Algorithmes de descente de gradient pour les problèmes d'optimisation différentiables

Guillaume Lecué[1]

Résumé

L'objectif de ce chapitre est de présenter des algorithmes de descentes pour la résolution approximative de problèmes d'optimisation différentiables avec ou sans contraintes. On verra en particulier que dans ce cadre, l'opposé du gradient est une direction de descente qui apparaît naturellement. On fera l'étude de ces algorithmes sous certaines conditions portant sur la convexité de f et la régularité de son gradient ou de sa Hessienne lorsqu'elle existe.

Table des matières

1 Introduction : algorithmes de descentes

La plupart des algorithmes servant à approcher une solution à un problème d'optimisation (avec ou sans contrainte) sont itératifs. C'est-à-dire, l'algorithme fournit le début $x_0, x_1, \ldots, x_T$ d'une suite $(x_k)_k$ d'éléments de $\mathbb{R}^n$ qui, sous certaines hypothèses sur la fonction objectif f (et la contrainte K si on traite un problème sous contrainte), converge vers une solution du problème d'optimisation. La sortie x_T donnée par l'algorithme fournit alors une approximation (plus ou moins bonne) à une solution au problème $\min(f(x) : x \in \mathbb{R}^n)$ ou $\min(f(x) : x \in K)$.

En règle générale, le premier point x_0 est choisi par l'utilisateur, on l'appelle **l'initialisation**. Si on n'a pas trop d'a priori sur la solution du problème, on peut prendre x_0 de manière aléatoire

1. CREST, ENSAE. Bureau 3029. 5 avenue Henry Le Chatelier. 91120 Palaiseau. Email : guillaume.lecue@ensae.fr.

– par exemple, x_0 suit une Gaussienne $\mathcal{N}(0, I_n)$. Si on pense que la solution du problème a beaucoup de coordonnées à 0 (cas des vecteurs sparses), on peut prendre $x_0 = 0$. Parfois, trouver une bonne initialisation, appelé **warm start**, est crucial pour la convergence de l'algorithme vers une solution du problème. En particulier, pour les problèmes ayant des minima locaux qui ne sont pas tous minima globaux ; c'est souvent le cas pour les problèmes d'optimisations non-convexes.

L'indice T d'arrêt de l'algorithme, qui revoit donc x_T en sortie comme une approximation de la solution du problème, est décidé à partir d'une **stopping rule** décidée en amont. La stopping rule la plus simple étant de choisir a priori une certaine valeur pour T, par exemple $T = 1000$; ce qui signifie qu'on sort de l'algorithme à la millième itération et qu'on renvoi la dernière itération obtenue – ici, x_{1000}. Il existe une multitude de critères d'arrêt. On en verra quelques uns dans ce chapitre.

Il existe une multitude d'algorithmes pour les problèmes d'optimisation, comme les algorithmes de descente, les algorithmes de points fixes, les algorithmes maximization/minimization (MM), algorithmes expectation/maximization (EM), algorithmes génétiques, algorithmes aléatoires, etc.. On étudiera uniquement les algorithmes de descente dans ce chapitre.

On dit qu'un algorithme est un **algorithme de descente** quand la valeur de la fonction objectif le long de l'algorithme ne fait que décroître : càd un algorithme produisant une suite $(x_k)_k$ telle que $(f(x_k))_k$ est une suite décroissante. Tout l'enjeu des algorithmes itératifs est de décider comment passer de $x_0, x_1, \cdots, x_k$ à x_{k+1} – et souvent même seulement le passage de x_k à x_{k+1} permet de déterminer l'algorithme.

Pour les algorithmes de descente que nous étudierons dans ce chapitre, le passage de x_k à x_{k+1} se résume à trouver une direction $p_k \in \mathbb{R}^N$, appelée **direction de descente**, le long de laquelle on va aller et un certain **pas de descente** ou **step size** $\eta_k > 0$ qui quantifie de combien on va descendre partant de x_k dans la direction p_k. On obtient alors x_{k+1} en partant de x_k et en avançant de η_k dans la direction p_k, qu'on écrit sous la forme $x_{k+1} = x_k + \eta_k p_k$. Il faut donc déterminer cette direction p_k et ce step size η_k à chaque étape de telle sorte qu'on se rapproche de plus an plus à chaque étape d'une solution à notre problème d'optimisation.

On peut formaliser la notion de direction de descente avec la définition suivante.

Définition 1.1. *Soit $f : \mathbb{R}^n \to \mathbb{R}$ et $x \in \mathbb{R}^n$. Le vecteur $p \in \mathbb{R}^n$ est appelé **direction de descente de f en** x quand il existe $\bar{\alpha} > 0$ tel que pour tout $0 < \alpha \leq \bar{\alpha}$,*

$$f(x + \alpha p) < f(x).$$

Cette définition permet de définir d'une manière très générale n'importe quel algorithme de descente (on ne précise pas notre politique de critère d'arrêt dans l'Algorithme 1 ci-dessous).

input : x_0 : un point initial
output: une approximation x_T de x^* solution de $\min_{x \in \mathbb{R}^n} f(x)$

while *critère d'arrêt non satisfait* **do**
  On spécifie p_k une *direction de descente de f en x_k*
  On détermine un *step size* $\alpha_k > 0$ tel que $f(x_k + \alpha_k p_k) < f(x_k)$
  On pose $x_{k+1} = x_k + \alpha_k p_k$
end

Algorithm 1: Forme générale des algorithmes de descente.

Le but de ce chapitre est de proposer des directions de descente et des choix de step size qui permettent à la suite des itérés $(x_k)_k$ d'un algorithme de converger vers une solution au problème

d'optimisation sans contrainte $\min_{x\in\mathbb{R}^n} f(x)$ ainsi qu'au problème d'optimisation sous contraintes $\min_{x\in K} f(x)$ ou au moins que d'assurer que $(f(x_k))_k$ tends vers $\min(f(x) : x \in \mathbb{R}^n)$, en particulier, quand on n'a pas forcément unicité d'une solution à ce problème. Il est aussi important d'avoir des garanties de convergence de ces algorithmes pour pouvoir ainsi assurer que le vecteur obtenu en sortie de notre algorithme permet bien d'approcher une solution de notre problème à un niveau d'approximation donné a priori. On étudiera donc les propriété de convergence des algorithmes de descentes que nous introduirons. On peut déjà donner un premier résultat de convergence (très général) sur les algorithmes de descente comme décrit dans Algorithme 1.

Théorème 1.2. *Soit f une fonction de classe $\mathcal{C}^1$. Soit $(x_k)_k$ la suite des itérés de l'Algorithme 1. On suppose qu'il existe des constantes $\theta_1 > 0$ et $\theta_2 > 0$ telles que*

a)(condition d'angle) $\langle -\nabla f(x_k), p_k\rangle \geq \theta_1 \|\nabla f(x_k)\|_2 \|p_k\|$

b)(décroissance suffisante)

$$f(x_k + \alpha_k p_k) \leq f(x_k) - \theta_2 \left(\frac{\langle \nabla f(x_k), p_k\rangle}{\|p_k\|}\right)^2.$$

Alors tout point d'accumulation de $(x_k)_k$ est un point critique de f.

Preuve. Comme le step size satisfait la condition de décroissance suffisante, on a

$$f(x_{k+1}) \leq f(x_k) - \theta_2 \left(\frac{\langle \nabla f(x_k), p_k\rangle}{\|p_k\|}\right)^2.$$

Par ailleurs, la condition d'angle donne

$$-\theta_2 \left(\frac{\langle \nabla f(x_k), p_k\rangle}{\|p_k\|}\right)^2 \leq -\theta_1^2\theta_2 \|\nabla f(x_k)\|_2^2.$$

On obtient alors que $f(x_{k+1}) \leq f(x_k) - \theta_1^2\theta_2 \|\nabla f(x_k)\|_2^2$ et donc la suite $(f(x_k))_k$ décroît (c'est donc bien une algorithme de descente).

On note par x^* un point d'accumulation de $(x_k)_k$. Par continuité de f, $(f(x_{\varphi_k}))_k$ tend vers $f(x^*)$ pour une certaine sous-suite $(\varphi_k)_k$. Mais par monotonicité de $(f(x_k))_k$, on a que toute la suite $(f(x_k))_k$ converge vers $f(x^*)$. En particulier,

$$f(x_k) - f(x_{k+1}) \to 0$$

quand $k \to \infty$. Comme $f(x_k) - f(x_{k+1}) \geq \theta_1^2\theta_2 \|\nabla f(x_k)\|_2^2$ pour tout $k \in \mathbb{N}$, on a que $\|\nabla f(x_k)\|_2 \to 0$ quand $k \to \infty$. Alors par continuité du gradient, on a bien $\nabla f(x^*) = 0$. Donc $(x_k)_k$ converge vers un point critique de f. ∎

Dans ce premier théorème, on voit apparaître deux conditions sur le choix de la direction de descente (condition *a)*) et sur le choix du step size (condition *b)*). La condition *a)* portant sur le choix de p_k dit qu'on doit choisir une direction de descente positivement corrélée avec l'**opposé du gradient** de f en x_k, en particulier, $p_k = -\nabla f(x_k)$ satisfait cette condition. On verra dans la suite de ce chapitre que cette direction est particulièrement à privilégiée et que l'interprétation géométrique du gradient en fonction des lignes de niveau de f permet de comprendre ce choix.

La conclusion du Théorème 1.2 est aussi intéressante : elle assure la convergence de l'algorithme vers un point critique de f et non vers une solution au problème $\min(f(x) : x \in \mathbb{R}^n)$. On sait qu'une solution à ce problème est un point critique mais – sans la convexité de f – on n'a pas la

réciproque en général. C'est pour cette raison que la notion de convexité est si importante pour l'étude de la convergence des algorithmes de descente car elle assure que les points critiques sont aussi solution de notre problème d'optimisation. Il existe cependant des preuves de convergence d'algorithme de descente qui se passent de la convexité par exemple quand on a un warm start assurant une certaine proximité entre l'initialisation de l'algorithme (càd x_0) et une solution au problème ou quand on peut montrer que tous les points critiques d'une fonction sont proches de son minimum global. On ne traitera pas ce genre de preuve dans ce chapitre et on supposera plus simplement la convexité de f qui est un cas assez fréquent en pratique.

Il est assez fréquent de voir certains algorithmes de descente comme des algorithmes de point fixes. L'avantage de ce point de vue est qu'il permet de comprendre les raisons de la convergence de l'algoritgme. En effet, dans cette approche, on essaie de trouver une fonction F vérifiant la propriété "x^* est solution du problème si et seulement si $x^* = F(x^*)$" et $x_{k+1} = F(x_k)$. Dans ce cas, l'étude de la convergence de la suite $(x_k)_k$ se résume à prouver que la fonction F est contractante, càd $\|F(x) - F(y)\|_2 \leq \gamma \|x - y\|_2$ pour un certain $0 \leq \gamma < 1$. Il y a de nombreux exemple d'algorithmes de descente qui peuvent se voir comme des algorithmes de points fixe. Dans ce cas l'analyse de leur convergence se résume à prouver une propriété de contraction.

2 Algorithmes pour les problèmes d'optimisation sans contraintes

Dans cette section, on s'interesse à la construction d'algorithmes de descente pour la résolution approximative du problème d'optimisation sans contrainte $\min_{x\in\mathbb{R}^n} f(x)$. C'est pour ce problème que les algorithmes de Newton et de descente de gradient sont introduits. Ce sont probablement ces algorithmes et leurs variants qui sont les plus utilisés à l'heure actuelle.

2.1 Méthode de Newton pour résoudre "$F(x) = 0, x \in \mathbb{R}$"

Soit $F : \mathbb{R} \to \mathbb{R}$ une fonction différentiable sur $\mathbb{R}$. On souhaite trouver $x^* \in \mathbb{R}$ tel que $F(x^*) = 0$. La méthode de Newton consiste à approcher la fonction F par son développement du premier ordre de Taylor (en chaque point de l'itération) : en tout point $x \in \mathbb{R}$ le développement de Taylor d'ordre 1 est donné par : quand $p \to 0$,

$$F(x + p) = F(x) + F'(x)p + o(p).$$

Ainsi, si on choisit p^* tel que

$$F(x) + F'(x)p^* = 0 \tag{2.1}$$

on aura $F(x + p^*) = o(p^*)$ et donc si p^* est petit, on aura $F(x + p^*)$ qui sera proche de zéro, et donc $x + p^*$ sera une solution approchante au problème qu'on souhaite résoudre ici.

L'algorithme de Newton pour résoudre "$F(x) = 0$" est un algorithme itératif qui part d'un point initial donné x_0 (qu'on choisit) et s'arrête lorsqu'un critère d'arrêt (décidé en amont) est rencontré (par exemple, lorsque deux points successifs x_{k+1} et x_k sont proches, indiquant une stagnation de l'algorithme) :

input : x_0 : un point initial ; $\epsilon > 0$: un paramètre de critère d'arrêt
output: une approximation x_k de x^* tel que $F(x^*) = 0$

```
while |x_{k+1} − x_k| ≥ ε do
    Calcul de p_k solution de F(x_k) + F'(x_k)p_k = 0
    x_{k+1} = x_k + p_k
end
```

Algorithm 2: Algorithme de Newton pour le problème $F(x) = 0$.

