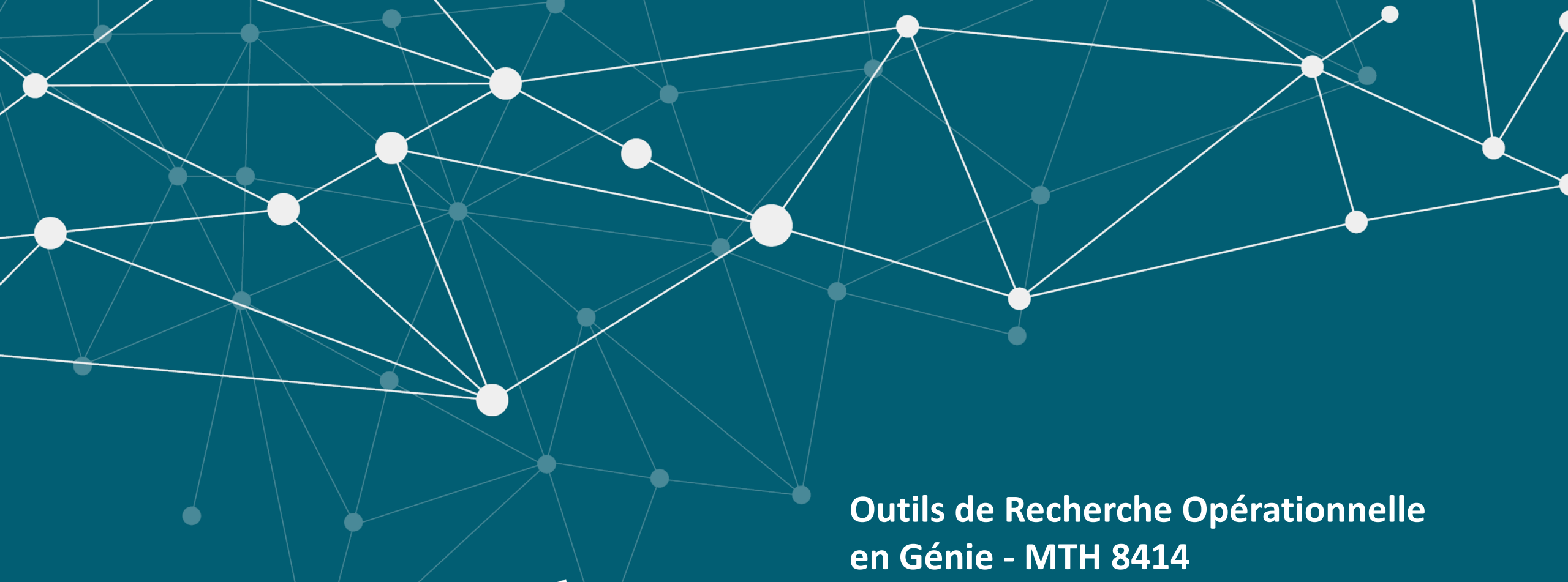


Outils de Recherche Opérationnelle en Génie - MTH 8414

Programmation non linéaire

- Intro et difficultés
- Les cas simples
- Méthodologies

Louis-Martin Rousseau

Bureau : A520.21 Tel.: #4569
louis-martin.rousseau@polymtl.ca

Introduction à la Programmation Non-Linéaire

Un programme non linéaire est similaire à un programme linéaire, car il est constitué:

- D'une fonction objectif
- Des contraintes et des bornes sur les variables

Toutefois ces équations n'ont pas à être linéaires

Elles sont généralement données comme des fonctions (analytiques)

Il y a de nombreuses applications pratiques de la PNL:

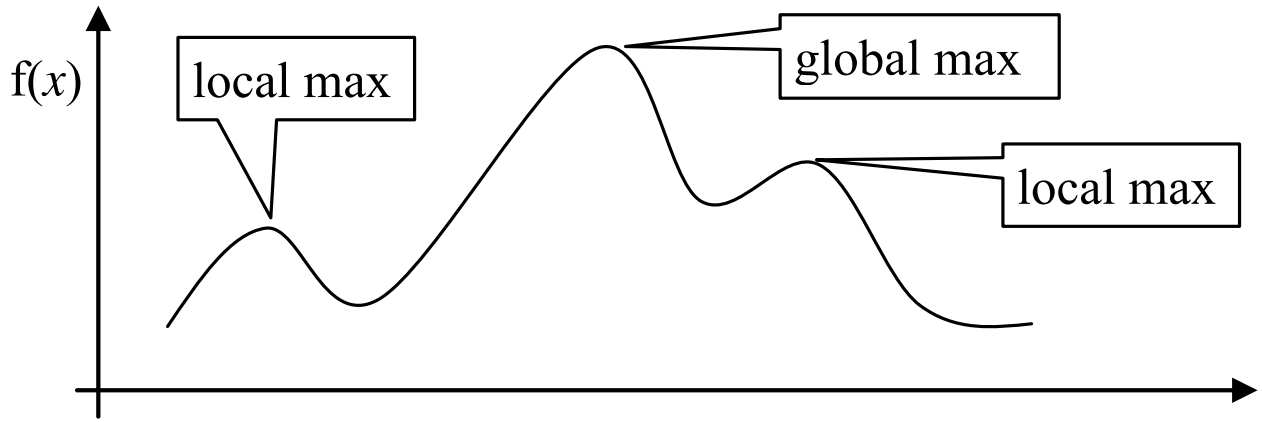
- Problème d'optimisation en finance
- Problème d'optimisation en énergie
- Problème d'optimisation en (pétro) chimie
- …

Un PNL est toutefois beaucoup plus difficile à résoudre qu'un PL.

Voyons pourquoi…

Raison 1: Local ou global cet optimum ?

- En l'absence de convexité (comme pour les PL) les méthodes de résolution de PNL ne peuvent garantir d'identifier la solution optimale
- Elles font généralement une recherche dans l'espace, basée sur:
  - Un point courant
  - La valeur de l'objectif à ce point
  - La valeur du gradient de la fonction objectif
  - La matrice Hessien (dérivée seconde) de la fonction objectif
- Ces informations permettent d'identifier un maximum (ou minimum) local, mais pas de dire s'il est global.

