Optimisation non linéaire

Cours et exercices

Guillaume Laurent

La mise en page a été réalisée avec le style *The Legrand Orange Book* de Mathias Legrand disponible à l'adresse https://www.latextemplates.com/template/legrand-orange-book et les images de https://www.freepik.com/

Table des matières

Table of contents

1. Introduction

Trop souvent, le terme optimisation est utilisé à mauvais escient comme un synonyme d'amélioration. Mais l'optimisation ne vise pas à améliorer mais à trouver LA meilleure solution à un problème à l'aide de méthodes analytiques ou numériques. Optimiser requiert de définir précisément trois choses : la fonction objectif, les variables de décision et le domaine de recherche dans lequel les variables peuvent évoluer. Trouver un optimum est en général un problème difficile. Il n'existe pas de méthode d'optimisation parfaite résolvant tous les problèmes. En fonction du problème, on cherchera donc la méthode la mieux adaptée.

Ce cours aborde les problèmes d'optimisation continus non linéaires, c'est-à-dire dont la fonction objectif et les variables de décision évoluent de façon continus comme c'est le cas de beaucoup de systèmes physiques et de modèles mathématiques. Il présente les méthodes les plus courantes de manière synthétiques. On pourra s'aider de l'organigramme présenté en figure 1.1 pour sélectionner la méthode en fonction des caractéristiques de la fonction objectif. Étant donné que la maximisation de f est équivalente à la minimisation de $-f$, tous les algorithmes présentés ici recherchent un minimiseur de la fonction objectif.

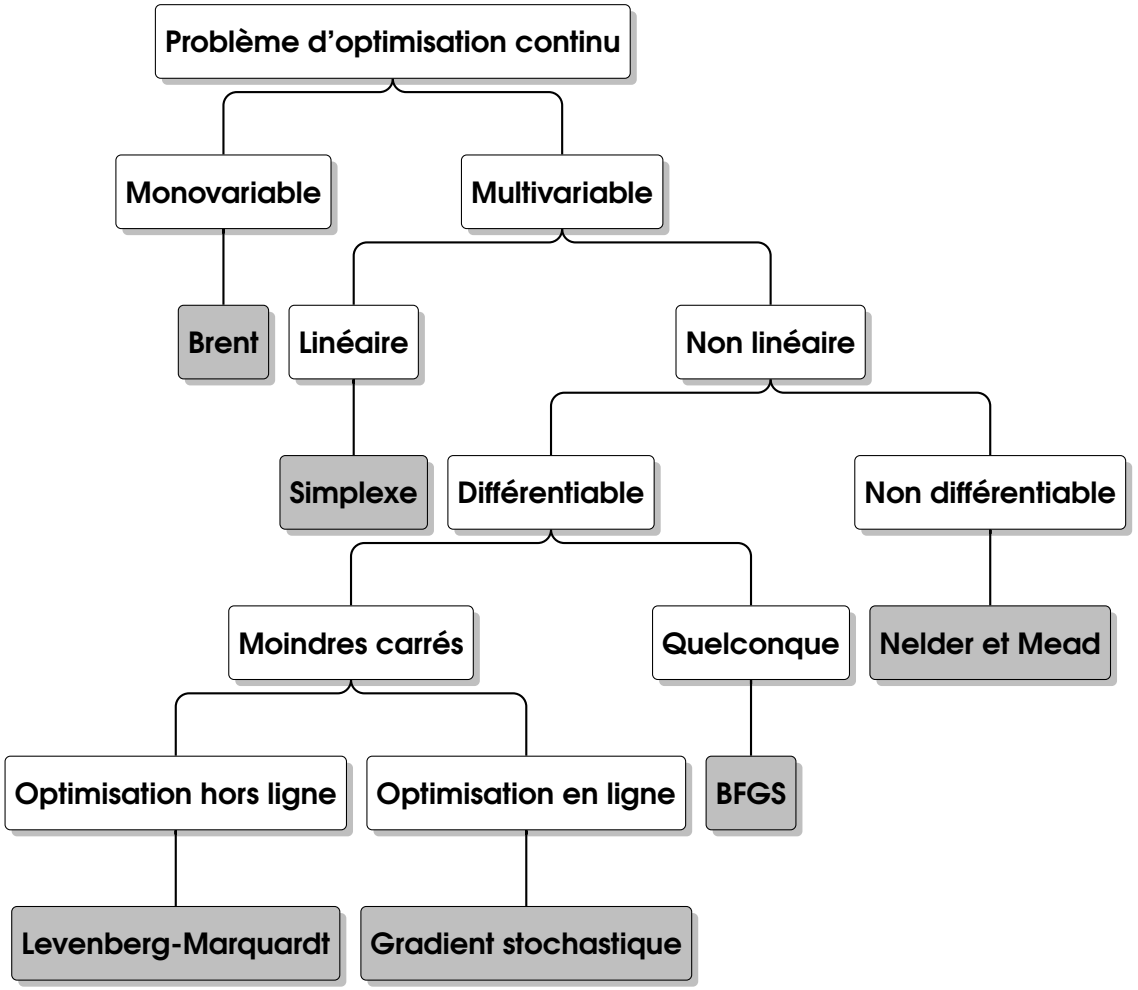


FIGURE 1.1 – Organigramme de choix des méthodes d'optimisation continue.