On voit ici que le choix du vecteur p_k au point x_k est donné par l'équation (2.1) au point $x = x_k$ qui consiste à annuler le développement du premier ordre de F en x_k. Quand $F'(x_k) \neq 0$ alors la direction de descente est donnée par $p_k = -F(x_k)/F'(x_k)$ qui est la forme classique de l'algorithme de Newton :

$$x_{k+1} = x_k - \frac{F(x_k)}{F'(x_k)}. \tag{2.2}$$

Quand $F'(x_k) = 0$, x_k est un point critique de F et il est alors difficile avec seulement un DL1 de F de savoir dans quelle direction aller pour trouver un zéro de F. Il faut dans ce cas faire une DL2 si F est $\mathcal{C}^2$ ou bien bouger un peu autour de x_k en lui ajoutant un bruit par exemple aléatoire.

La formule (2.2) est assez simple mais elle est en fait fondamentale dans de nombreux algorithmes utilisé aujourd'hui. Elle est aussi au coeur de nombreuse recherche théorique en math : on peut voir la participation de Stephen Smale au Concinnitas project et la discussion qui suit sur cette formule.

Algorithme de Newton sur un exemple. On considère la fonction $F : x \in \mathbb{R} \to x^2 - 2$. L'équation $F(x) = 0$ a deux solutions données par $-\sqrt{2}$ et $\sqrt{2}$. On va lancer l'algorithme de Newton sur cet exemple pour voir comment il évolue. Le passage de x_k à x_{k+1} se fait avec

$$p_k = \frac{-F(x_k)}{F'(x_k)} = \frac{2 - x_k^2}{2x_k}$$

quand $x_k \neq 0$ et donc

$$x_{k+1} = x_k + \frac{2 - x_k^2}{2x_k} = \frac{1}{x_k} + \frac{x_k}{2} = g(x_k) \text{ où } g : x \neq 0 \to \frac{1}{x} + \frac{x}{2}.$$

L'algorithme de Newton produit une suite récurrente d'ordre 1 avec g pour fonction de de lien. C'est aussi un algorithme de point fixe et donc les propriétés de contraction de g vont joué un rôle dans l'étude de sa convergence.

On observe que pour tout $t > 0$, $g(t) \geq \sqrt{2}$. Ainsi, pour une initialisation $x_0 > 0$, on aura pour tout $k \geq 1$, $x_k \geq \sqrt{2}$. Or, $g'(t) = -1/t^2 + 1/2$ pour tout $t > 0$ donc $\sup_{t \geq \sqrt{2}} |g'(t)| = 1/2$ et donc g est contractante sur $[\sqrt{2}, +\infty)$. On a alors pour tout $k \geq 2$ que x_k et x_{k-1} sont dans $[\sqrt{2}, +\infty)$ et donc

$$|x_{k+1} - x_k| = |g(x_k) - g(x_{k-1})| \leq \sup_{t \geq \sqrt{2}} |g'(t)||x_k - x_{k-1}| \leq \left(\frac{1}{2}\right) |x_k - x_{k-1}|.$$

On a alors pour tout $p > q$ que

$$|x_p - x_q| \leq \sum_{k=q}^{p-1} |x_{k+1} - x_k| \leq \sum_{k=q}^{p-1} \left(\frac{1}{2}\right)^k |x_{q+1} - x_q| \leq 2|x_{q+1} - x_q| \leq 2\left(\frac{1}{2}\right)^{q-1} |x_2 - x_1|.$$

On en déduit que $|x_p - x_q| \to 0$ quand $p > q \to +\infty$ donc $(x_k)_k$ est une suite de Cauchy et donc elle converge. On note $x^* \in \mathbb{R}$ tel que $x_k \to x^*$, comme $x_k \geq \sqrt{2}$ pour tout $k \geq 1$, on a aussi $x^* \geq \sqrt{2}$ et comme g est continue sur $\mathbb{R}_*^+$, on en déduit que $g(x_k) \to g(x^*)$. Comme $x_{k+1} = g(x_k)$, on a en passant à la limite que $x^* = g(x^*)$ et donc vu que $x^* \geq \sqrt{2}$, on a $x^* = \sqrt{2}$. On a donc montrer que si $x_0 > 0$ alors $x_k \to \sqrt{2}$. Par symétrie, on a que si $x_0 < 0$ alors $x_k \to -\sqrt{2}$. Pour $x_0 = 0$, l'algorithme de Newton n'est pas défini.

2.2 Algorithmes de Newton et de descente de gradient dans $\mathbb{R}$

Soit $f : \mathbb{R} \to \mathbb{R}$ une fonction convexe deux fois différentiable. On souhaite trouver une solution au problème $\min_{x\in\mathbb{R}} f(x)$.

Par convexité et différentiabilité de f, résoudre ce problème est équivalent à résoudre le problème $f'(x) = 0$. On est donc amené à résoudre un problème du type $F(x) = 0$ pour $F = f'$ qu'on peut résoudre de manière approchée par la méthode de Newton vue dans la section précédente.

Pour le problème de minimisation $\min_{x\in\mathbb{R}} f(x)$ d'une fonction convexe, la méthode de Newton consiste à minimiser une approximation de Taylor du second ordre de f : pour tout $x \in \mathbb{R}$, quand $p \to 0$,

$$f(x+p) = f(x) + f'(x)p + \frac{p^2 f''(x)}{2} + o(p^2).$$

Minimiser la fonction convexe $p \to f(x) + f'(x)p + (1/2)p^2 f''(x)$ revient à trouver un zéro de son gradient, càd à trouver p^* tel que

$$f'(x) + f''(x)p^* = 0$$

qui est exactement la même équation que celle définissant la direction de descente pour la méthode de Newton dans (2.1) pour $F = f'$.

input : $x_0 \in \mathbb{R}$: un point initial ; $\epsilon > 0$: un paramètre de critère d'arrêt
output: une approximation $x_k \in \mathbb{R}$ de x^* tel que $f(x^*) = \min(f(x) : x \in \mathbb{R}^n)$

1 **while** $|x_{k+1} - x_k| \geq \epsilon$ **do**
2 Calcul de p_k, une **direction de descente**, solution de $f'(x_k) + f''(x_k)p_k = 0$
3 $x_{k+1} = x_k + p_k$
4 **end**

Algorithm 3: Algorithme de Newton pour le problème $\min_{x\in\mathbb{R}} f(x)$.

Ainsi quand $f''(x_k) \neq 0$, la direction de descente est donnée par $p_k = -f'(x_k)/f''(x_k)$ et donc l'algorithme de Newton pour la résolution approximative du problème $\min(f(x) : x \in \mathbb{R})$ pour f convexe et $\mathcal{C}^2$ est donné par

$$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}. \tag{2.3}$$

Dans certain cas, il est coûteux, voire impossible de calculer $f''(x_k)$. On a donc recours à une suite donnée a priori, ou calculée au fil des itérations, qu'on appelle les **steps size** : $(\eta_k)_k$ où $\eta_k > 0$. La direction de descente est obtenue à l'étape k comme étant solution de l'équation

$$f'(x) + \frac{p_k}{\eta_k} = 0.$$

On a donc "remplacer" la quantité difficile à obtenir $f''(x_k)$ par $1/\eta_k$. La direction de descente ainsi obtenue est $p_k = -\eta_k f'(x_k)$ et l'algorithme associé est

$$x_{k+1} = x_k - \eta_k f'(x_k) \tag{2.4}$$

qui est l'algorithme de **descente de gradient**. Pour cet algorithme, on a seulement besoin que f soit différentiable (pas besoin d'existence d'une dérivée seconde). L'algorithme de Newton (2.3) est aussi un algorithme de descente de gradient pour le step size $\eta_k = 1/f''(x_k)$. Il est important de noter que pour ces deux algorithmes (2.3) et (2.4), la direction de descente qui est apparue naturellement est donnée par l'opposé du gradient de f en x_k. La direction $-f'(x_k)$ est donc celle qu'on utilisera le plus souvent et le plus naturellement.

2.3 Algorithme de Newton pour des fonctions de plusieurs variables

Soit $f : \mathbb{R}^n \to \mathbb{R}$ une fonction convexe deux fois différentiable. On souhaite résoudre le problème d'optimisation $\min_{x\in\mathbb{R}^n} f(x)$.

La méthode de Newton consiste à approcher f au point courant x_k par son développement d'ordre 2 de Taylor : quand $p \to 0$,

$$f(x_k + p) = f(x_k) + \langle \nabla f(x_k), p\rangle + \frac{1}{2} p^\top \nabla^2 f(x_k) p + o(\|p\|_2^2)$$

et à minimiser en $p \in \mathbb{R}^n$ cette approximation $p \to f(x_k) + \langle \nabla f(x_k), p\rangle + \frac{1}{2} p^\top \nabla^2 f(x_k) p$. L'approximation étant convexe et différentiable, cela revient à annuler son gradient. On obtient donc une direction de descente p_k solution de l'équation :

$$\nabla f(x_k) + \nabla^2 f(x_k) p_k = 0 \tag{2.5}$$

où $\nabla f(x_k) = (\partial_i f(x_k))_{i=1}^n$ est le **gradient** de f en x_k et $\nabla^2 f(x_k) = (\partial^2_{ij} f(x_k))_{1\le i,j\le n}$ est la matrice **Hessienne** de f en x_k.

Ainsi, quand $\nabla^2 f(x_k)$ est inversible, la direction de descente est donnée par

$$p_k = -\big(\nabla^2 f(x_k)\big)^{-1} \nabla f(x_k)$$

et l'algorithme de Newton associé est décrit dans Algorithme 4.

input : $x_0 \in \mathbb{R}^n$: un point initial ; $\epsilon > 0$: un paramètre de critère d'arrêt
output: une approximation $x_k \in \mathbb{R}^n$ de x^* tel que $f(x^*) = \min_{x\in\mathbb{R}^n} f(x)$

```
while ‖x_{k+1} − x_k‖_2 ≥ ε do
    x_{k+1} = x_k − (∇²f(x_k))^{-1} ∇f(x_k)
end
```

Algorithm 4: Algorithme de Newton pour la construction d'une solution approchante au problème $\min_{x\in\mathbb{R}^n} f(x)$.

L'algorithme de Newton converge très rapidement vers la solution du problème d'optimisation $\min_{x\in\mathbb{R}^n} f(x)$ quand $x \to \nabla^2 f(x)$ est Lipschitz et $\nabla^2 f(x)$ a une plus petite valeur singulière plus grande que c uniformément en x. On rappelle la définition de la norme d'opérateur d'une matrice :

$$\|M\| = \max_{\|x\|_2=1} \|Mx\|_2 .$$

C'est la norme de M vue comme opérateur de $\ell_2^n \mapsto \ell_2^n$. En particulier, on a pour tout $x \in \mathbb{R}^n$, $\|Mx\|_2 \le \|M\|\,\|x\|_2$.

Théorème 2.1. *Soit $f : \mathbb{R}^n \to \mathbb{R}$ une fonction convexe deux fois différentiable. On suppose qu'il existe $x^* \in \mathbb{R}^n$ tel que $f(x^*) = \min_{x\in\mathbb{R}^n} f(x)$. On suppose que :*

i) $\nabla^2 f$ est L-Lipschitz pour la norme d'opérateur : pour tout $x, y \in \mathbb{R}^n$,

$$\left\|\nabla^2 f(x) - \nabla^2 f(y)\right\| \leq L \|x - y\|_2$$

ii) pour tout $x \in \mathbb{R}^n$, $\nabla^2 f(x) \succeq cI_n$ (càd $\langle \nabla^2 f(x)y, y\rangle \geq c\|y\|_2^2$ pour tout $y \in \mathbb{R}^n$).

Soit $x_0 \in \mathbb{R}^n$. Alors, l'algorithme de Newton $(x_k)_k$ défini par x_0 et $x_{k+1} = x_k - \big(\nabla^2 f(x_k)\big)^{-1}\nabla f(x_k)$ vérifie $\|x_{k+1} - x^*\|_2 \leq [L/(2c)]\|x_k - x^*\|_2^2$ pour tout $k \in \mathbb{N}$.

Démonstration. On a $\nabla f(x^*) = 0$, alors d'après le théorème fondamental de l'analyse,

$$\nabla f(x_k) = \nabla f(x_k) - \nabla f(x^*) = \int_0^1 \nabla^2 f(x^* + \tau(x_k - x^*))(x_k - x^*)d\tau.$$

On a donc

$$\begin{aligned} x_{k+1} - x^* &= x_k - x^* - \big(\nabla^2 f(x_k)\big)^{-1}\nabla f(x_k) \\ &= x_k - x^* - \big(\nabla^2 f(x_k)\big)^{-1}\int_0^1 \nabla^2 f\big(x^* + \tau(x_k - x^*)\big)(x_k - x^*)d\tau \\ &= \big(\nabla^2 f(x_k)\big)^{-1}\int_0^1 \big[\nabla^2 f(x_k) - \nabla^2 f\big(x^* + \tau(x_k - x^*)\big)\big](x_k - x^*)d\tau. \end{aligned}$$

On en déduit que

$$\|x_{k+1} - x^*\|_2 \leq \left\|\nabla^2 f(x_k)^{-1}\right\| \left\|\int_0^1 \big[\nabla^2 f(x_k) - \nabla^2 f\big(x^* + \tau(x_k - x^*)\big)\big]d\tau\right\| \|x_k - x^*\|_2 .$$

On conclut en observant que $\left\|\nabla^2 f(x_k)^{-1}\right\| = \left\|\nabla^2 f(x_k)\right\|^{-1} \leq (1/c)$ et

$$\begin{aligned} &\left\|\int_0^1 \big[\nabla^2 f(x_k) - \nabla^2 f\big(x^* + \tau(x_k - x^*)\big)\big]d\tau\right\| \leq \int_0^1 \left\|\nabla^2 f(x_k) - \nabla^2 f\big(x^* + \tau(x_k - x^*)\big)\right\| d\tau \\ &\leq \int_0^1 L(1-\tau)\|x_k - x^*\|_2 \, d\tau = (L/2)\|x_k - x^*\|_2 . \end{aligned}$$

■

En particulier, sous les hypothèses du Théorème 2.1, on a

$$\|x_k - x^*\|_2 \leq \left(\frac{L}{2c}\right)^{2^{k+1}-1}\|x_0 - x^*\|_2^{2^k} = \frac{2c}{L}\left(\frac{L^2\|x_0 - x^*\|_2}{(2c)^2}\right)^{2^k}.$$

Donc si $L^2\|x_0 - x^*\|_2 < (2c)^2$ alors $(x_k)_k$ converge très rapidement vers x^*.