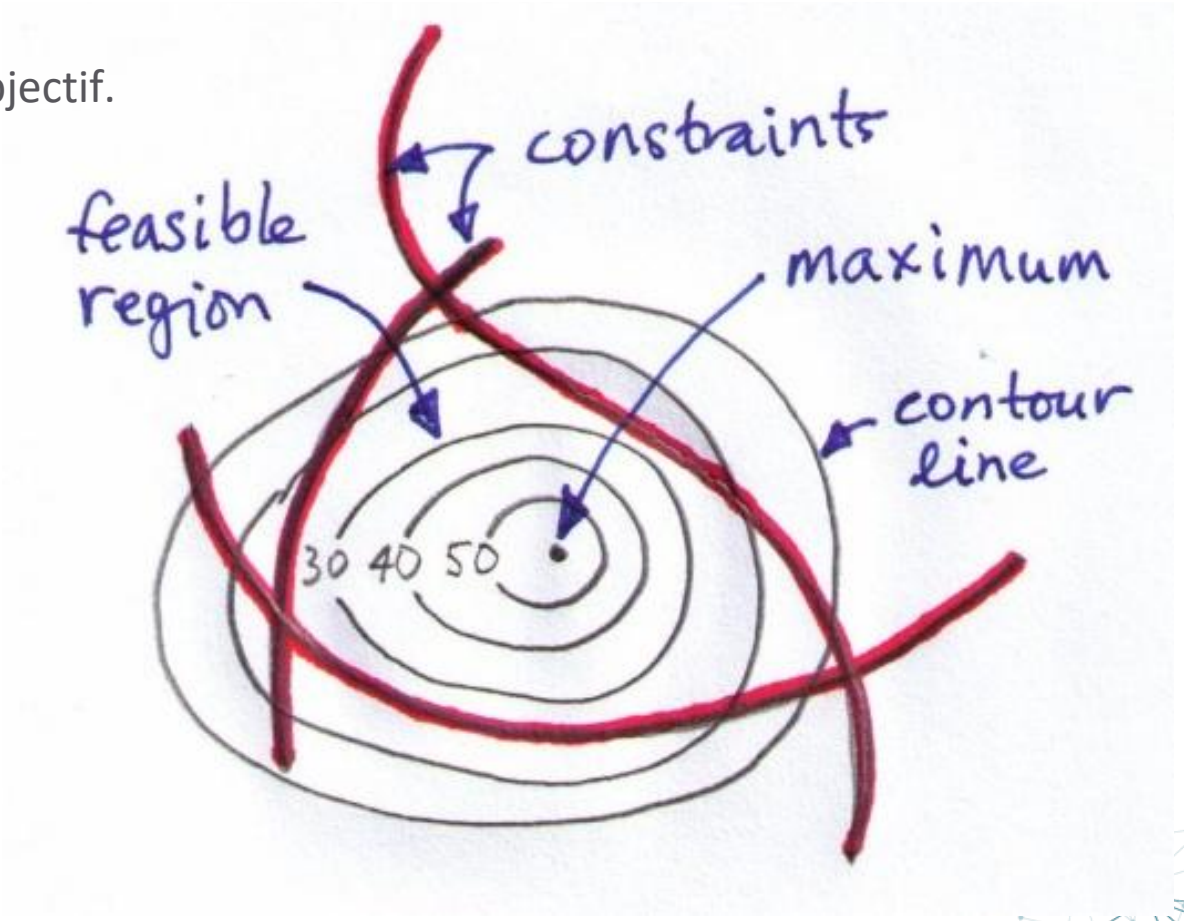


Raison 2: L’optimum et les points extrêmes.

En PL, s’il existe une solution optimale, alors il existe un point extrême du polyèdre qui est optimal.

En PNL, l’optimum peut se trouver n’importe où

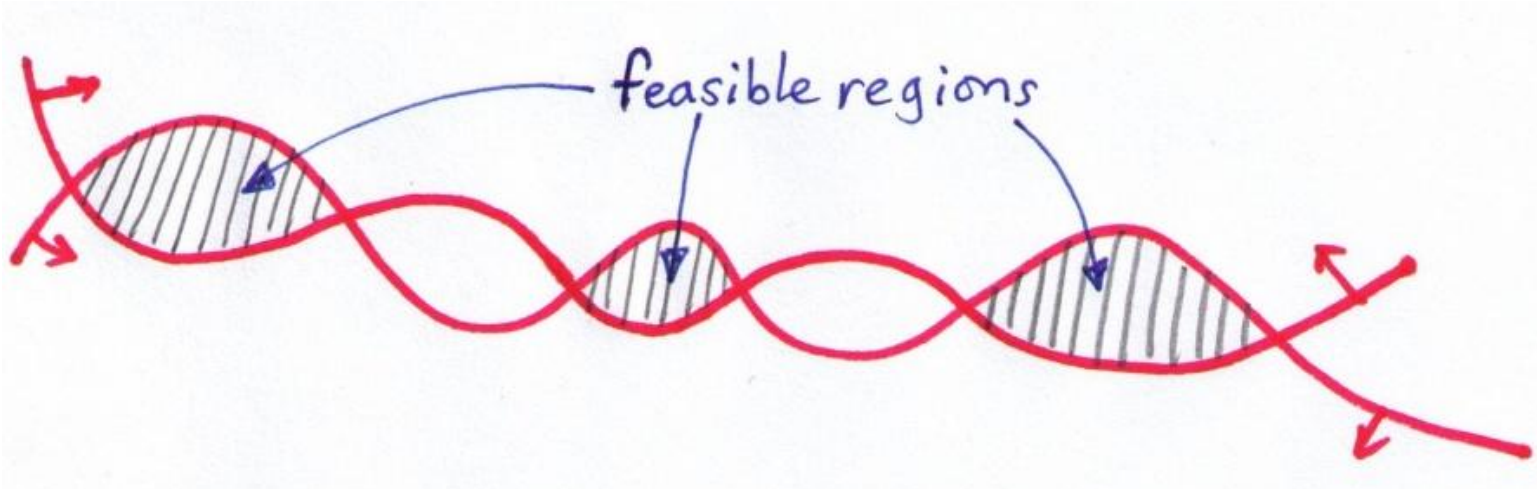
- En rouge les lignes de la région réalisables
- En noir des courbes de niveau de la fonction objectif.



Raison 3: Régions réalisables déconnectées

Rien ne garantit que l'espace des solutions réalisables est connecté.

Donc, même si on trouve l'optimum d'une région, rien ne nous dit qu'il n'existe pas d'autres régions réalisables…



Raison 4: L’importance du point de départ

Comme on ne peut pas garantir l’obtention d’un optimum global, le point d’où on débute la recherche joue un rôle important sur la solution que nous pouvons trouver.

En effet la plupart des solveurs non linéaires font une recherche locale

- À partir d’une solution de départ ou solution courante
- On détermine la meilleure direction (avec le gradient)
- On fait un « pas » dans cette direction

Une des solutions est de redémarrer le solveur régulièrement à partir de points différents.

