*Un point  $x^*$  est un minimiseur local isolé s'il existe un voisinage  $\mathcal{V}$  de  $x^*$  tel que  $f(x^*) < f(x)$  pour tout  $x \in \mathcal{V} \cap S$  avec  $x \neq x^*$  et il n'y a pas d'autre minimiseur local dans  $\mathcal{V}$ .*

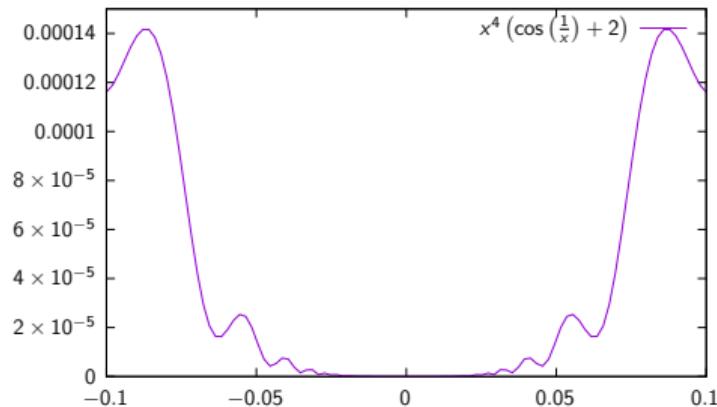
Un minimiseur local strict n'est pas toujours isolé, mais tout minimiseur local isolé est strict.

## Exemple

(Tiré de Nocedal et Wright) Considérons la fonction suivante:

$$f(x) = x^4 \cos \frac{1}{x} + 2x^4.$$

$f(x) \in C^2$  (deux fois continûment différentiable) et a un minimum local strict en  $x^* = 0$ . Toutefois, tout voisinage de  $x^*$  contient d'autres minimums locaux stricts.



## Ensemble réalisable

Généralement, l'ensemble réalisable  $S$  sera représenté sous forme de contraintes d'égalités et d'inégalités:

$$\begin{aligned}c_i(x) &= 0, \quad i \in \mathcal{E}, \\c_i(x) &\leq 0, \quad i \in \mathcal{I}.\end{aligned}$$

- $\mathcal{E}$ : ensemble des contraintes d'égalité
- $\mathcal{I}$ : ensemble des contraintes d'inégalité

En résumé, nous considérerons des problèmes de la forme

$$\begin{aligned}\min_x \quad & f(x) \\ \text{t.q. } & c_i(x) = 0, \quad i \in \mathcal{E}, \\ & c_i(x) \leq 0, \quad i \in \mathcal{I}.\end{aligned}$$

## Reformulations

La formulation précédente est assez générale, et couvre d'autres variantes par simple reformulation.

Nous pouvons par exemple considérer un problème de maximisation avec la transformation

$$\max_{x \in S} f(x) = -\min_{x \in S} (-f(x))$$

Les problèmes sont aussi définis à une constante près, comme

$$\min_x (f(x) + \kappa) = \kappa + \min_x f(x)$$

## Reformulations des contraintes

Nous pourrions aussi ne considérer que des contraintes d'égalité, en introduisant des variables d'écart:

$$c_i(x) \leq 0$$

$$\Leftrightarrow$$

$$c_i(x) + s_i = 0, \quad s_i \geq 0$$

Similairement, nous pourrions juste écrire des contraintes d'inégalité:

$$c_i(x) = 0$$

$$\Leftrightarrow$$

$$c_i(x) \leq 0,$$

$$-c_i(x) \leq 0$$

## Structure générale d'un algorithme d'optimisation

On fixe un point de départ  $x_0$  et on génère une séquence de points  $x_1, x_2, \dots$  sur le principe suivant. Poser  $k = 0$ , et calculer  $f(x_0)$ . Tant qu'une condition d'arrêt n'est pas satisfaite, répéter

1. Poser  $k := k + 1$ .
2. Déterminer une solution candidate  $x_{k+1}$ , et calculer  $f(x_{k+1})$ .

$k$  est appelé *indice d'itération*. On voudrait que le nombre d'itérations soit fini, et si ce n'est pas le cas, que la séquence de point converge vers une solution, i.e.

$$\exists x^* \text{ tel que } \lim_{n \rightarrow \infty} x_n = x^*,$$

et

$$x^* = \arg \min f(x).$$

## Convergence

La convergence peut être testée de différentes manières, dépendamment du problème et de la technique d'optimisation utilisée. Par exemple, on peut décider de s'arrêter quand la réduction de la fonction objectif entre deux itérations devient négligeable:

$$|\Delta f| = |f(x_{k+1}) - f(x_k)| \leq \epsilon,$$

où  $\epsilon$  est la **tolérance d'optimisation** pour la fonction objectif  $f$ .

Alternativement, on peut s'arrêter lorsque le changement entre les itérés devient non significatif:

$$\|x_{k+1} - x_k\|_k \leq \epsilon,$$

où  $\|\cdot\|_k$  est une certaine norme (pouvant théoriquement dépendre de l'indice d'itération  $k$ ).

## Conditions d'optimalité

Supposons que nous nous arrêtons à l'itération  $\ell$ . Ceci garantit-il que  $x_\ell$  est une solution du problème d'optimisation?

Pas nécessairement!

Nous voudrions pouvoir établir des **conditions d'optimalité**, qui peuvent être vérifiées, et telles que si celles-ci sont remplies,  $x_\ell$  soit solution.

En pratique, celles-ci ne seront jamais satisfaites, mais nous voudrions pouvoir détecter que  $x_\ell$  est “quasi”-optimal, dans le sens où

$$f(x_\ell) \leq f(x) + \delta$$

pour tout  $x \in S$ , et  $\delta > 0$  petit.

## Pas de méthode miracle

[...] for any algorithm, any elevated performance over one class of problems is offset by performance over another class

*No free lunch theorems for optimization*, David H. Wolpert and William G. Macready, IEEE Transactions on Evolutionary Computation 1(1), pp. 67–82, 1997.

## Avertissement

Beaucoup de fausses idées circulent sur les propriétés et les algorithmes d'optimisation. Plusieurs sont recensées dans le document “Myths and Counterexamples in Mathematical Programming”.

Nous tâcherons d'éviter ces pièges dans le cours, mais vous êtes invités à me corriger au besoin!