La condition ii) est en particulier vérifiée pour les fonctions deux fois différentiables et c-**convexe**, càd qui vérifient pour tout $x, y \in \mathbb{R}^n$ et $0 \leq t \leq 1$,

$$f\big(tx + (1-t)y\big) \leq tf(x) + (1-t)f(y) - \frac{ct(1-t)}{2}\|x - y\|_2^2 \tag{2.6}$$

On parle aussi de fonction **fortement convexe** quand il existe c tel que f est c-convexe. On retrouvera cette propriété plus tard pour l'étude de la convergence d'autres algorithmes de descente.

2.4 Algorithmes de descente de gradient dans $\mathbb{R}^n$

Pour la construction de l'algorithme de Newton (voir Algorithme 4), on a besoin que f soit de classe $\mathcal{C}^2$ et de calculer sa Hessienne et de l'inverser. C'est potentiellement très coûteux et parfois impossible si on n'a pas une fonction de classe $\mathcal{C}^2$. Dans ce cas, on utilise un algorithme de descente de gradient : on prends $-\nabla f(x_k)$ comme direction de descente de f partant de x_k et on choisit un step size η_k, soit a priori, soit de manière itérative (en se fixant au préalable une certaine politique de choix du step size). L'algorithme de descente de gradient s'écrit alors :

$$x_{k+1} = x_k - \eta_k \nabla f(x_k). \tag{2.7}$$

C'est l'algorithme le plus important de ce cours.

L'opposé du gradient est donc choisi comme direction de descente dans (2.7). On appelle parfois $-\nabla f(x_k)$ la **steepest descent direction** – direction de descente de plus forte pente. Le choix de prendre l'opposé du gradient comme direction de descente peut s'expliquer géométriquement comme nous l'avons déjà fait dans les chapitres précédent pour avoir une intuition géométrique sur le théorème de KKT. On rappelle ici cette représentation géométrique du gradient qui nous aide cette fois à comprendre ce choix de direction de descente.

2.4.1 Interprétation géométrique du gradient

Géométriquement, le gradient d'une fonction s'interprète assez facilement. On peut en effet voir que le gradient $\nabla f(x)$ est un vecteur normal à la ligne de niveau $f(x)$ de f et que de plus il pointe vers la direction de plus forte pente de f partant de x. Il est donc assez naturel de prendre $-\nabla f(x)$ comme direction de descente quand on cherche à minimiser f. On peut formaliser ces propriétés de la manière suivante.

Pour cela, on introduit quelques notions de géométrie. On commence par rappeler les notions de lignes et ensemble de niveau d'une fonction.

Définition 2.2. *Soit $f : U \to \mathbb{R}$. Soit $\alpha \in \mathbb{R}$. La **courbe de niveau** α **de** f est*

$$\mathcal{L}_f(\alpha) = \{x \in U : f(x) = \alpha\}.$$

***L'ensemble de niveau** α **de** f est $\mathcal{L}_f(\leq \alpha) = \{x \in U : f(x) \leq \alpha\}$.*

Les courbes de niveaux d'une fonction qu'on cherche à minimiser (càd une fonction objectif) sont des bons "repères géométriques" car c'est le long de ces surfaces que le critère a minimiser reste constant. Pour bien se représenter le gradient, il faut d'abord bien se représenter une fonction. En optimisation, on préférera représenter visuellement une fonction $f : \mathbb{R}^2 \to \mathbb{R}$ par plusieurs de ces lignes de niveau dans $\mathbb{R}^2$ plutôt que par un graphique en $3D$ dans $\mathbb{R}^3$. C'est pour cette représentation visuelle de f qu'on pourra bien représenter sa fonction gradient ∇f.

On donne maintenant quelques outils qui permettent de représenter le gradient d'une fonction en fonction de ces lignes de niveau.

Définition 2.3. *Soit S un sous-ensemble de $\mathbb{R}^n$ non vide. Soit $x \in S$ et $v \in \mathbb{R}^p$. On dit que v est un **vecteur tangent à** S **en** x quand il existe deux suites $(\lambda_k)_k \subset \mathbb{R}_+^*$ et $(v_k)_k \subset \mathbb{R}^n$ telles que $(\lambda_k) \downarrow 0$, $x + \lambda_k v_k \in S$ et $v = \lim_k v_k$.*

*On dit qu'un vecteur $v^\perp$ est **orthogonal à** S **en** x quand pour tout vecteur v tangent à S en x on a $\langle v, v^\perp \rangle = 0$. On dit qu'un vecteur v^* est **normal à** S **en** x quand pour tout vecteur v tangent à S en x on a $\langle v, v^* \rangle \leq 0$.*

Par exemple quand $S = \mathbb{R}^p$ tous les éléments de $\mathbb{R}^p$ sont des vecteurs tangents à S en chacun de ces points. On renvoie au chapitre sur la différentiation d'une fonction à plusieurs variables pour plus d'exemples et d'intuition sur les vecteurs tangents à une surface. Ce qui nous intéresse pour le moment est d'étudier les vecteurs tangents aux lignes et ensembles de niveau d'une fonction différentiable et de montrer que le gradient donne un champs de vecteurs orthogonaux (resp. normaux) à ces lignes (resp. ensembles) de niveau. On peut en effet voir le gradient ∇f comme un champs de vecteur sur $\mathbb{R}^n$ car c'est une application de $\mathbb{R}^n$ dans lui-même et pour tout point $x \in \mathbb{R}^n$, $\nabla f(x)$ est un vecteur de $\mathbb{R}^n$ (on fait ici la distinction entre points et vecteurs de $\mathbb{R}^n$).

Proposition 2.4. *Soit $f : U \to \mathbb{R}$ où U est un ouvert de $\mathbb{R}^n$. Soit $x \in U$. On suppose que f est différentiable en x. Alors $\nabla f(x)$ est orthogonal à la courbe de niveau $f(x)$ de f en x, càd à $\mathcal{L}_f(f(x))$. Le gradient $\nabla f(x)$ est normal à l'ensemble de niveau $f(x)$ de f, càd à $\mathcal{L}_f(\leq f(x))$.*

Preuve. Soit v un vecteur tangent à $\mathcal{L}_f(f(x))$ en x. Montrons que $\langle v, \nabla f(x)\rangle = 0$. Par définition, il existe deux suites $(\lambda_k)_k \subset \mathbb{R}_+^*$ et $(v_k)_k \subset \mathbb{R}^n$ telles que $(\lambda_k) \downarrow 0$, $x + \lambda_k v_k \in \mathcal{L}_f(f(x))$ et $v = \lim_k v_k$. On a alors

$$\langle v, \nabla f(x)\rangle = \lim_{k\to+\infty} \langle v_k, \nabla f(x)\rangle.$$

Par ailleurs, comme $v_k \to v$, $(v_k)_k$ est une suite bornée et comme $(\lambda_k) \downarrow 0$, on a que $\lambda_k v_k \to 0$. Ainsi, quand $k \to +\infty$,

$$f(x + \lambda_k v_k) = f(x) + \langle \lambda_k v_k, \nabla f(x)\rangle + o(\lambda_k)$$

et comme $x + \lambda_k v_k \in \mathcal{L}_f(f(x))$, on a $f(x + \lambda_k v_k) = f(x)$. Alors

$$\langle \lambda_k v_k, \nabla f(x)\rangle = o(\lambda_k)$$

autrement dit $\lim_k \langle v_k, \nabla f(x)\rangle = 0$.

Pour la normalité du gradient de f en x à l'ensemble de niveau $f(x)$ de f, on procède comme précédemment sauf qu'on a seulement $f(x + \lambda_k v_k) \leq f(x)$ (au lieu d'avoir l'égalité comme précédemment). ∎

La Proposition 2.4 nous aide donc à visualiser le gradient par rapport aux lignes de niveau d'une fonction comme dans la Figure 1.

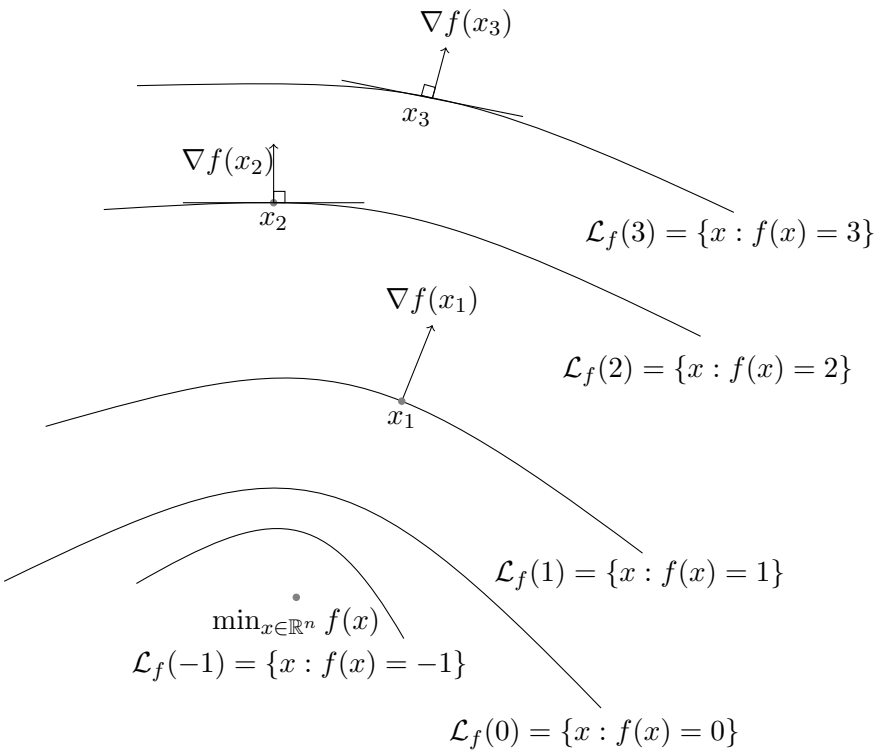

FIGURE 1 – Le gradient est un champs de vecteur orthogonaux aux lignes de niveau de f.

La Proposition 2.4 nous aide aussi à visualiser le gradient comme un champs de vecteurs normaux aux ensembles de niveaux de f comme dans la Figure 2.

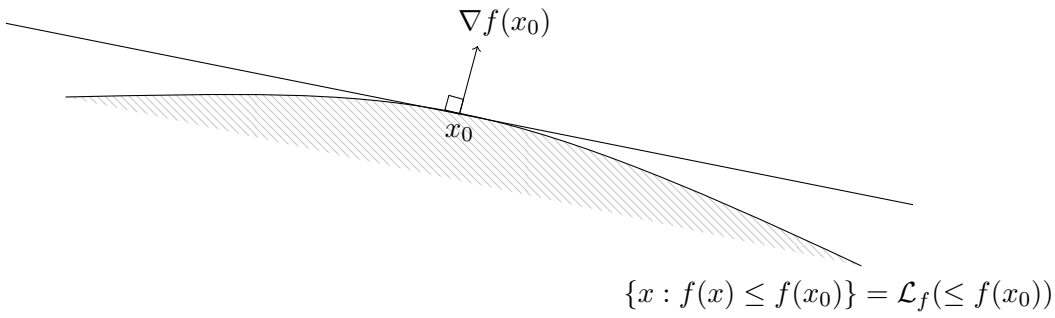

FIGURE 2 – Le gradient d'une fonction est un champs de vecteurs normaux aux ensembles de niveau de cette fonction.

Par ailleurs, du point de vue de l'optimisation, une bonne façon de voir le gradient de f en x est de se placer en x et de chercher la direction de plus forte pente. Il se trouve que c'est le gradient (renormalisé) de f en ce point qui indique la direction de plus forte pente. En effet, si $f : U \to \mathbb{R}$ est différentiable en x alors la direction $v \in S_2^{n-1} = \{x \in \mathbb{R}^n : \|x\|_2 = 1\}$ de plus forte pente est celle qui maximise

$$\lim_{t \to 0} \frac{f(x + tv) - f(x)}{t} = \partial_v f(x) = \langle \nabla f(x), v \rangle.$$

Or $\max_{v \in S_2^{n-1}} \langle \nabla f(x), v \rangle$ est atteint en $\nabla f(x) / \|\nabla f(x)\|_2$. Ainsi, en l'absence d'informations sup-

plémentaires sur l'approximation locale de f autres que son gradient, il faut donc bien aller dans la direction du gradient $\nabla f(x)$ pour accroître le plus la valeur de f à partir de x et donc aller dans la direction opposée au gradient $-\nabla f(x)$ pour faire décroître le plus f.

On peut même aller un peu plus loin sur le rôle du gradient pour les algorithmes de descente. En effet, le résultat suivant montre qu'une direction de descente doit nécessairement être négativement corréler avec le gradient, càd, une direction de descente doit pointer dans la direction opposée au gradient. On reprend la définition d'une direction de descente donnée dans la Définition 1.1.