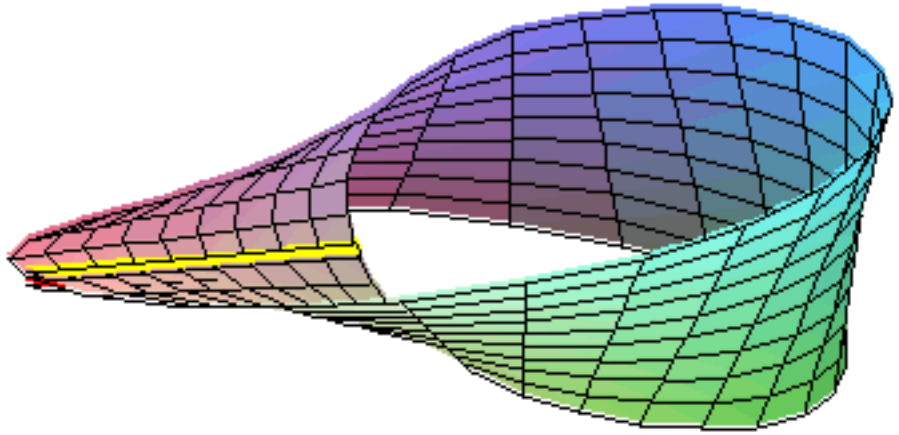
Raison 5: Trouver un point de départ

- Il peut être parfois difficile de trouver un point de départ réalisable.
  - En PL, il existe une « phase 1 » qui permet toujours de trouver une solution initiale ou de montrer qu'il n'en existe aucun.
- En PNL, on commence en minimisant une distance vis-à-vis de la zone réalisable, comme le carré des violations par exemple.
- Donc si on optimise ce deuxième problème et on trouve une solution optimale dont le coût est 0, alors on a identifié une solution réalisable à notre problème d'origine
- Toutefois, ce deuxième problème est lui-même un problème NL…
- Dans ce cas, les solveurs peuvent donc rester pris dans un minimum local dont la valeur n'est pas 0… ce qui implique
  - Qu'on ne peut pas optimiser le problème
  - On ne peut pas prouver qu'il est irréalisable.

Raison 6: Dures égalités

Si on a une courbe très « tordue » il peut s'avérer difficile de trouver un point qui tombe « pile » sur la courbe.

De plus, lors de l'optimisation (via recherche locale) il risque d'être encore plus difficile de rester sur cette courbe.



Raison 7: Un zoo de méthodologies

- La PNL regroupe sous une même bannière une vaste gamme de techniques associées à chaque type de fonctions (quadratiques, convexes, etc.)
- Face à un problème, quelle technique appliquer ?
- Les solveurs nécessitent parfois que les fonctions linéaires respectent des conditions particulières… Il peut donc être difficile d'identifier le bon solveur à utiliser.
- Il existe très peu d'outils permettant d'analyser un problème NL, d'identifier ses caractéristiques et de choisir un bon solveur.
- Différents solveurs vont donner différentes solutions…
- Différentes modélisations vont donner différentes solutions…

Raison 8: Difficulté d’usage

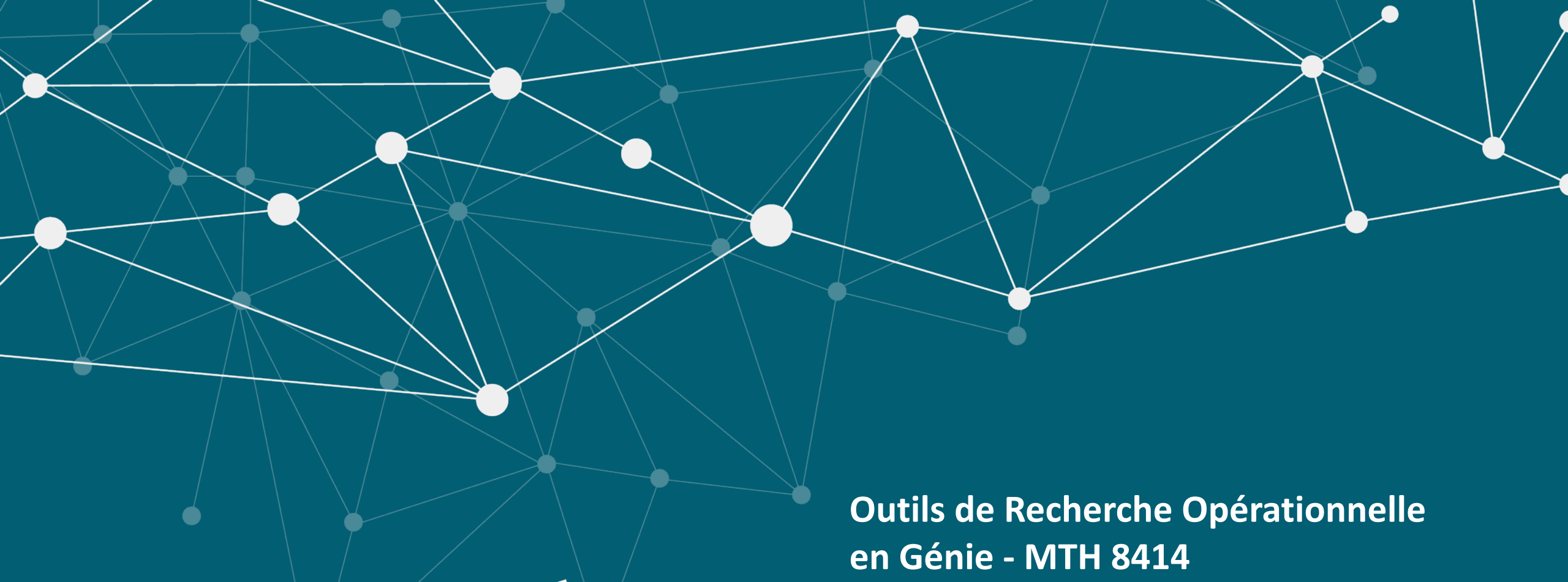
Les solveurs disponibles ont généralement un très grand nombre de paramètres à ajuster (> 50).

Optimiser ces paramètres peut s’avérer une tâche complexe

- Quel type d’optimisation locale, quelle tolérance, inversion de la base, etc.

Souvent, on a besoin des dérivées des fonctions : elles peuvent être difficiles à trouver.

Pas de standard pour l'entrée des données : difficile de passer d'un solveur à un autre.