Proposition 2.5. *Soit $f : U \to \mathbb{R}$ où U est un ouvert de $\mathbb{R}^n$. Soit $x \in U$. On suppose que f est différentiable en x. Soit $d \in \mathbb{R}^n$. On a :*

1) Si $\langle \nabla f(x), d \rangle < 0$ alors d est une direction de descente

2) Si d est une direction de descente alors $\langle \nabla f(x), d \rangle \leq 0$.

Preuve. On montre *1)* : On note $r = \langle \nabla f(x), d \rangle$. Comme f est différentiable en x, on a quand $\alpha \to 0$

$$f(x + \alpha d) = f(x) + \alpha \langle \nabla f(x), d \rangle + o(\alpha).$$

Il existe $\bar{\alpha} > 0$ tel que pour tout $0 \leq \alpha \leq \bar{\alpha}$, $|o(\alpha)| \leq \alpha |r|/2$. On en déduit que $f(x + \alpha d) < f(x)$ pour tout $0 < \alpha \leq \bar{\alpha}$.

On montre *2)* : Comme f est différentiable en x, on a quand $\alpha \to 0$

$$f(x + \alpha d) = f(x) + \alpha \langle \nabla f(x), d \rangle + o(\alpha).$$

Comme $f(x + \alpha d) < f(x)$ pour tout $0 < \alpha \leq \bar{\alpha}$, on en déduit que $\langle \nabla f(x), d \rangle + o(1) < 0$ quand $\alpha \to 0$ et donc par passage à la limite $\langle \nabla f(x), d \rangle \leq 0$. ∎

Ainsi, la Proposition 2.5 montre qu'une direction de descente de f en x pointe nécessairement dans la même direction que $-\nabla f(x)$. On peut par exemple prendre $d = -\nabla f(x)$ ou $d = -A\nabla f(x)$ où A est symétrique définie positive pour avoir $\langle \nabla f(x), d \rangle < 0$ quand $\nabla f(x) \neq 0$.

Conclusion : Le gradient d'une fonction f en un point x est orthogonal à la courbe de niveau $f(x)$ de f en x, il est aussi normal à l'ensemble de niveau $f(x)$ de f en x. C'est de plus la direction de plus forte pente de f en ce point x. Finalement, toute direction de descente doit être négativement corrélé avec le gradient. Il est donc naturel de choisir $-\nabla f(x)$ comme direction de descente dès qu'on peut le faire.

2.4.2 Algorithme de descente de gradient à pas optimal : la line search

Pour l'algorithme de descente de gradient c'est $-\nabla f(x_k)$ qui est la direction de descente. La dernière quantité qu'il reste à déterminer pour décrire entièrement le passage de l'itération k à $k + 1$ est le step size. Une manière de choisir le step size est de le choisir de telle sorte que f soit minimale sur la droite de descente càd à choisir η_k tel que f est minimale sur la droite $\{x_k - \eta \nabla f(x_k) : \eta \in \mathbb{R}\}$:

$$\eta_k \in \operatorname*{argmin}_{\eta \in \mathbb{R}} f(x_k - \eta \nabla f(x_k)). \tag{2.8}$$

C'est cette étape supplémentaire dans notre algorithme de descente de gradient qui est appelée la **line search**. Elle nécessite de résoudre un problème d'optimisation uni-dimensionnel vu que $\eta \in \mathbb{R} \to f(x_k - \eta \nabla f(x_k))$ est la fonction qu'on cherche à minimiser durant la line search.

Si $\nabla f(x_k) = 0$, on prends $\eta_k = 0$ car $\nabla f(x_k) = 0$ signifie que x_k est un point critique de f et donc, quand f est convexe, x_k est solution du problème $\min_{x \in \mathbb{R}^n} f(x)$ (et donc inutile de quitter

x_k : on prends $x_{k+1} = x_k$). Pour ce choix de step size, l'algorithme associé est appelé **algorithme de descente de gradient à pas optimal**.

input : $x_0 \in \mathbb{R}^n$: un point initial ; $\epsilon > 0$: un paramètre de critère d'arrêt
output: une approximation $x_k \in \mathbb{R}^n$ de x^* tel que $f(x^*) = \min_{x\in\mathbb{R}^n} f(x)$

1 **while** $\|x_{k+1} - x_k\|_2 \geq \epsilon$ **do**
2 $\quad \eta_k \in \operatorname{argmin}_{\eta\in\mathbb{R}} f(x_k - \eta\nabla f(x_k))$
3 $\quad x_{k+1} = x_k - \eta_k \nabla f(x_k)$
4 **end**

Algorithm 5: Algorithme de descente de gradient à pas optimal pour la construction d'une solution approchante au problème $\min_{x\in\mathbb{R}^n} f(x)$.

Le problème d'optimisation (2.8) qu'on retrouve à l'étape **2** de l'Algorithme 5 est un problème d'optimisation sur $\mathbb{R}$ qui peut se résoudre rapidement sous certaine hypothèse sur f (ce n'est cependant pas toujours le cas). On démontre maintenant la convergence de l'algorithme de descente de gradient à pas optimal.

Théorème 2.6. *On suppose que f est différentiable et c-convexe (càd f satisfait (2.6)) et que ∇f est Lipschitzienne sur tout bornés de $\mathbb{R}^n$, càd pour tout $M > 0$ il existe C_M tel que pour tout $x, y \in \mathbb{R}^n$, si $\|x\|_2, \|y\|_2 \leq M$ alors $\|\nabla f(x) - \nabla f(y)\|_2 \leq C_M \|x - y\|_2$.*

Alors l'algorithme de descente de gradient à pas optimal (voir Algorithme 5) converge : quel que soit x_0, la suite $(x_k)_k$ définie par (2.7) et (2.8) converge vers une solution du problème $\min_{x\in\mathbb{R}^n} f(x)$.

Preuve. La fonction $F : \eta \in \mathbb{R} \to f(u_k - \eta\nabla f(x_k))$ est $c\|\nabla f(x_k)\|$-convexe car pour tout $\eta, \mu \in \mathbb{R}$ et $0 \leq t \leq 1$, on a

$$\begin{aligned} F(t\eta + (1-t)\mu) &= f(t(u_k - \eta\nabla f(x_k)) + (1-t)(u_k - \mu\nabla f(x_k))) \\ &\leq tf(u_k - \eta\nabla f(x_k)) + (1-t)f(u_k - \mu\nabla f(x_k)) - \frac{c\|\nabla f(x_k)\|_2 t(1-t)}{2}|\eta - \mu|. \end{aligned}$$

Ainsi quand $\nabla f(x_k) \neq 0$, le problème de minimisation $\min_{\eta\in\mathbb{R}} F(\eta)$ admet une unique une solution η_k donnée par $F'(\eta_k) = 0$ càd η_k est telle que

$$\langle \nabla f(x_k), \nabla f(x_{k+1}) \rangle = 0. \tag{2.9}$$

Ce qui signifie que deux directions de descente consécutive sont orthogonales pour l'algorithme de descente de gradient à pas optimal.

D'après (2.9), on a

$$\langle \nabla f(x_{k+1}), x_{k+1} - x_k \rangle = \langle \nabla f(x_{k+1}), -\eta_k \nabla f(x_k) \rangle = 0.$$

Par ailleurs, comme f est c-convexe, on a

$$f(x_k) - f(x_{k+1}) \geq \langle \nabla f(x_k), x_k - x_{k+1} \rangle + \frac{c}{2}\|x_k - x_{k+1}\|_2^2.$$

On en déduit donc que

$$f(x_k) - f(x_{k+1}) \geq \frac{c}{2}\|x_k - x_{k+1}\|_2^2. \tag{2.10}$$

De plus, par définition de η_k, on a $f(x_{k+1}) = \min_{\eta\in\mathbb{R}} f(x_k - \eta\nabla f(x_k))$ alors $f(x_{k+1}) \leq f(x_k)$ donc $(f(x_k))_k$ est décroissante. Par ailleurs, f étant fortement convexe elle admet un unique minimum sur $\mathbb{R}^n$, noté x^*. Ainsi $(f(x_k))_k$ est une suite décroissante et minorée (par $f(x^*)$) donc elle converge. On déduit de (2.10) que $(x_{k+1} - x_k)_k$ converge vers 0. D'autre part, comme f est fortement convexe et que $(f(x_k))_k$ est bornée, on en déduit que $(x_k)_k$ est bornée, càd il existe M tel que pour tout $n, \|x_n\|_2 \leq M$. En appliquant la propriété Lipschitz de ∇f sur tout borné de $\mathbb{R}^n$, on a

$$\|\nabla f(x_{k+1}) - \nabla f(x_k)\|_2 \leq C_M \|x_{k+1} - x_k\|_2$$

et comme $\langle \nabla f(x_{k+1}), \nabla f(x_k) \rangle = 0$, on en déduit que

$$\|\nabla f(x_k)\|_2 \leq C_M \|x_{k+1} - x_k\|_2$$

et comme $x_{k+1} - x_k \to 0$ quand $k \to \infty$, on obtient que $\nabla f(x_k) \to 0$ quand $k \to \infty$. Finalement, la c-convexité de f donne

$$c\|x_k - x^*\|_2^2 \leq \langle \nabla f(x_k) - \nabla f(x^*), x_k - x^* \rangle = \langle \nabla f(x_k), x_k - x^* \rangle \leq \|\nabla f(x_k)\|_2 \|x_k - x^*\|_2$$

donc $c\|x_k - x^*\|_2 \leq \|\nabla f(x_k)\|_2$ et comme $\|\nabla f(x_k)\|_2 \to 0$ quand $k \to \infty$, on a bien $x_k \to x^*$ quand $k \to \infty$. ■

Remarque 2.7. *On a montré que pour tout k, on a $c\|x_k - x^*\|_2 \leq \|\nabla f(x_k)\|_2$. On a donc une majoration calculable de l'erreur d'approximation de x^* par x_k. En particulier, on peut se servir de $\|\nabla f(x_k)\|_2$ comme d'un critère d'arrêt pour l'algorithme de descente de gradient à pas optimal.*

2.4.3 Algorithme de descente de gradient à pas fixe

Il n'est cependant par tout le temps possible ou facile de déterminer le η_k optimal solution de (2.8). La solution la plus simple dans ce cas est de se donner une valeur a priori pour η_k qui reste la même, égale à η, pour chaque itération : on obtient alors les itérations

$$x_{k+1} = x_k - \eta\nabla f(x_k) \tag{2.11}$$

où η est un paramètre positif fixé. C'est la **méthode de descente de gradient à pas fixe**.

input : $x_0 \in \mathbb{R}^n$: un point initial ; $\epsilon > 0$: un paramètre de critère d'arrêt ; η : un step size
output: une approximation $x_k \in \mathbb{R}^n$ de x^* tel que $f(x^*) = \min_{x\in\mathbb{R}^n} f(x)$

```
while ‖x_{k+1} − x_k‖_2 ≥ ε do
    x_{k+1} = x_k − η∇f(x_k)
end
```

Algorithm 6: Algorithme de descente de gradient à pas fixe pour la construction d'une solution approchante au problème $\min_{x\in\mathbb{R}^n} f(x)$.

Contrairement à l'algorithme de descente à pas optimal chaque itération de l'Algorithme 6 ne nécessite "que" le calcul de $\nabla f(x_k)$ alors que pour l'Algorithme 5, on a besoin de résoudre un problème d'optimisation supplémentaire pour le calcul du step size η_k. Cependant, on peut espérer devoir faire moins d'itérations avec un choix optimal de step size plutôt qu'avec un choix de pas fixe.

Convergence de l'algorithme de descente de gradient à pas fixe sous hypothèse de convexité forte et de gradient Lipschitz. On montre maintenant la convergence de l'algorithme de descente de gradient à pas fixe pour un certain choix de η sous l'hypothèse de convexité forte de f et d'un gradient Lipschitz.

Théorème 2.8. *On suppose que f est différentiable et c-convexe (càd f satisfait (2.6)) et que ∇f est Lipschitzienne sur $\mathbb{R}^n$: il existe $C > 0$ tel que pour tout $x, y \in \mathbb{R}^n$, $\|\nabla f(x) - \nabla f(y)\|_2 \leq C \|x - y\|_2$.*

Si $0 < \eta < 2c/C$ alors l'algorithme de descente de gradient à pas fixe (voir Algorithme 6) converge : quel que soit l'initialisation $x_0 \in \mathbb{R}^d$, la suite $(x_k)_k$ définie par (2.11) converge vers une solution du problème $\min_{x\in\mathbb{R}^n} f(x)$. De plus, on a une vitesse de décroissance géométrique : pour tout $k \in \mathbb{N}$, $\|x_k - x^*\|_2 \leq \left(1 - 2c\eta + \eta^2 C^2\right)^{k/2} \|x_0 - x^*\|_2$.

Preuve. Comme f est c-convexe il existe une unique solution x^* au problème $\min_{x\in\mathbb{R}^n} f(x)$, de plus f est différentiable alors $\nabla f(x^*) = 0$. Pour tout $k \in \mathbb{N}$, on a

$$\|x_{k+1} - x^*\|_2^2 = \|x_k - x^*\|_2^2 - 2\eta\langle \nabla f(x_k) - \nabla f(x^*), x_k - x^*\rangle + \eta^2 \|\nabla f(x_k) - \nabla f(x^*)\|_2^2 .$$

D'après la propriété Lipschitz de ∇f, on a

$$\|\nabla f(x_k) - \nabla f(x^*)\|_2^2 \leq C^2 \|x_k - x^*\|_2^2$$

et comme f est c-convexe on a

$$\langle \nabla f(x_k) - \nabla f(x^*), x_k - x^*\rangle \geq c \|x_k - x^*\|_2^2 .$$

On en déduit que

$$\|x_{k+1} - x^*\|_2 \leq \left(1 - 2c\eta + \eta^2 C^2\right)^{1/2} \|x_k - x^*\|_2$$

et comme $0 < 1 - 2c\eta + \eta^2 C^2 < 1$ quand $0 < \eta < 2c/C$, on a bien la convergence de $(x_k)_k$ vers x^*. ∎

Convergence de l'algorithme de descente de gradient à pas fixe sous hypothèses de convexité et de gradient Lipschitz. On montre maintenant la convergence de l'algorithme de descente de gradient à pas fixe pour un certain choix de η sous l'hypothèse de convexité de f et d'un gradient Lipschitz. On ne suppose plus que f est fortement convexe contrairement au Théorème 2.8 du paragraphe précédent. On ne peux plus utiliser la propriété de 'croissante forte' du gradient : $\langle \nabla f(x_k) - \nabla f(x^*), x_k - x^*\rangle \geq c \|x_k - x^*\|_2^2$. Cependant on peut montrer l'existence de propriétés semblables à celles obtenues sous la c-convexité seulement sous les hypothèses de convexité et gradient Lipschitz.

Proposition 2.9. *On suppose $f : \mathbb{R}^n \to \mathbb{R}$ différentiable et telle que ∇f est Lipschitzienne sur $\mathbb{R}^n$: il existe $C > 0$ tel que pour tout $x, y \in \mathbb{R}^n$, $\|\nabla f(x) - \nabla f(y)\|_2 \leq C \|x - y\|_2$. Alors, pour tout $x, y \in \mathbb{R}^n$, on a*

$$f(y) \leq f(x) + \langle \nabla f(x), y - x\rangle + \frac{C}{2} \|y - x\|_2^2 . \tag{2.12}$$

Si de plus f est convexe alors, pour tout $x, y \in \mathbb{R}^n$, on a

$$f(y) - f(x) \leq \langle \nabla f(y), y - x\rangle - \frac{1}{2C} \|\nabla f(y) - \nabla f(x)\|_2^2 \tag{2.13}$$

et

$$\langle \nabla f(x) - \nabla f(y), x - y\rangle \geq \frac{1}{C} \|x - y\|_2 . \tag{2.14}$$

Preuve. On démontre d'abord le résultat (2.12). On pose $g : t \in [0,1] \to f(tx + (1-t)y)$ et on a $g'(t) = \langle \nabla f(tx + (1-t)y), x - y \rangle$ pour tout $t \in]0,1[$ (par exemple, en utilisant la chain rule). Le théorème fondamental de l'analyse et Cauchy-Schwartz disent que

$$\begin{aligned} f(x) - f(y) = g(1) - g(0) &= \int_0^1 g'(t)dt = \int_0^1 \langle \nabla f(tx + (1-t)y), x - y \rangle dt \\ &= \langle \nabla f(y), x - y \rangle + \int_0^1 \langle \nabla f(tx + (1-t)y) - \nabla f(y), x - y \rangle dt \\ &\leq \langle \nabla f(y), x - y \rangle + C \|x - y\|_2^2 \int_0^1 t dt = \langle \nabla f(x), y - x \rangle + \frac{C}{2} \|y - x\|_2^2 . \end{aligned}$$

On démontre maintenant (2.13). On a pour tout $z \in \mathbb{R}^n$,

$$f(y) - f(x) \leq f(y) - f(z) + f(z) - f(x) \leq \langle \nabla f(y), y - z \rangle + \langle \nabla f(x), z - x \rangle + \frac{C}{2} \|z - x\|_2^2 \quad (2.15)$$

car f est convexe, donc au-dessus de ses tangentes : $f(z) \geq f(y) + \langle \nabla f(y), z - y \rangle$ et on a le lemme de descente, càd (2.12) : $f(z) \leq f(x) + \langle \nabla f(x), z - x \rangle + (C/2) \|z - x\|_2^2$. On optimise en z la borne de droite dans (2.15) ; on obtient un minimum en $z = x - (1/C)(\nabla f(x) - \nabla f(y))$ (on est en train de minimiser une fonction différentiable et fortement convexe, il suffit alors de trouver son unique point critique). On remplace z par cette valeur dans le terme de droite de (2.15) pour obtenir

$$\begin{aligned} f(y) - f(x) \leq & \langle \nabla f(y), y - x + (1/C)(\nabla f(x) - \nabla f(y)) \rangle - (1/C) \langle \nabla f(x), \nabla f(x) - \nabla f(y) \rangle \\ & + \frac{1}{2C} \|\nabla f(x) - \nabla f(y)\|_2^2 = \langle \nabla f(y), y - x \rangle - \frac{1}{2C} \|\nabla f(x) - \nabla f(y)\|_2^2 . \end{aligned}$$

On montre l'inégalité de co-coercivité, càd (2.14). On applique (2.13) en inter-changeant les rôles de x et y :

$$\begin{aligned} f(y) - f(x) &\leq \langle \nabla f(y), y - x \rangle - \frac{1}{2C} \|\nabla f(y) - \nabla f(x)\|_2^2 \\ f(x) - f(y) &\leq \langle \nabla f(x), x - y \rangle - \frac{1}{2C} \|\nabla f(x) - \nabla f(y)\|_2^2 \end{aligned}$$

et en sommant ces deux inégalités, on obtient le résultat. ∎

L'inégalité (2.12) est parfois appelée le **lemme de descente** car elle implique qu'étant en un point $x \in \mathbb{R}^n$ et allant dans la direction opposée au gradient avec un step size en $(1/C)$ on fait décroître la valeur de la fonction objective (ce qui est bien une propriété de 'descente' et ce qu'on cherche à faire) :

$$\begin{aligned} f\left(x - \frac{1}{C}\nabla f(x)\right) - f(x) &\leq \langle \nabla f(x), \left(x - \frac{1}{C}\nabla f(x)\right) - x \rangle + \frac{C}{2} \left\| \left(x - \frac{1}{C}\nabla f(x)\right) - x \right\|_2^2 \\ &\leq -\frac{1}{2C} \|\nabla f(x)\|_2^2 . \end{aligned} \quad (2.16)$$

Le lemme de descente permet donc de quantifier le pas de descente vers l'optimum quand f est supposée en plus convexe. En effet, si $x^* \in \mathbb{R}^n$ est solution du problème $\min_{x \in \mathbb{R}^n} f(x)$ alors pour tout $x \in \mathbb{R}^n$, on a

$$f(x^*) - f(x) \leq f\left(x - \frac{1}{C}\nabla f(x)\right) - f(x) \leq -\frac{1}{2C} \|\nabla f(x)\|_2^2 .$$

Ainsi, on est sûr de faire décroître d'au moins $(-1/2C)\|\nabla f(x_k)\|_2^2$ la fonction objective en faisant une étape de descente en $(-1/C)\nabla f(x_k)$ partant de x_k.

Le lemme de descente est aussi la propriété clef qui nous a servie à démontrer la propriété (2.14) qu'on appelle parfois la 'co-coercivité'. C'est cette propriété qu'on utilise pour démontrer la propriété de décroissance de $(f(x_k))_k$ vers la valeur optimale $\min_{x\in\mathbb{R}^n} f(x)$ le long des itérés de l'algorithme de descente de gradient à pas fixe et qui remplace la propriété de croissance forte du gradient quand on a de la convexité forte.

Théorème 2.10. *On suppose $f : \mathbb{R}^n \to \mathbb{R}$ différentiable, convexe et telle que ∇f est Lipschitzienne sur $\mathbb{R}^n$: il existe $C > 0$ tel que pour tout $x, y \in \mathbb{R}^n$, $\|\nabla f(x) - \nabla f(y)\|_2 \leq C\|x - y\|_2$. On suppose qu'il existe x^* une solution au problème $\min_{x\in\mathbb{R}^n} f(x)$.*

Si $\eta = 1/C$ alors l'algorithme de descente de gradient à pas fixe (voir Algorithme 6) est tel que pour tout $k \in \mathbb{N}$,

$$f(x_k) - f(x^*) \leq \frac{2C\|x_0 - x^*\|}{k}.$$

Preuve. Comme f est convexe différentiable et x^* est solution au problème $\min_{x\in\mathbb{R}^n} f(x)$ alors $\nabla f(x^*) = 0$. Soit $k \in \mathbb{N}^*$. On a

$$\begin{aligned}
&\|x_{k+1} - x^*\|_2^2 = \|x_k - (1/L)\nabla f(x_k) - x^*\|_2^2 \\
&= \|x_k - x^*\|_2^2 - \frac{2}{C}\langle \nabla f(x_k) - \nabla f(x^*), x_k - x^*\rangle + \frac{1}{C^2}\|\nabla f(x_k) - \nabla f(x^*)\|_2^2 \\
&\leq \|x_k - x^*\|_2^2 - \frac{1}{C^2}\|\nabla f(x_k) - \nabla f(x^*)\|_2^2
\end{aligned}$$

où on a utilisé la propriété de co-coercivité : $\langle \nabla f(x_k) - \nabla f(x^*), x_k - x^*\rangle \geq (1/C)\|x_k - x^*\|$ et la propriété Lipschitz du gradient : $\|\nabla f(x_k) - \nabla f(x^*)\|_2 \leq C\|x_k - x^*\|_2$. On en déduit donc que la suite $(\|x_k - x^*\|_2)_k$ décroît.

Par ailleurs, f étant convexe, elle est au-dessus de ses pentes et la suite $(\|x_k - x^*\|_2)_k$ décroît, donc

$$f(x_k) - f(x^*) \leq \langle \nabla f(x_k), x_k - x^*\rangle \leq \|\nabla f(x_k)\|_2 \|x_k - x^*\|_2 \leq \|\nabla f(x_k)\|_2 \|x_1 - x^*\|_2 .$$

On obtient ainsi que $\|\nabla f(x_k)\|_2 \geq [f(x_k) - f(x^*)]/\|x_1 - x^*\|_2$. De plus, en utilisant la propriété de descente (2.16) où on soustrait des deux côtés la quantité $f(x^*)$, on a

$$f(x_{k+1}) - f(x^*) \leq f(x_k) - f(x^*) - \frac{1}{2C}\|\nabla f(x_k)\|_2^2 \leq f(x_k) - f(x^*) - \beta(f(x_k) - f(x^*))^2$$

où $\beta := [2C\|x_1 - x^*\|_2]^{-1}$.

On pose $\delta_k = f(x_k) - f(x^*)$. On a alors $\delta_{k+1} \leq \delta_k$ et pour tout $k \in \mathbb{N}, \delta_{k+1} \leq \delta_k - \beta\delta_k^2$. En multipliant la dernière inégalité par $1/(\delta_k\delta_{k+1})$, on obtient $\beta(\delta_k/\delta_{k+1}) \leq 1/\delta_{k+1} - 1/\delta_k$ et vu que $\delta_{k+1} \leq \delta_k$, on en déduit que $\beta \leq 1/\delta_{k+1} - 1/\delta_k$. On somme cette dernière inégalité sur $k = 0, 1, \cdots, n$ et on utilise les télescopage de certains termes pour obtenir : $k\beta \leq 1/\delta_k - 1/\delta_0 \leq 1/\delta_k$; càd $\delta_k \leq 1/(k\beta)$. On conclut avec la définition de β. ∎

Contrairement au Théorème 2.1 obtenu sous convexité forte, on n'a pas ici dans le Théorème 2.10, la convergence de la suite $(x_k)_k$ vers une solution x^* mais seulement la convergence de $(f(x_k))_k$ vers $f(x^*)$. En fait, sans l'hypothèse de convexité forte, on ne sait pas s'il existe une unique solution au problème $\min_{x\in\mathbb{R}^n} f(x)$; il est donc plus difficile d'identifier un élément particulier de $\operatorname{argmin}_{x\in\mathbb{R}^n} f(x)$ vers lequel $(x_k)_k$ pourrait converger. Par contre toutes les solutions ont même valeur objective $f(x^*)$.

3 Algorithmes pour les problèmes d'optimisation sous contraintes

Dans la section précédente, les directions de descente p étaient obtenues par minimisation d'un développement de Taylor du second ordre de la fonction convexe f à minimiser. Le choix de p n'était pas contraint et la minimisation de ce critère était le seul objectif.

Pour un problème d'optimisation sous contraintes, le choix de la direction de descente ne peut plus uniquement se faire dans le seul objectif de minimiser une approximation de f : au point courant x_k, on doit s'assurer que l'itération suivante $x_{k+1} = x_k + p_k$ est dans l'ensemble des contraintes. On doit donc s'assurer que la direction de descente n'amène pas l'algorithme à produire des itérations qui sortent de l'ensemble de contrainte (où la fonction objectif f n'est peut-être même pas définie).

On considère un problème d'optimisation sous contrainte sous sa forme générale

$$\min_{x \in K} f(x) \tag{3.1}$$

où $K \subset \mathbb{R}^n$ est un ensemble convexe non vide et $f : \mathbb{R}^n \to \mathbb{R}$ est une fonction convexe deux fois différentiable. Le but de cette section est de construire des algorithmes permettant sous certaines hypothèses sur f d'approcher une solution du problème (3.1). Il existe de nombreux algorithmes permettant de résoudre ce problème. Ceux qu'on va étudier ici dans cette section sont des algorithmes de descentes de gradient sauf qu'on doit en plus s'assurer que les itérés successifs sont bien dans la contrainte K. Pour ce faire on va utiliser deux stratégies : soit on approche f en x_k par un DL1 (ou un DL2) et on minimise son approximation sous la contrainte que $x_k + p$ reste dans K, c'est ce qu'on appelle **l'algorithme du gradient conditionnel ou algorithme de Franck-Wolfe**, soit on fait une étape de descente de gradient comme s'il n'y avait pas de contrainte et ensuite on projette ce nouvel itéré sur la contrainte, c'est ce qu'on appelle **l'algorithme du gradient projeté**.