Outils de Recherche Opérationnelle en Génie - MTH 8414

Programmation non linéaire

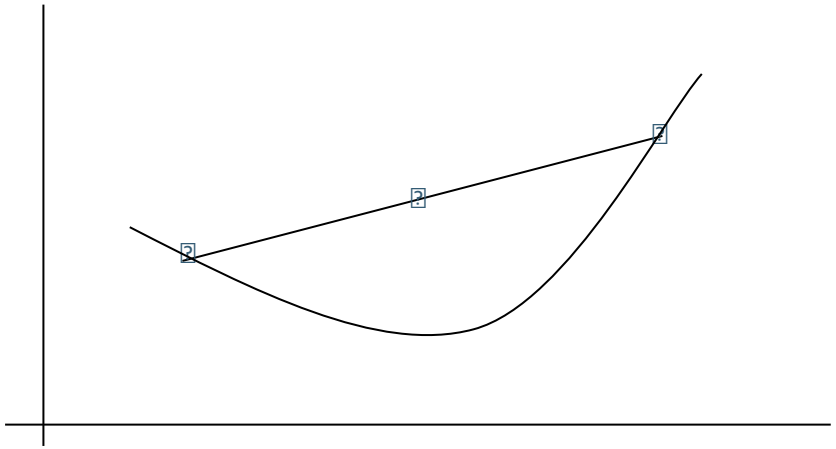
- Intro et difficultés
- Les cas simples
- Méthodologies

Louis-Martin Rousseau

Bureau : A520.21 Tel.: #4569
louis-martin.rousseau@polymtl.ca

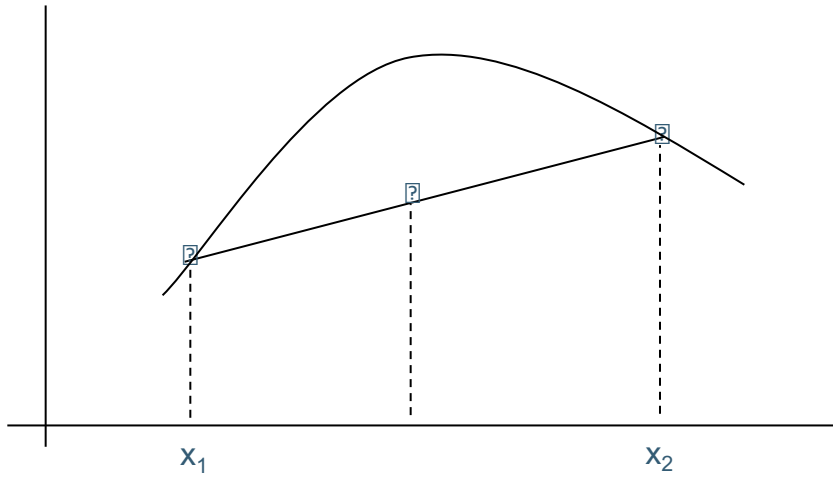
Fonction objectif convexe

$$\lambda f(x_1) + (1-\lambda) f(x_2) \geq f(\lambda x_1 + (1-\lambda) x_2) \text{ avec } 0 \leq \lambda \leq 1$$



Fonction objectif concave

$$\lambda f(x_1) + (1-\lambda) f(x_2) \leq f(\lambda x_1 + (1-\lambda) x_2) \text{ avec } 0 \leq \lambda \leq 1$$



Conditions de Karush-Kuhn-Tucker

$$\max \quad f(x)$$
$$s.t. \quad g_i(x) = b_i, \qquad i = 1, \dots, m$$

conditions pour que x soit maximum local

$$\nabla f(x) = \sum_i^m v_i \nabla g_i(x)$$
$$v_i \geq 0, \qquad i = 1, \dots, m$$

faisabilité duale

$$g_i(x) \leq b_i \quad i = 1, \dots, m$$

faisabilité primale

$$v_i(b_i - g_i(x)) = 0 \quad i = 1, \dots, m$$

complémentarité

Programmation Quadratique

$f(x)$ est quadratique, i.e. inclus les termes x_j^2 et $x_j x_k \quad (j \neq k)_j$

Les contraintes sont toutes linéaires $g_i(x)$

$$X = (x_1,\ x_2,\ \ldots,\ x_n) \quad t.q.$$

$$\min \frac{1}{2} x^t Q x + c^t x$$
$$\text{s.c. } Ax\ (\leq, =, \geq)\ b$$
$$l \leq x \leq u$$

Un solveur commercial peut résoudre :

min obj convexe avec Q positif semi-défini : $x^tQx \geq 0$

ou

max obj concave avec Q négatif semi-défini : $x^tQx \leq 0$

Problème de programmation convexe

La **fonction objectif:** $f(X)$ est **concave** si on max et convexe si on min.

Chaque **contrainte**: $g_i(X)$ est **convexe**.

Algorithmes de type « gradient »

- par exemple : gradient réduit généralisé

Algorithmes d'optimisation non contraint séquentiels

- transforme le problème contraint en envoyant les contraintes dans l'objectif (méthodes de pénalités ou de barrières)

Algorithmes d'approximations séquentielles

- remplace l'objectif par une succession d'approximations linéaires ou quadratiques

Problème de programmation séparable

La fonction objectif $f(x)$ est concave et séparable, i.e.

$$f(x) = f_1(x_1) + f_2(x_2) + \cdots + f_n(x_n) = \sum_{j=1}^{n} f_j(x_j)$$

Chaque contrainte $g_i(x)$ est convexe et séparable pour chaque i.

Séparable

$$x_1^2 + 1/x_2 - 2x_3 = f_1(x_1) + f_2(x_2) + f_3(x_3)$$
$$x_1^2 + 5x_1 - x_2 = g_1(x_1) + g_2(x_2)$$

Non séparable

$$x_1x_2 + 3x_2 + x_2^2 = f_1(x_1, x_2) + f_2(x_2)$$
$$1/(x_1 + x_2) + x_3 = g_1(x_1, x_2) + g_2(x_3)$$

Fonctions linéaires par morceaux

Soit le problème suivant:

Minimize:

$$\sum_{j \in J} f_j(x_j)$$

Subject to:

$$\sum_{j \in J} a_{ij} x_j \gtrless b_i \qquad \forall i \in I$$

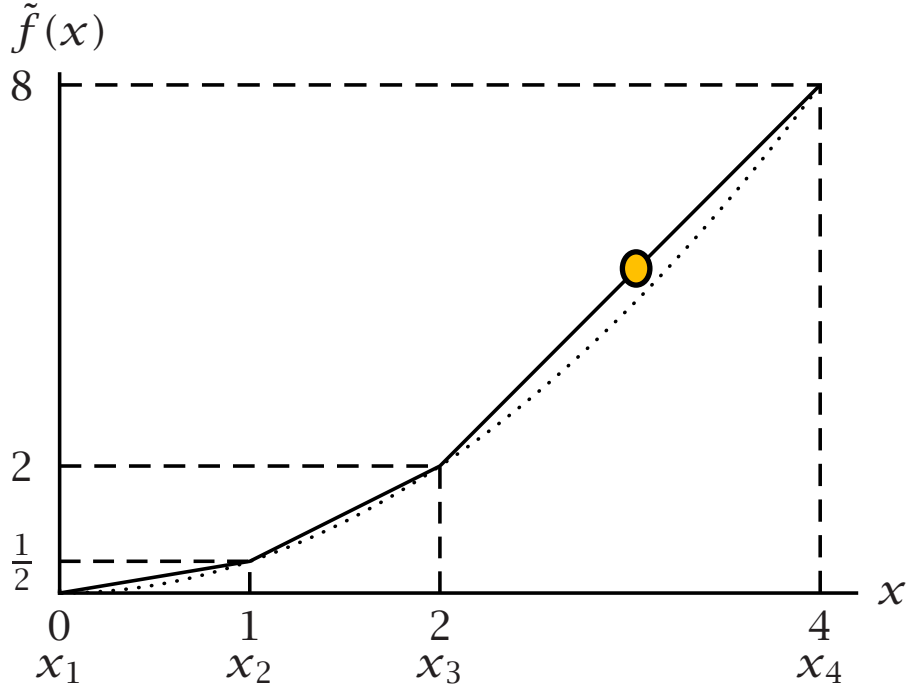
$$x_j \geq 0 \qquad \forall j \in J$$

Ici on remarque que l'objective, bien que non linéaire, est séparable.

C'est-à-dire que l'objectif est une somme de fonctions définies sur une variable à la fois.

Fonctions linéaires par morceaux

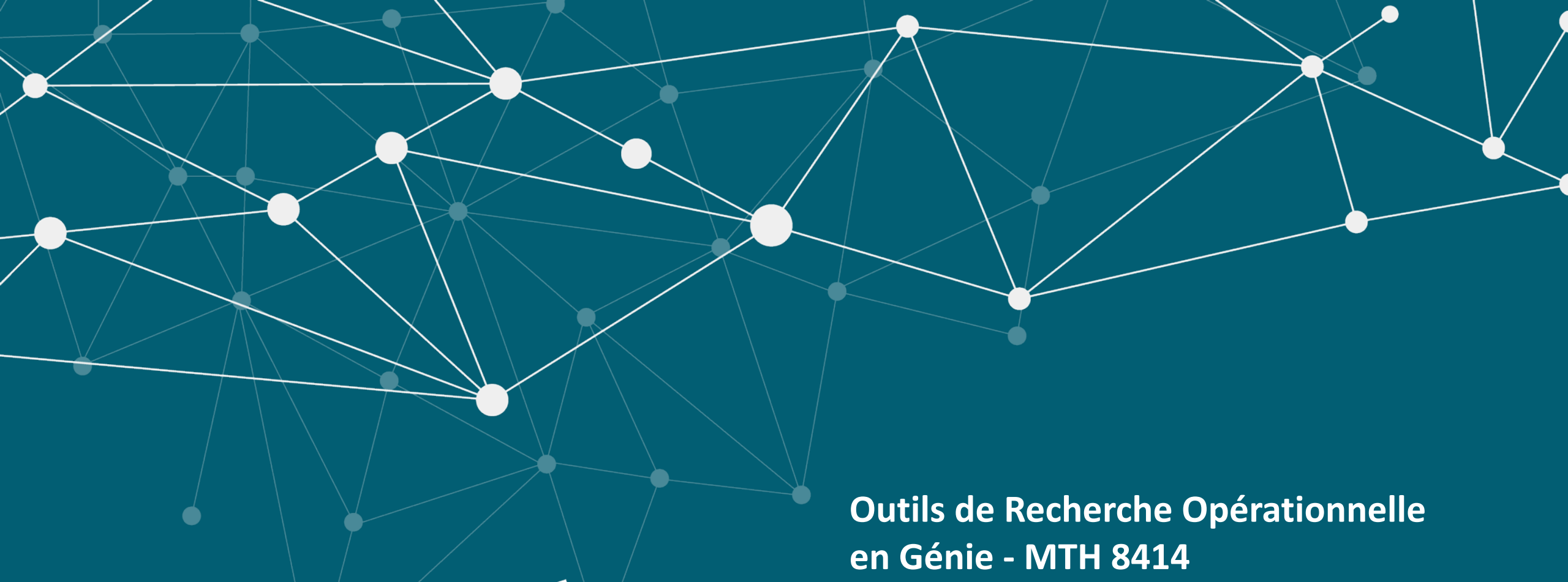
Soit la fonction $f(x) = \frac{1}{2}x^2$



Soit x_1, x_2, x_3, x_4 des points de « cassure » (breakpoints) auquel on évalue la fonction f(x) (soit 0,1,2,4)

On approxime donc linéairement tout point situé entre deux cassures, par exemple f(3) = ½*f(2) + ½*f(4) = ½*1 + ½*8 = 5.

La majorité des solveurs supporte les fonction linéaires par morceaux.



Outils de Recherche Opérationnelle en Génie - MTH 8414

Programmation non linéaire

- Intro et difficultés
- Les cas simples
- Méthodologies

Louis-Martin Rousseau

Bureau : A520.21 Tel.: #4569
louis-martin.rousseau@polymtl.ca

Méthodologie

On optimise les problèmes à une variables à l'aide des dérivées premières et secondes.

Nous utiliserons les mêmes concepts pour les problèmes comportant plusieurs variables.

Le gradient correspond à la dérivée première, alors que la matrice hessienne représente les dérivés secondes.