Il existe d'autres idées clefs qui permettent de construire d'autres algorithmes. Par exemple, l'utilisation de la dualité Lagrangienne permet de construire des algorithmes. Une idée clef utilisée dans l'**algorithme d'Uzawa** est que la contrainte du problème dual est souvent beaucoup plus simple que la contrainte primale : en effet, projeter sur K est souvent beaucoup plus compliqué que projeter la variable duale μ sur son ensemble de contrainte $\{\mu \geq 0\}$. L'idée est alors de faire des algorithmes de 'monté' de gradient projeté ou conditionnel sur le fonction duale – on parle ici d'algorithme de montée car le problème dual est un problème de maximisation d'une fonction concave, on va donc aller dans la direction du gradient plutôt que dans son opposé. On verre aussi d'autres algorithmes utilisant la dualité Lagrangienne dans leurs construction. On commence par des algorithmes 'primal' (càd qui n'utilisent pas d'argument de dualité dans leur construction) que sont les algorithmes de descentes de gradient projeté et conditionnel.

3.1 Algorithme de gradient projeté

Dans cette section, on construit un algorithme permettant d'approcher une solution au problème

$$\min_{x \in K} f(x) \tag{3.2}$$

où $f : U \to \mathbb{R}$ est une fonction convexe différentiable sur un ouvert convexe U de $\mathbb{R}^n$ et K est un sous-ensemble convexe fermé de $\mathbb{R}^n$ inclut dans U.

D'après la condition du premier ordre, on sait que x^* est solution du problème (3.2) si et seulement si pour tout $y \in K$, $\langle \nabla f(x^*), y - x^* \rangle \geq 0$. Par ailleurs, en appliquant la condition du

premier ordre à

$$\text{proj}_K : \left\{ \begin{array}{ccc} \mathbb{R}^n & \longrightarrow & K \\ x & \longrightarrow & \text{argmin}_{z\in K} \|x-z\|_2 \end{array} \right.$$

l'opérateur de projection sur K, on voit que pour tout $x \in \mathbb{R}^n$, $p = \text{proj}_K(x)$ si et seulement si pour tout $y \in K, \langle p-x, y-p\rangle \geq 0$. Car on a $p = \text{proj}_K(x)$ si et seulement si $p \in \text{argmin}_{z\in K}(1/2)\|x-z\|_2^2$ et le gradient de $z \to (1/2)\|z-x\|_2^2$ en p vaut $p-x$.

On a donc la suite d'équivalence :

$$\begin{aligned} x^* \in \underset{x\in K}{\text{argmin}}\, f(x) &\Leftrightarrow \forall y \in K, \langle \nabla f(x^*), y-x^*\rangle \geq 0 \\ &\Leftrightarrow \forall y \in K, \langle x^* - (x^* - \nabla f(x^*)), y-x^*\rangle \geq 0 \\ &\Leftrightarrow x^* = \text{proj}_K(x^* - \nabla f(x^*)). \end{aligned}$$

On obtient donc que x^* est solution d'une équation de point fixe. Par ailleurs, minimiser f où minimiser ηf pour un certain $\eta > 0$ sont deux problème équivalent. On peut donc remplacer f par ηf dans ce qui précède. Ainsi

$$x^* \in \underset{x\in K}{\text{argmin}}\, f(x) \Leftrightarrow \forall \eta > 0, x^* = \text{proj}_K(x^* - \eta\nabla f(x^*)).$$

On peut donc utiliser un algorithme de point fixe pour approcher une solution de l'équation de point fixe précédente : étant donné $\eta > 0$,

$$x_{k+1} = \text{proj}_K(x_k - \eta\nabla f(x_k)) \tag{3.3}$$

C'est l'algorithme du **gradient projeté à pas fixe**. Ici on parle de pas fixe car on choisit un step size contant égal à η pour toutes les itérations. On peut généraliser cet algorithme en choisissant un step size η_k différent à chaque étape de descente. On obtient ainsi **l'algorithme du gradient projeté** pour une suite de step size $(\eta_k)_k$ donnée en entrée.

input : $x_0 \in \mathbb{R}^n$: un point initial ; $\epsilon > 0$: un paramètre de critère d'arrêt et $(\eta_k)_k \subset \mathbb{R}_+^*$ une suite de steps size
output: une solution approchante au problème $\min_{x\in K} f(x)$

1 **while** $\|x_{k+1} - x_k\|_2 \geq \epsilon$ **do**
2 $\quad y_{k+1} = x_k - \eta_k \nabla f(x_k)$ (étape forward)
3 $\quad x_{k+1} = \text{proj}_K(y_{k+1})$ (étape backward)
4 **end**

Algorithm 7: Algorithme de descente de gradient projeté pour la construction d'une solution approchante au problème $\min_{x\in K} f(x)$.

Dans l'Algorithme 7, on différencie l'étape de descente de gradient de l'étape **2**, qu'on appelle étape **forward** de l'étape de projection de **3** appelée **backward**. Le principe est de faire dans l'étape forward comme si on n'avait pas de contrainte K et donc de faire une descente dans la direction opposée au gradient (c'est ce qu'on fait dans l'algorithme de descente de gradient sans contrainte) et ensuite on s'assure que les itérés sont bien dans la contrainte K grâce à l'étape backward qui projette l'itération sur la contrainte K.

Théorème 3.1. *On suppose que $f : U \to \mathbb{R}$ est c-convexe et de classe $\mathcal{C}^1$. On suppose que ∇f est C-Lipschitzien sur U. Si $0 < \eta < 2c/C^2$ alors l'algorithme du gradient projeté à pas fixe converge : quelque soit le point initial x_0, la suite $(x_k)_k$ du gradient projeté à pas fixe $\eta_k = \eta$ définie dans (3.3) converge vers la solution de $\min_{x\in K} f(x)$. On a de plus une décroissance à vitesse géométrique vu que pour tout $k \in \mathbb{N}$, $\|x_{k+1} - x^*\|_2 \leq (1 + \eta^2C^2 - 2\eta c)^{k/2} \|x_0 - x^*\|_2$ et que $1 + \eta^2C^2 - 2\eta c < 1$ quand $0 < \eta < 2c/C^2$.*

Preuve. La suite $(x_k)_x$ est issue d'un algorithme de point fixe du style $x_{k+1} = F(x_k)$ où ici $F : x \in U \to \text{proj}_K(x - \eta\nabla f(x))$. Si F est strictement contractante, càd pour un certain $0 \leq \gamma < 1$, pour tout $x, y \in U$, $\|F(x) - F(y)\|_2 \leq \gamma \|x - y\|_2$ alors

$$\|x_{k+1} - x^*\|_2 = \|F(x_k) - F(x^*)\|_2 \leq \gamma \|x_k - x^*\|_2$$

et donc $(\|x_k - x^*\|_2)_k$ décroît géométriquement vers 0.

Il suffit donc de montrer que $F : x \in U \to \text{proj}_K(x-\eta\nabla f(x))$ est strictement contractante pour avoir le résultat. On montre ce résultat en deux étapes : 1) $x \in U \to x - \eta\nabla f(x)$ est strictement contractante quand $0 < \eta < 2c/C^2$, 2) $x \in \mathbb{R}^n \to \text{proj}_K(x)$ est contractante. La conclusion sera ensuite immédiate vu que la composée d'une fonction contractante et d'une fonction strictement contractante est bien strictement contractante.

On commence par montrer que $x \in U \to x - \eta\nabla f(x)$ est strictement contractante quand $0 < \eta < 2c/C^2$. Comme f est c-convexe et gradient Lipschitz, pour tout $x, y \in U$, on a

$$\begin{aligned}
&\|(x - \eta\nabla f(x)) - (y - \eta\nabla f(y))\|_2^2 = \|(x - y) + \eta(\nabla f(y) - \nabla f(x))\|_2^2 \\
&\leq \|x - y\|_2^2 + \eta^2 \|\nabla f(y) - \nabla f(x)\|_2^2 + 2\eta\langle x - y, \nabla f(y) - \nabla f(x)\rangle \\
&\leq (1 + \eta^2C^2 - 2\eta c) \|x - y\|_2^2 .
\end{aligned}$$

Comme $0 < \eta < 2c/C^2$, on a $0 < 1 + \eta^2C^2 - 2\eta c < 1$ donc $x \in U \to x - \eta\nabla f(x)$ est strictement contractante.

On montre dans le lemme suivant que l'opérateur de projection est contractant, ce qui conclut la preuve. ∎

Lemme 3.2. *Soit K un sous-ensemble convexe fermé de $\mathbb{R}^n$. On note*

$$\text{proj}_K : \left\{ \begin{array}{ccc} \mathbb{R}^n & \longrightarrow & K \\ x & \longrightarrow & \text{argmin}_{z\in K} \|x - z\|_2 \end{array} \right.$$

l'opérateur de projection sur K. Alors, pour tout $x, y \in \mathbb{R}^n$, on a

$$\|\text{proj}_K(y) - \text{proj}_K(x)\|_2 \leq \|x - y\|_2 .$$

Preuve. Pour tout $x, y \in \mathbb{R}^n$, la condition du premier ordre donne

$$\begin{aligned}
\langle x - \text{proj}_K(x), \text{proj}_K(y) - \text{proj}_K(x)\rangle \leq 0 \\
\langle y - \text{proj}_K(y), \text{proj}_K(x) - \text{proj}_K(y)\rangle \leq 0.
\end{aligned}$$

Alors, en additionnant les deux expressions, on obtient

$$\langle x - \text{proj}_K(x) - y + \text{proj}_K(y), \text{proj}_K(y) - \text{proj}_K(x)\rangle \leq 0$$

donc, par Cauchy-Schwarz,

$$\|\text{proj}_K(y) - \text{proj}_K(x)\|_2^2 \leq \langle x - y, \text{proj}_K(y) - \text{proj}_K(x)\rangle \leq \|x - y\|_2 \|\text{proj}_K(y) - \text{proj}_K(x)\|_2$$

et donc proj_K est bien une contraction. ∎

3.2 L'algorithme du gradient conditionnel et algorithme de Frank-Wolfe.

Dans la section précédente, nous avons vu que la méthode de descente de gradient est un algorithme itératif de la forme

$$x_{k+1} = x_k - \eta_k \nabla f(x_k)$$

qui a pour but d'approcher le minimum d'une fonction convexe différentiable f pour un problème d'optimisation sans contrainte. Pour un problème d'optimisation sous contrainte comme (3.2), on n'a aucune assurance que partant d'un point $x_k \in K$, l'itérée suivante x_{k+1} obtenue après une étape de descente de gradient reste dans la contrainte K. On doit cependant s'assurer que la suite des itérés d'un algorithme pour la résolution approximative de (3.2) reste bien dans K ; f n'étant peut-être même pas définie en dehors de K. Pour ce faire on a déjà vu l'idée de projeter les itérés sur la contrainte K, c'est ce qui a donné l'algorithme de descente de gradient projeté dans la sous-section précédente. Dans cette section, on utilise une approche différente qui consiste à ne chercher que des directions de descente et un step size assurant directement que la prochaine itération reste bien dans la contrainte. On n'a donc plus besoin de projeter sur K pour s'assurer que l'algorithme évolue bien dans K ; cependant la recherche d'une direction de descente est plus compliquée car elle nécessite de résoudre un problème sous contrainte. C'est cette méthode qui est appelé **algorithme de descente de gradient conditionnel**.

L'algorithme de descente de gradient conditionnel utilise directement l'idée centrale des algorithmes de Newton : on souhaite ici minimiser f sur K, l'idée de Newton pour ce problème est alors de minimiser successivement des développements limités de f sur K. Si on fait un DL1 alors on trouve l'algorithme dit de **Frank-Wolfe** et si on fait une DL2, on trouve un algorithme de Newton conditionnel. Dans les deux cas, on parle bien d'une méthode de descente de gradient conditionnel car la recherche de la direction de descente partant de x_k est conditionnée au fait de rester dans la contrainte (contrairement au cas des algorithmes de descente de gradient de la section précédente qui ne devait remplir aucune condition).

On commence par l'algorithme de Frank-Wolfe qui utilise des DL1 de f localement en les itérés x_k : quand $p \to 0$, $f(x_k + p) = f(x_k) + \langle \nabla f(x_k), p \rangle + o(\|p\|_2)$. Au lieu de chercher à minimiser f sur K, on va minimiser son DL1 sur K, c'est ainsi qu'on obtient la nouvelle itération :

$$x_{k+1} = x_k + p_k$$

où

$$p_k \in \operatorname*{argmin}_{p \in \mathbb{R}^n} \Big(\langle \nabla f(x_k), p \rangle : x_k + p \in K \Big). \tag{3.4}$$

C'est l'algorithme de Frank-Wolfe. Cet algorithme est particulièrement utile car la fonction objectif impliquée pour trouver le vecteur de descente p_k est linéaire. En particulier, si la contrainte K est formée uniquement de contraintes linéaires alors le problème d'optimisation (3.4) est un problème de programmation linéaire et peut donc se résoudre assez rapidement.

Si maintenant on fait un DL2 de f en les itérés x_k de l'algorithme : on a quand $p \to 0$,

$$f(x_k + p) = f(x_k) + \langle \nabla f(x_k), p \rangle + \frac{1}{2} p^\top \nabla^2 f(x_k) p + o(\|p\|_2^2).$$

Le principe de Newton est de minimiser cette approximation locale de f pour déterminer la prochaine itération : $x_{k+1} = x_k + p_k$ où

$$p_k \in \operatorname*{argmin}_{p \in \mathbb{R}^n} \Big(\langle \nabla f(x_k), p \rangle + \frac{1}{2} p^\top \nabla^2 f(x_k) p : x_k + p \in K \Big). \tag{3.5}$$

Ce vecteur de descente est appelé **vecteur de descente conditionnelle de Newton** et nécessite de résoudre un problème d'optimisation quadratique sous contrainte. De même que pour les problèmes d'optimisation sans contrainte, le calcul de la Hessienne d'une fonction peut être coûteux en temps de calcul. On utilise alors la même approximation que celle utilisée plus haut pour le passe de l'algorithme de descente de Newton à celui de descente de gradient : on remplace $\nabla^2 f(x_k)$ par $(1/\eta_k)I_n$. On obtient ainsi un algorithme de descente de gradient conditionnelle au second ordre : $x_{k+1} = x_k + p_k$ où

$$p_k \in \operatorname*{argmin}_p \Big(\langle \nabla f(x_k), p \rangle + \frac{1}{2\eta_k} \|p\|_2^2 : x_k + p \in K \Big).$$

Comme précédemment, la contrainte '$x_k + p \in K$" est ici pour assurer que l'itération suivante x_{k+1} reste bien dans K.

3.3 Algorithme d'Uzawa

L'algorithme du gradient conditionnel vu dans la sous-section précédente est efficace si on sait résoudre facilement les problèmes d'optimisation sous contrainte que $x_k + p \in K$ de fonction objectives linéaires ou quadratiques. Ce n'est cependant pas toujours le cas. De même, l'algorithme du gradient projeté ne peut fonctionner que si la projection sur K est facile à obtenir. Ce qui n'est pas le cas pour de nombreux exemple de convexes K. Il y a cependant quelques exemple d'ensemble convexes pour lesquels projeter dessus peut être réalisé de manière assez efficace. C'est par exemple le cas des ensembles convexes de la forme

$$K = \prod_{i=1}^{n} [a_i, b_i] \tag{3.6}$$

où $-\infty \leq a_i \leq b_i \leq +\infty$. La projection sur K d'un vecteur $x = (x_i)_{i=1}^n \in \mathbb{R}^n$ est explicitement donnée par

$$\mathrm{proj}_K(x) = (\min(\max(a_i, x_i), b_i))_{i=1}^n.$$

Projeter sur K revient donc à tronquer les coordonnées de x. C'est la première idée derrière l'algorithme d'Uzawa.

La deuxième idée derrière l'algorithme d'Uzawa est que même pour un problème primal avec des contraintes compliquées menant à de grandes difficultés pour construire d'un gradient projeté ou conditionnel, le problème dual à une contrainte beaucoup plus facile à gérer qui est de la forme (3.6) : $(\lambda, \mu) \in \mathbb{R}^r \times (\mathbb{R}_+)^l$.

L'algorithme d'Uzawa peut donc se décrire par les étapes suivantes :

1. On résout le problème dual grâce à l'algorithme du gradient projeté (avec un pas constant par exemple)
2. On résout le problème primal grâce à la solution du problème dual en résolvant un problème d'optimisation sans contrainte.

On peut donc voir l'algorithme d'Uzawa comme un algorithme de recherche de point-selle de la fonction de Lagrange pour lequel la solution au problème dual est approchée par un algorithme de gradient projeté. C'est le premier exemple de construction d'un algorithme utilisant la dualité Lagrangienne. On voit ici qu'on effectue un algorithme de 'montée' dans l'espace dual (on parle ici d'algorithme de 'montée' car le problème dual est un problème de maximisation d'une fonction concave – mais quitte à prendre l'opposé de la fonction duale, on retombe bien sur un problème de minimisation d'une fonction convexe pour qui on parle d'algorithme de descente de gradient).

On considère le problème d'optimisation convexe différentiable de la forme $\min_{x\in K} f(x)$ où $f : U \to \mathbb{R}$, U est un ouvert convexe de $\mathbb{R}^n$ et K est une contrainte de la forme

$$K = \{x \in U : Ax = b \text{ et } h_1(x) \leq 0, \dots, h_l(x) \leq 0\} \tag{3.7}$$

où U est un convexe ouvert et $h_1, \dots, h_l$ sont convexes et de classe $\mathcal{C}^1$. On voit que $K = \{x \in U : F(x) \leq 0\}$ où on note

$$F : \left\{ \begin{array}{ccc} U & \to & \mathbb{R} \\ & & \\ x & \to & \begin{pmatrix} Ax - b \\ b - Ax \\ H(x) \end{pmatrix} \end{array} \right. \tag{3.8}$$

On considère la fonction de Lagrange définie par

$$\mathcal{L} : \left\{ \begin{array}{ccc} U \times (\mathbb{R}_+)^{2r+l} & \to & \mathbb{R} \\ (x, \mu) & \to & f(x) + \langle \mu, F(x) \rangle. \end{array} \right.$$

La variable $\mu \in (\mathbb{R}_+)^{2r+l}$ est appelée **multiplicateur de Lagrange**.

On suppose que la contrainte K est qualifiée (par exemple, si K vérifie la condition de Slater : il existe $x_0 \in K$ tel que $h_j(x_0) < 0$ pour tout $j = 1, \dots, l$) alors le théorème de KKT dit qu'il y a équivalence entre :

a) $x^* \in U$ est solution du problème primal et $\mu^* \in (\mathbb{R}_+)^{2r+l}$ est solution du problème dual

b) (x^*, μ^*) est un point-selle de $\mathcal{L}$.

Par définition, dire que (x^*, μ^*) est un point-selle de $\mathcal{L}$ signifie que pour tout $x \in U$ et $\mu \in (\mathbb{R}_+)^{2r+l}$, on a

$$\mathcal{L}(x^*, \mu) \overset{(a)}{\leq} \mathcal{L}(x^*, \mu^*) \overset{(b)}{\leq} \mathcal{L}(x, \mu^*) \tag{3.9}$$

et donc d'après *(a)*, on a $\langle \mu^* - \mu, F(x^*) \rangle \geq 0$ pour tout $\mu \in (\mathbb{R}_+)^{2r+l}$. On peut réécrire cette dernière inégalité sous la forme : pour tout $\eta > 0$, pour tout $\mu \in (\mathbb{R}_+)^{2r+l}$,

$$\langle \mu^* - \mu, \mu^* - (\eta F(x^*) + \mu^*) \rangle \leq 0.$$

On en déduit donc que μ^* est la projection sur $(\mathbb{R}_+)^{2r+l}$ de $\eta F(x^*) + \mu^*$. En d'autres termes, μ^* est solution de l'équation de point fixe

$$\mu^* = \text{proj}_{(\mathbb{R}_+)^{2r+l}} (\eta F(x^*) + \mu^*)$$

pour tout $\eta > 0$. Comme dans la Section 3.1, on peut approcher μ^* par un algorithme de gradient projeté : $\mu_{k+1} = \text{proj}_{(\mathbb{R}_+)^{2r+l}} (\eta F(x^*) + \mu_k)$.

Par ailleurs, au vu de *(b)* dans (3.9), nous pouvons introduire **l'algorithme d'Uzawa** (à pas fixe) : étant donné $\mu_0 \in (\mathbb{R}_+)^{2r+l}$, on construit les itérés par

$$\begin{aligned} x_k &\in \operatorname*{argmin}_{x \in U} \mathcal{L}(x, \mu_k) \\ \mu_{k+1} &= \text{proj}_{(\mathbb{R}_+)^{2r+l}} (\eta F(x_k) + \mu_k) \end{aligned} \tag{3.10}$$

Le problème (3.10) est un problème d'optimisation convexe non contraint qui peut alors se résoudre par descente de gradient par exemple. On démontre maintenant la convergence de l'algorithme d'Uzawa.

Théorème 3.3. *Soit $f : \mathbb{R}^n \to \mathbb{R}$ une fonction c-convexe (définie en (2.6)) et différentiable. On suppose que F (définie en (3.8)) est convexe et C-Lipschitzienne. On suppose que $\mathcal{L}$ admet un point-selle (x^*, μ^*).*

Si $0 < \mu < 2c/C^2$ alors l'algorithme d'Uzawa converge : quel que soit $\mu_0 \in (\mathbb{R}_+)^{2r+l}$ la suite $(x_k)_k$ définie par l'algorithme d'Uzawa dans (3.10) converge vers la solution x^* du problème $\min_{x\in K} f(x)$.

Preuve. Comme $\mathcal{L}$ admet un point-selle (x^*, μ^*), le problème primal a une solution donnée par x^* et comme f est c-convexe, cette solution est unique. De même pour $U = \mathbb{R}^n$, le problème d'optimisation (3.10) (étant donné $\mu_k \in (\mathbb{R}_+)^{2r+l}$) admet une unique solution, donnée par x_k.

D'après *(b)* de (3.9), x^* minimise $x \in \mathbb{R}^n \to \mathcal{L}(x, \mu^*) = f(x) + \langle \mu^*, F(x)\rangle$ où f est convexe et différentiable et $x \to \langle \mu^*, F(x)\rangle$ est convexe car $\mu^* \geq 0$ et F est convexe. De même x_k minimise $x \in \mathbb{R}^n \to \mathcal{L}(x, \mu_k) = f(x) + \langle \mu^*, F(x)\rangle$ qui est la somme d'une fonction convexe différentiable et d'une fonction convexe. Alors, en écrivant les inéquations d'Euler satisfaites par x^* et x_k, données par la Proposition 3.4, on a pour tout $x \in \mathbb{R}^n$,

$$\begin{aligned}\langle f(x^*), x - x^*\rangle + \langle \mu^*, F(x) - F(x^*)\rangle \geq 0\\ \langle f(x_k), x - x_k\rangle + \langle \mu_k, F(x) - F(x_k)\rangle \geq 0.\end{aligned}$$

En prenant, $x = x_k$ dans la première inégalité et $x = x^*$ dans la deuxième et en sommant, on obtient

$$\langle f(x^*) - f(x_k), x_k - x^*\rangle + \langle \mu^* - \mu_k, F(x_k) - F(x^*)\rangle \geq 0$$

et en utilisant la forte convexité de f, on a $\langle f(x^*) - f(x_k), x_k - x^*\rangle \leq -c\,\|x_k - x^*\|$. On obtient donc

$$\langle \mu_k - \mu^*, F(x_k) - F(x^*)\rangle \leq -c\,\|x_k - x^*\|_2^2\,.$$

D'autre part, la projection sur $(\mathbb{R}_+)^{2r+l}$ est contractante (voir Lemme 3.2), on a donc

$$\begin{aligned}\|\mu_{k+1} - \mu^*\|_2 &= \left\|\mathrm{proj}_{(\mathbb{R}_+)^{2r+l}}(\eta F(x_k) + \mu_k) - \mathrm{proj}_{(\mathbb{R}_+)^{2r+l}}(\eta F(x^*) + \mu^*)\right\|_2\\ &\leq \|\eta(F(x_k) - F(x^*)) + \mu_k - \mu^*\|_2\,.\end{aligned}$$

On obtient en combinant les deux derniers résultats

$$\begin{aligned}\|\mu_{k+1} - \mu^*\|_2^2 &\leq \|\eta(F(x_k) - F(x^*)) + \mu_k - \mu^*\|_2^2\\ &\leq \|\mu_k - \mu^*\|_2^2 + \eta^2\,\|F(x_k) - F(x^*)\| + 2\eta\langle F(x_k) - F(x^*), \mu_k - \mu^*\rangle\\ &\leq \|\mu_k - \mu^*\|_2^2 + \eta^2 C^2\,\|x_k - x^*\|_2^2 - 2c\eta\,\|x_k - x^*\|_2^2\end{aligned}$$

et comme $0 < \mu < 2c/C^2$, on peut trouver $\beta > 0$ tel que $(\eta^2 C^2 - 2c\eta) < -\beta$, on obtient

$$\beta\,\|x_k - x^*\|_2^2 \leq \|\mu_k - \mu^*\|_2^2 - \|\mu_{k+1} - \mu^*\|_2^2\,. \tag{3.11}$$

La dernière inégalité montre que $(\|\mu_k - \mu^*\|_2^2)_k$ décroît donc elle converge (elle est minorée par 0) donc le membre de droite de (3.11) tends vers 0 et donc $(x_k)_k$ tends vers x^*. ∎

Proposition 3.4. *Soit U un ouvert convexe de $\mathbb{R}^n$. Soient $f, g : U \to \mathbb{R}$ deux fonctions convexes. On suppose (seulement) que f est différentiable (g n'est pas supposée différentiable). Soit K un convexe fermé de $\mathbb{R}^n$ inclut dans U. Soit $x^* \in K$, il y a équivalence entre :*

1) $x^* \in \mathrm{argmin}_{x\in K} f(x) + g(x)$

2) pour tout $y \in K$, $\langle \nabla f(x^*), y - x^*\rangle + g(y) - g(x^*) \geq 0$.

Preuve. On montre que *1)* implique *2)* : Pour tout $y \in K$, on a quand $\lambda \to 0$, $f(x^* + \lambda(y - x^*)) = f(x^*) + \langle \nabla f(x^*), \lambda(y - x^*) \rangle + o(\lambda)$. Comme $x^* + \lambda(y - x^*) \in K$ par convexité de K, on a

$$f(x^* + \lambda(y - x^*)) + g(x^* + \lambda(y - x^*)) \geq f(x^*) + g(x^*).$$

Alors quand $0 \leq \lambda \leq 1$ et $\lambda \to 0$, on a

$$\langle \nabla f(x^*), \lambda(y - x^*) \rangle + o(\lambda) + g(x^* + \lambda(y - x^*)) - g(x^*) \geq 0.$$

De plus, par convexité de g, on a

$$g(x^* + \lambda(y - x^*)) - g(x^*) \leq \lambda(g(y) - g(x^*)).$$

On obtient donc, quand $0 \leq \lambda \leq 1$ et $\lambda \to 0$,

$$\langle \nabla f(x^*), y - x^* \rangle + o(1) + g(y) - g(x^*) \geq 0.$$

Le résultat en découle par passage à la limite.