Le Gradient

Rappel sur le gradient (∇):

Pour une fonction f sur les variables $x_1, x_2, \ldots, x_n$:

$$\nabla f = \left[\frac{\partial f}{\partial x_1} \quad \frac{\partial f}{\partial x_2} \quad \ldots \quad \frac{\partial f}{\partial x_n}\right]$$

Exemple : $f = 15x_1 + 2(x_2)^3 - 3x_1(x_3)^2$

$$\nabla f = [15 - 3(x_3)^2 \quad 6(x_2)^2 \quad -6x_1x_3]$$

La matrice hessienne

La matrice hessienne (∇^2) de $f(x_1, x_2, \ldots, x_n)$ est :

$$\nabla^2 f = \begin{bmatrix} \frac{\partial f^2}{\partial {x_1}^2} & \frac{\partial f^2}{\partial x_1 \partial x_2} & \cdots & \frac{\partial f^2}{\partial x_1 \partial x_n} \\ \frac{\partial f^2}{\partial x_2 \partial x_1} & \frac{\partial f^2}{\partial {x_2}^2} & \cdots & \frac{\partial f^2}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f^2}{\partial x_n \partial x_1} & \frac{\partial f^2}{\partial x_n \partial x_2} & \cdots & \frac{\partial f^2}{\partial {x_n}^2} \end{bmatrix}$$

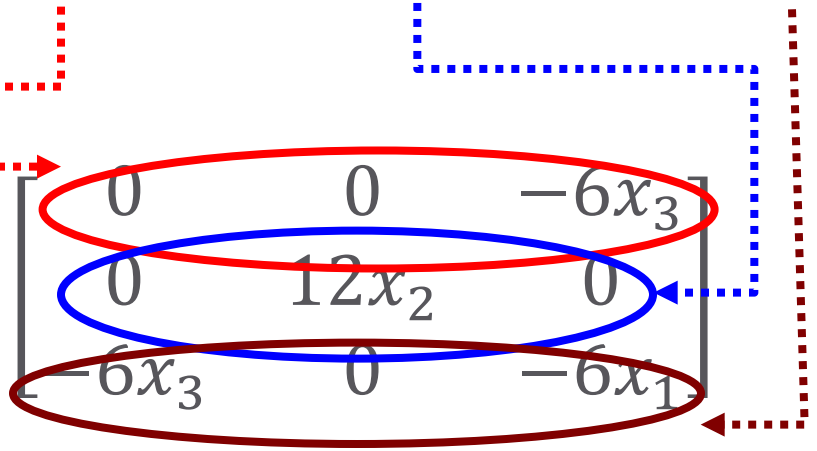
Exemple de matrice hessienne

Exemple (vue précédemment):

$$f = 15x_1 + 2(x_2)^3 - 3x_1(x_3)^2$$

$$\nabla f = [15 - 3(x_3)^2 \quad 6(x_2)^2 \quad -6x_1x_3]$$

$$\nabla^2 f = \begin{bmatrix} 0 & 0 & -6x_3 \\ 0 & 12x_2 & 0 \\ -6x_3 & 0 & -6x_1 \end{bmatrix}$$



Optimisation non linéaire sans contraintes

La procédure d'optimisation sur plusieurs variables fonctionne comme suit:

1. Résoudre le problème qui consiste à trouver les points pour lesquels le gradient est nul.

2. Calculer le Hessien de la fonction objectif et l'évaluer pour chacun des points obtenus en 1.
   - Si la matrice est "définie positive" alors le point est un minimum local.
   - Si la matrice est "définie négative" alors le point est un maximum local.

Matrice définie positive/négative

Une matrice est “définie positive” si toutes ses valeurs propres sont positives. (> 0)

Une matrice est “définie négative” si toutes ses valeurs propres sont négatives. (< 0)

Une matrice est “semi-définie positive” si toutes ses valeurs propres sont positives ou nulles. (≥ 0)

Une matrice est “semi-définie négative” si toutes ses valeurs propres sont négatives nulles. (≤ 0)

Matrice Exemple

Soit la matrice A : $$A = \begin{bmatrix} 2 & 4 & 5 \\ -5 & -7 & -1 \\ 1 & 1 & 2 \end{bmatrix}$$

Les valeurs propres de *A* sont :

$$\lambda_1 = -3.702 \qquad \lambda_2 = -2 \qquad \lambda_3 = 2.702$$

Cette matrice est indéfini, que peut-on dire ?

Un exemple de PNL sans contrainte

On considère le problème :

Minimiser $f(x_1, x_2, x_3) = (x_1)^2 + x_1(1 - x_2) + (x_2)^2 - x_2x_3 + (x_3)^2 + x_3$

D'abord, trouvons le gradient pour les x_i :

$$\nabla f = \begin{bmatrix} 2x_1 + 1 - x_2 \\ -x_1 + 2x_2 - x_3 \\ -x_2 + 2x_3 + 1 \end{bmatrix}$$

Un exemple de PNL sans contrainte

Ensuite, mettre le gradient = 0 :

$$\nabla f = 0 \;\Rightarrow\; \begin{bmatrix} 2x_1 + 1 - x_2 \\ -x_1 + 2_2 - x_3 \\ -x_2 + 2x_3 + 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Nous avons un système à trois équations et trois inconnues, que nous pouvons résoudre:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix}$$

Un exemple de PNL sans contrainte

Nous avons donc un candidat à valider.

Trouvons la matrice hessienne :

$$\nabla^2 f = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix}$$

Un exemple de PNL sans contrainte

Les valeurs propres de cette matrice sont:

$$\lambda_1 = 3.414 \qquad \lambda_2 = 0.586 \qquad \lambda_3 = 2$$

Comme toutes les valeurs propres sont > 0, alors la matrice hessienne est définie positive.

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix}$$ est donc un minimum.

Méthode de résolution

- Dans l’exemple précédent, on avait un système à trois équations linéaires et trois inconnus.
- Pour certains autres problèmes, nous pourrions obtenir des équations non linéaires.
- On doit alors résoudre un système d’équations non linéaires… ce qui n’est pas simple.
- Pour éviter ce problème, les PNL sont généralement résolus numériquement.