On montre que *2)* implique *1)* : Pour tout $y \in K$, on a par convexité de f que $f(y) - f(x^*) \geq \langle \nabla f(x^*), y - x^* \rangle$ et donc

$$f(y) + g(y) - (f(x^*) + g(x^*)) \geq \langle \nabla f(x^*), y - x^* \rangle + g(y) - g(x^*) \geq 0.$$

■

Remarque 3.5. *La mise à jour du coefficient de Lagrange par l'algorithme du gradient projeté :*

$$\mu_{k+1} = \mathrm{proj}_{(\mathbb{R}_+)^{2r+l}} (\eta F(x_k) + \mu_k)$$

est très simple à faire vu que la projection est ici un simple seuillage par coordonnées. Le seul coût computationnel de l'algorithme d'Uzawa est donc la résolution d'un problème fortement convexe sans contrainte : $x_k \in \mathrm{argmin}_{x \in U} \mathcal{L}(x, \mu_k)$.

Remarque 3.6. *La mise à jour de* μ_k *a été introduite comme un algorithme de point fixe vu que*

$$\mu^* = \mathrm{proj}_{(\mathbb{R}_+)^{2r+l}} (\eta F(x^*) + \mu^*)$$

. On peut montrer que c'est vraiment un algorithme de gradient projeté en regardant le problème dual dont μ^* *est solution. En effet,*

$$\mu^* \in \underset{\mu \geq 0}{\mathrm{argmax}}\, \psi(\mu) \text{ où } \psi : \mu \in (\mathbb{R}_+)^{2r+l} \to \inf_{x \in \mathbb{R}^n} \mathcal{L}(x, \mu).$$

Pour tout $\mu \geq 0$*, on note* x_μ *l'unique solution au problème* $\inf_{x \in \mathbb{R}^n} \mathcal{L}(x, \mu)$*. On a* $\psi(\mu) = \mathcal{L}(x_\mu, \mu)$*. Sous des hypothèses faibles, on peut montrer que* $\mu \to x_\mu$ *est différentiable alors* ψ *l'est aussi et, si on note* $J(G)(x)$ *la Jacobienne d'une fonction* G *en un point* x*, alors on a pour tout* $\mu \geq 0$*,*

$$\nabla \psi(\mu) = J(\mu \to x_\mu)(\mu)^\top \nabla f(x_\mu) + F(x_\mu) + J(\mu \to x_\mu)(\mu^*)^\top J(F)(x_\mu)^\top \mu = F(x_\mu)$$

car $\nabla f(x_\mu) + J(F)(x_\mu)^\top \mu = 0$ *vue que* x_μ *minimise* $x \in \mathbb{R}^n \to \mathcal{L}(x, \mu)$ *sur* $\mathbb{R}^n$ *qui est une fonction convexe donc* $\nabla \mathcal{L}(\cdot, \mu)(x_\mu) = 0$*. Ainsi l'algorithme du gradient projeté pour approcher* μ^* *(solution d'un problème de maximisation) est donné par*

$$\mu_{k+1} = \mathrm{proj}_{(\mathbb{R}_+)^{2r+l}} (\mu_k + \eta \nabla \psi(\mu_k)) = \mathrm{proj}_{(\mathbb{R}_+)^{2r+l}} (\mu_k + \eta \nabla F(x_{\mu_k}))$$

où x_{μ_k} *minimise* $\mathcal{L}(\cdot, \mu_k)$ *sur* $\mathbb{R}^n$*. C'est exactement l'algorithme d'Uzawa.*

3.4 Les méthodes de barrières : pénalisation des contraintes

Les méthodes de barrières sont aussi connues sous le nom de méthodes de points intérieurs, au-delà du cadre des problèmes de programmation linéaire. Elles sont utilisées pour construire des solutions approchantes aux problèmes d'optimisation sous contraintes.

On considère ici les problèmes d'optimisation convexe différentiable (OCD) comme dans (P1) (càd les contraintes d'égalité sont affines et la fonction objectif ainsi que les contraintes d'inégalités sont convexes et $\mathcal{C}^1$) : pour U un ouvert convexe de $\mathbb{R}^n$,

$$\boxed{\min_{x\in K} f(x) \text{ où } K = \left\{ x \in U : \begin{array}{c} h_i(x) \leq 0, \forall i \in \mathcal{I} \\ Ax = b \end{array} \right\} \text{ et } \begin{array}{l} f, (h_i)_{i\in\mathcal{I}} \text{ sont convexes et } \mathcal{C}^1 \\ A \in \mathbb{R}^{m\times n}, b \in \mathbb{R}^m. \end{array}} \tag{P1}$$

Les méthodes de barrières se proposent de trouver une solution approchante au problème (P1) en résolvant une suite de problèmes d'optimisation sans contraintes d'inégalité. Pour cela, on ajoute à la fonction objectif $f(\cdot)$ une fonction de coût qui pénalise les points de K qui s'approche de la frontière de K (vu comme sous-ensemble de $\{x \in U : Ax = b\}$). C'est ce coût additionnel qui créé une sorte de barrière interdisant aux algorithmes de sortir de K.

Définition 3.7. *Une **fonction barrière pour** (P1) est une fonction continue* $\text{bar} : \overset{\circ}{K} \to \mathbb{R}$ *(où* $\overset{\circ}{K}$ *est l'intérieur de* K *dans* $\{x \in U : Ax = b\}$*) telle que* $\text{bar}(x) \to +\infty$ *quand* $\max_{i\in\mathcal{I}} h_i(x) \to 0^-$.

Les deux exemples classiques de fonctions barrières sont définies pour tout $x \in U$ par

$$\text{bar}(x) = -\sum_{i\in\mathcal{I}} \log(-h_i(x)) \text{ et } \text{bar}(x) = -\sum_{i\in\mathcal{I}} \frac{1}{h_i(x)}.$$

L'objectif est de dissuader les algorithmes évoluant dans l'intérieur de K de trop s'approcher de la frontière de K. On introduit alors une famille de fonctions objectif indexée par un paramètre $\mu > 0$:

$$B(x, \mu) = f(x) + \mu \text{bar}(x)$$

où $\text{bar}(\cdot)$ est une fonction barrière. Le problème d'optimisation associé est

$$\boxed{\min_{x\in U : Ax=b} B(x, \mu)} \tag{PBμ}$$

On obtient ainsi une suite de problèmes d'optimisation ((PBμ)) indexée par un paramètre $\mu > 0$. L'idée maintenant est de résoudre ces problèmes et de faire tendre μ vers 0 pour construire une solution approchante au problème d'origine (P1).

Remarque 3.8. *Une manière naturelle de construire une barrière (non-continue) du bord de* K *dans* $\{x \in U : Ax = b\}$ *est de prendre la fonction indicatrice de* $\{x \in U : h_i(x) \leq 0, i \in \mathcal{I}\}$*, càd, pour tout* $x \in U$*,*

$$\text{bar}(x) = \sum_{i\in\mathcal{I}} i_{\mathbb{R}_-}(h_i(x)) \text{ où } i_\Omega(t) = \left\{ \begin{array}{c} 0 \text{ si } t \in \Omega \\ +\infty \text{ si } t \notin \Omega \end{array} \right.$$

pour tout $t \in \mathbb{R}$ *et* $\Omega \subset \mathbb{R}$*. On peut voir les fonctions barrières de base* $t \in \mathbb{R}^*_- \to -\mu \log(-t)$ *et* $t \in \mathbb{R}^*_- \to -\mu/t$ *comme des fonctions continues convergeant simplement vers* $i_{\mathbb{R}_-}$ *sur* $\mathbb{R}^*_-$ *quand* $\mu \to 0^+$.

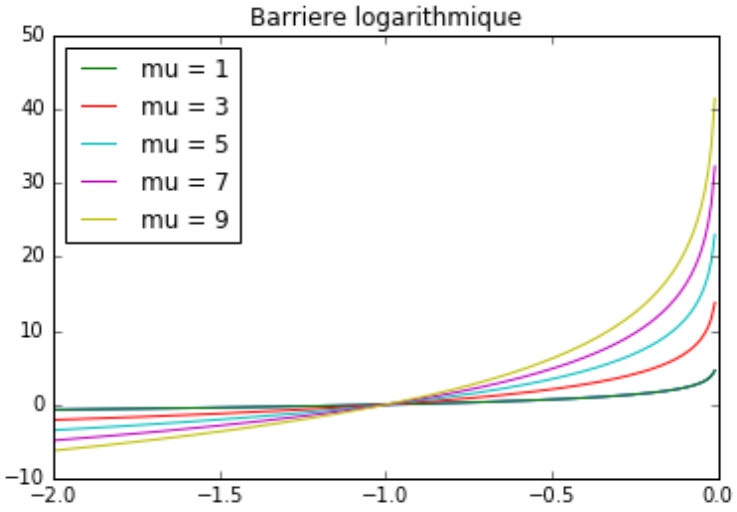

FIGURE 3 – Approximation de la barrière idéale $i_{\mathbb{R}_-}$ par les barrières logarithmiques $t \in \mathbb{R}_-^* \to -\mu \log(-t)$ quand $\mu \to 0^+$.

Théorème 3.9. *(Convergence des méthodes de barrière) On suppose que f et $(h_i)_{i\in\mathcal{I}}$ sont continues et que* bar *est une fonction barrière (continue) pour* (P1)*. Pour tout $\mu > 0$, on suppose que* (PBμ) *admet au moins une solution. On suppose aussi que K est sans point isolé.*

Soit $(\mu_k)_k$ une suite décroissante de réels strictement positifs et pour tout $k \in \mathbb{N}$, on note par x_k une solution de (PBμ) *pour $\mu = \mu_k$. Tout point d'adhérence de $(x_k)_k$ est solution de* (P1)*.*

Démonstration. Soit $\bar{x}$ un point d'adhérence de $(x_k)_k$. Pour ne pas charger les notations, on note encore par $(x_k)_k$ la sous-suite convergeante vers $\bar{x}$.

Par continuité de $x \to Ax$, on a $b = Ax_k \to A\bar{x}$ quand $k \to +\infty$. En particulier, $A\bar{x} = b$. De même, par continuité de h_i pour $i \in \mathcal{I}$, on a $h_i(\bar{x}) \leq 0$ pour tout $i \in \mathcal{I}$. On a donc $\bar{x} \in K$.

Si pour tout $i \in \mathcal{I}$, $h_i(\bar{x}) < 0$ alors $\mu_k h_i(x_k) \to 0$ (car $(\text{bar}(x_k))_k$ est une suite bornée) quand $k \to +\infty$ et ceci pour tout $i \in \mathcal{I}$. Si $\bar{x}$ est sur le bord de K alors par construction $\text{bar}(x_k) \to +\infty$ quand $k \to +\infty$. Dans les deux cas, on a $\liminf_k \mu_k \text{bar}(x_k) \geq 0$. On en déduit que

$$\liminf_k \big(f(x_k) + \mu_k \text{bar}(x_k)\big) = f(\bar{x}) + \liminf_k \mu_k \text{bar}(x_k) \geq f(\bar{x}).$$

Par définition de l'optimalité de x_k, on a pour tout $x \in U$ tel que $Ax = b$ et $\text{bar}(x) < \infty$

$$f(x) + \mu_k \text{bar}(x) \geq f(x_k) + \mu_k \text{bar}(x_k)$$

et en passant à la limite inférieure pour $k \to \infty$, on a $f(x) \geq f(\bar{x})$. Ceci étant vrai pour tout $x \in \overset{\circ}{K}$, on en déduit, par continuité de f et comme K est sans point isolé, que pour tout $x \in K$, $f(x) \geq f(\hat{x})$. Donc $\hat{x}$ est bien solution au problème (P1). ∎

Un algorithme de barrière consiste à résoudre une suite de problème (PBμ) pour $\mu = \mu_k$ où $\mu_k \downarrow 0$ et à utiliser cette suite de solution comme une approximation de la solution du problème initial (P1).

Concrètement, on se fixe un nombre fini de problèmes du type (PBμ) pour $\mu = \mu_k$ où $\mu_k \downarrow 0$ et $k \in \{1, \dots, k_{max}\}$. La dernière solution $x_{k_{max}}$ est utilisée comme solution approchante de (P1).

On peut alors se demander pourquoi on ne résout pas seulement le dernier problème (PBμ) pour $\mu = \mu_{k_{max}}$? Cela est dû au fait que la Hessienne de $B(\cdot, \mu)$ varie très rapidement en s'approchant du bord de K. La méthode de Newton qu'on utilise pour résoudre (PBμ) est donc très instable : un choix légèrement différent du point initial dans la méthode de Newton produira de grands changements dans la suite des itérations de l'algorithme. On va alors résoudre une suite de problèmes (PBμ) en décroissant le paramètre μ à chaque itération et utiliser la solution du dernier problème comme point initial du problème suivant.

Les solutions $x^*(\mu)$ de (PBμ) joue un rôle central pour les méthodes de barrière. L'ensemble de ces points décrit un "chemin" dans l'intérieur des contraintes.

Définition 3.10. *On suppose que pour tout $\mu > 0$, le problème (PBμ) admet une unique solution notée $x^*(\mu)$. L'ensemble $\{x^*(\mu) : \mu > 0\}$ est appelé* ***central path****.*