Méthode de Newton

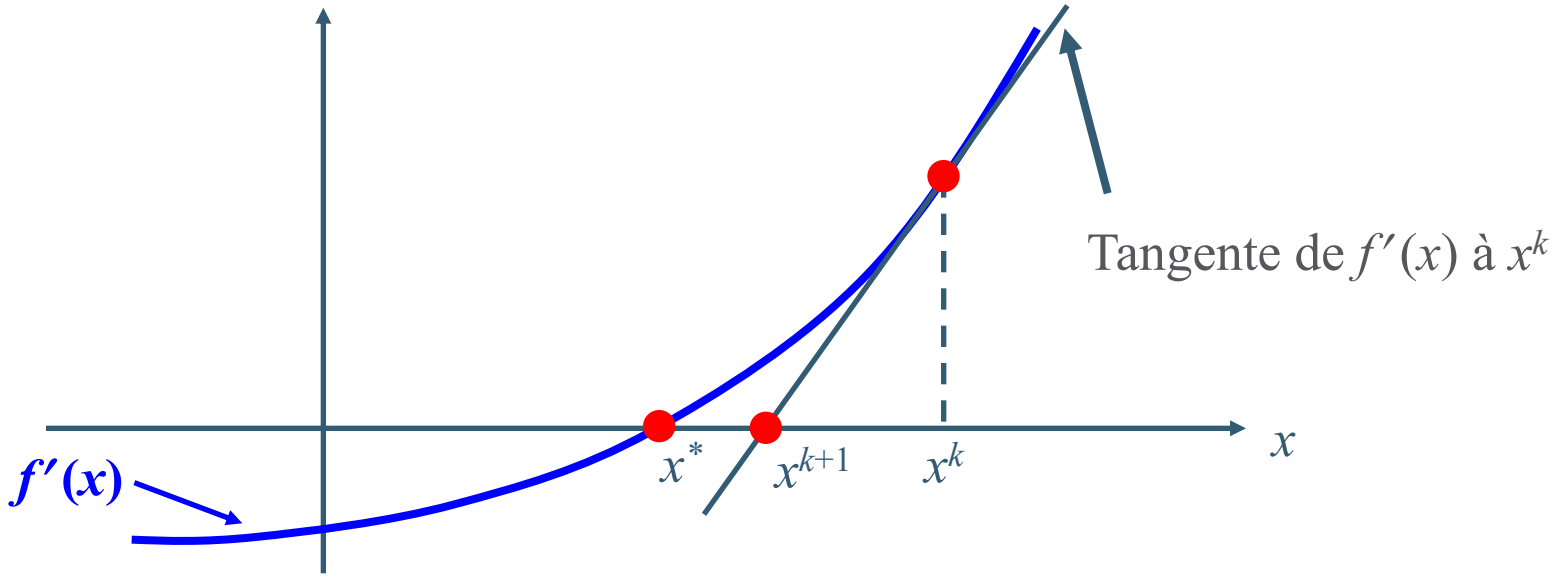
Lors de la résolution de $f'(\mathrm{x}) = 0$ pour trouver un maximum ou un minimum, on peut utiliser les itérations suivantes:

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

- k est l'itération courante
- $|x^{k+1} - x^k| < e$ où e est une tolérance spécifiée

Diagramme de la méthode de Newton

La méthode de Newton approxime $f'(x)$ avec une droite au point x^k et on obtient le point (x^{k+1}), qui est utilisé pour approximer la fonction au prochain point. Jusqu'à ce qu'on rapproche suffisamment de x^*.



Commentaire sur la méthode de Newton

Il faut s'assurer que

- $f(x^{k+1}) < f(x^k)$ pour trouver un minimum.
- $f(x^{k+1}) > f(x^k)$ pour trouver un maximum.

Désavantage:

- Il faut calculer les dérivées premières et secondes
- La solution initiale est importante, si elle est trop loin de l'optimum la méthode peut ne pas converger.

Méthode de Regula-Falsi

Cette méthode nécessite deux points x^a et x^b qui encadrent la solution de l'équation $f'(x) = 0$.

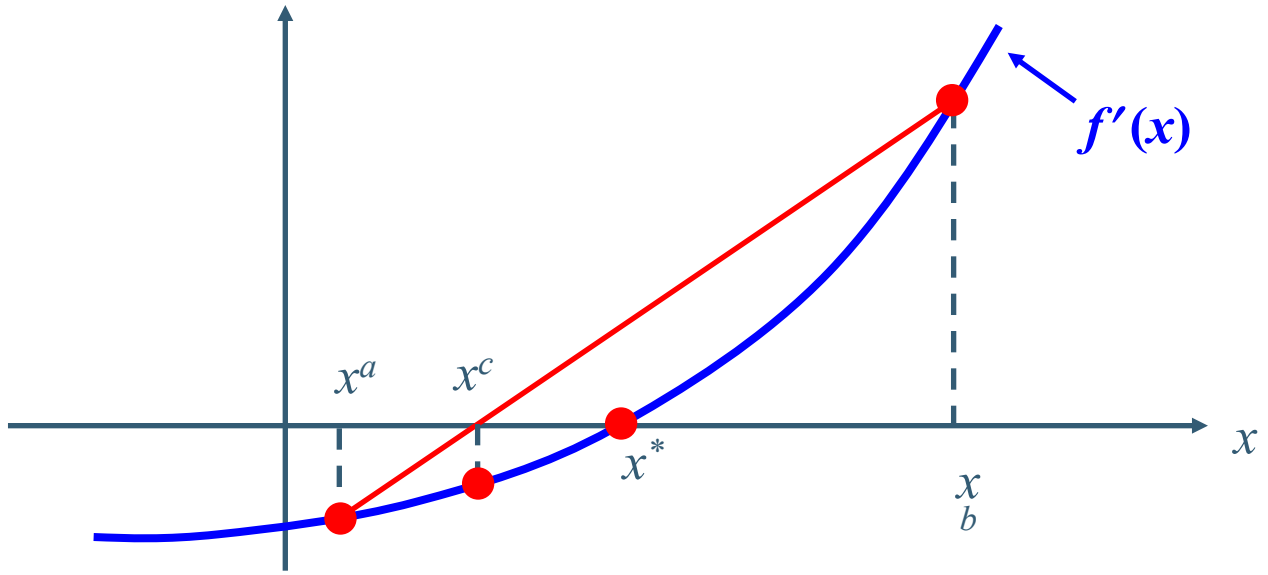
$$x^c = x^b - \frac{f'(x^b) \cdot (x^b - x^a)}{f'(x^b) - f'(x^a)}$$

où x^c sera entre x^a et x^b.

Le prochain interval sera défini par x^c et soit x^a ou x^b, celui dont le signe sera différent de x^c.

Diagramme de Regula-Falsi

La méthode de Regula-Falsi approxime fonction $f'(x)$ avec une droite et utilise l'interpolation pour trouver la racine.



Commentaires sur Regula-Falsi

- Cette méthode nécessite la connaissance de deux points qui entourent la solution.
- Elle ne nécessite toutefois pas le calcul de la dérivée seconde.
- La méthode de Regula-Falsi nécessite toutefois un peu plus d'itération que la méthode de Newton.

Optimisation multi variables

Considérons le cas avec plusieurs variables, sans contrainte.

Pratiquement toutes les méthodes suivent la recette suivante:

1. Choisir une direction de recherche $\mathbf{d}^k$
2. Chercher le minimum dans cette direction afin de trouver le prochain point.

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha^k \mathbf{d}^k$$

où k est le numéro d'itération courante et α^k est un scalaire positif appeler le pas

Le pas

Le pas, α^k, est calculé de la manière suivante :

- Nous voulons optimiser la fonction $f(\mathbf{x}^{k+1}) = f(\mathbf{x}^k + \alpha^k \mathbf{d}^k)$ où la seule variable est α^k parce que $\mathbf{x}^k$ et $\mathbf{d}^k$ sont connus.

Nous posons $\dfrac{\mathrm{d}f\left(\mathbf{x}^k + \alpha^k \mathbf{d}^k\right)}{\mathrm{d}\alpha^k} = 0$ et résolvons pour α^k en utilisant la méthode à une seule variable vue précédemment.

Méthode de la descente la plus rapide

Cette méthode est très simple

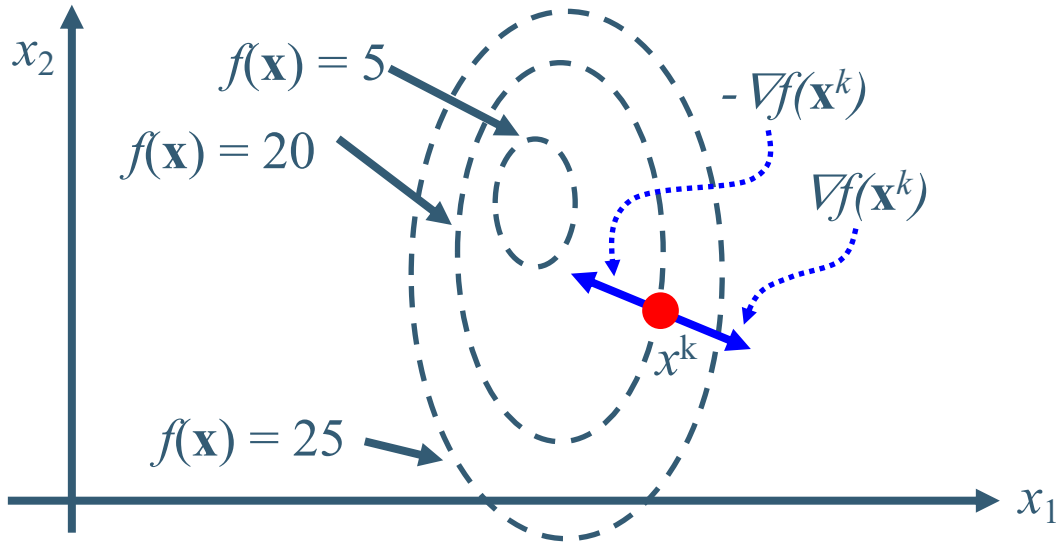
- Pour donner une direction, elle utilise le gradient (pour maximiser) ou le gradient négatif (pour minimiser)

$$\mathbf{d}^k = \begin{Bmatrix} + \\ - \end{Bmatrix} \nabla f(\mathbf{x}^k) \qquad \text{pour} \qquad \begin{Bmatrix} \max \\ \min \end{Bmatrix}$$

$$\mathbf{x}^{k+1} = \mathbf{x}^k \begin{Bmatrix} + \\ - \end{Bmatrix} \alpha^k \nabla f(\mathbf{x}^k)$$

Méthode de la descente la plus rapide

L'opposé du gradient nous indique la direction de descente la plus rapide.



Méthode de la descente la plus rapide

Les étapes sont :

1. Choisir un point initial $\mathbf{x}^0$
2. Calculer le gradient $\nabla f(\mathbf{x}^k)$ où k est un numéro d'iteration
3. Calculer la direction de recherche : $\mathbf{d}^k = \pm\nabla f(\mathbf{x}^k)$
4. Calculer le prochain point $\mathbf{x}$: $\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha^k \mathbf{d}^k$

   en utilisant une méthode d'optimisation à une variable sur α^k.

Méthode de la descente la plus rapide

Pour déterminer la convergence, utiliser une tolérance ε_1 et arrêter: $\left|f(\mathbf{x}^{k+1}) - f(\mathbf{x}^k)\right| < \varepsilon_1$

Où utiliser une tolérance ε_2 et arrêter: $\left\|\nabla f(\mathbf{x}^k)\right\| < \varepsilon_2$

Deux critères d'arrêt sont valides pour la majorité des méthodes d'optimisation à plusieurs variables.

Rappelons que la norme d'un vecteur, $\|\mathbf{x}\|$ est donné par:

$$\|\mathbf{x}\| = \sqrt{\mathbf{x}^T \cdot \mathbf{x}} = \sqrt{(x_1)^2 + (x_2)^2 + \cdots + (x_n)^2}$$

Exemple

Minimiser $f(x_1, x_2, x_3) = (x_1)^2 + x_1(1 - x_2) + (x_2)^2 - x_2x_3 + (x_3)^2 + x_3$

Avec le point de départ $\mathbf{x}^0 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$

Exemple

$$\nabla f(\mathbf{x}) = [2x_1 + (1 - x_2) \quad -x_1 + 2x_2 - x_3 \quad -x_2 + 2x_3 + 1]$$

$$\mathbf{d}^0 = -\nabla f(\mathbf{x}^0) = -[2(0) + 1 - 0 \quad -0 + 0 - 0 \quad -0 + 0 + 1]$$

$$= -[1 \quad 0 \quad 1] = [-1 \quad 0 \quad -1]$$

$$\mathbf{x}^1 = [0 \quad 0 \quad 0] + \alpha^0 \cdot [-1 \quad 0 \quad -1]$$

Il faut maintenant trouver α^0

Exemple

$$f(\mathbf{x}^1) = (\alpha^0)^2 + (-\alpha^0)(1) + 0 - 0 + (\alpha^0)^2 + (-\alpha^0)$$

$$= 2(\alpha^0)^2 - 2(\alpha^0)$$

$$\frac{df(\mathbf{x}^1)}{d\alpha^0} = 4(\alpha^0) - 2$$

On met = 0 et on résout :

$$4(\alpha^0) = 2 \quad \Rightarrow \quad \alpha^0 = {}^2/_4 = {}^1/_2$$

Exemple

$$\mathbf{x}^1 = [0 \quad 0 \quad 0] + \alpha^0 \cdot [-1 \quad 0 \quad -1]$$

$$= [0 \quad 0 \quad 0] + \left[-\frac{1}{2} \quad 0 \quad -\frac{1}{2}\right]$$

$$\therefore \mathbf{x}^1 = \left[-\frac{1}{2} \quad 0 \quad -\frac{1}{2}\right]$$