2. Minimisation de fonctions monovariables

On s'intéresse dans cette partie à la minimisation d'une fonction continue scalaire à une seule variable (fonction de $\mathbb{R}$ dans $\mathbb{R}$). La maximisation d'une fonction peut être réalisée par les mêmes méthodes en considérant l'opposée de la fonction objectif pour revenir à un problème de minimisation.
L'intérêt des algorithmes d'optimisation unidimensionnelle ne vient pas seulement du fait que dans les applications on rencontre naturellement des problèmes monovariables, mais aussi du fait que l'optimisation unidimensionnelle est un composant fondamental de bon nombre de méthodes d'optimisation de fonctions multivariables.

2.1 Définitions

Dans la suite, la fonction objectif f est supposée continue et monovariable. x désigne la variable de décision. Le domaine de recherche I est un intervalle fermé borné de $\mathbb{R}$. Le problème de minimisation consiste à trouver x^* tel que :

$$\forall x \in I,\ f(x) \geq f(x^*) \tag{2.1}$$

Definition 2.1 — minimum global. Soit $f : I \mapsto \mathbb{R}$ une fonction. $f(x^*)$ est le minimum global de f si et seulement si :

$$\forall x \in I,\ f(x) \geq f(x^*) \tag{2.2}$$

x^* est un minimiseur global de f.

Si la fonction objectif n'est pas convexe, il est possible qu'elle admette plusieurs minima locaux. Dans ce cas, il peut être difficile de trouver un minimum global. A défaut, on cherchera alors des méthodes qui convergent vers un minimum local.

Definition 2.2 — minimum local. Soit $f : I \mapsto \mathbb{R}$ une fonction. $f(x^*)$ est un minimum local de f si et seulement si :

$$\exists \varepsilon,\ \forall x \in]x^* - \varepsilon; x^* + \varepsilon[,\ f(x) \geq f(x^*) \tag{2.3}$$

Il est également possible de définir un minimum local en utilisant la pente et la courbure de la fonction.

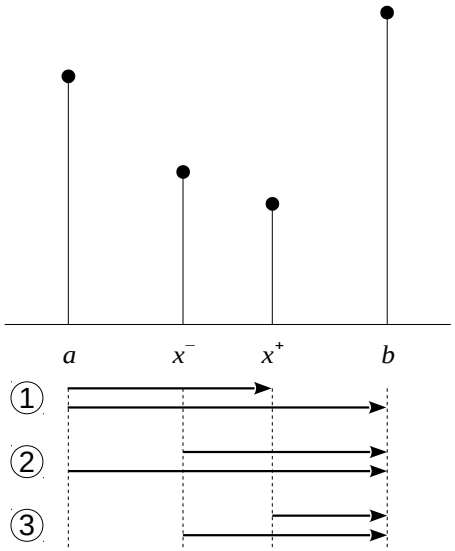


FIGURE 2.1 – Conservation des proportions dans la méthode de la section d'or.

Theorem 2.1 — condition suffisante de minimalité locale. Soit $f : I \mapsto \mathbb{R}$ une fonction $\mathscr{C}^2$. Si x^* vérifie les conditions :

$$\begin{cases} f'(x^*) = 0 \\ f''(x^*) > 0 \end{cases} \tag{2.4}$$

alors, $f(x^*)$ est un minimum local strict de f.

2.2 Méthode de la section d'or

Pour une fonction f continue, si l'on trouve trois points $a < x < b$ tels que $f(x) < f(a)$ et $f(x) < f(b)$, alors un minimum de f est atteint dans l'intervalle $]a,b[$. Des méthodes de minimisation exploitent cette idée en encadrant de plus en plus précisément le minimiseur. En effet, en prenant quatre points tels que $a < x^- < x^+ < b$, si $f(x^-) < f(a)$ et $f(x^-) < f(x^+)$ alors il y a un minimum dans $]a;x^+[$. Symétriquement, si $f(x^+) < f(b)$ et $f(x^+) < f(x^-)$ alors il y a un minimum dans $]x^-,b[$.

Il y a différentes possibilités pour le choix des points x^- et x^+. La méthode la plus utilisée est la méthode de la section d'or. L'idée est de conserver d'une itération à l'autre les mêmes proportions entre les points afin de les réutiliser comme l'illustre la figure 2.1. D'un point de vue plus formel, on choisit x^+ tel que :

$$\frac{(x^+ - a)}{(b-a)} = \rho \tag{2.5}$$

et x^- tel que :

$$\frac{(b - x^-)}{(b-a)} = \rho \tag{2.6}$$

avec $0,5 < \rho < 1$.

Données : f une fonction continue sur $[a;b]$ et $x \in [a;b]$ tels que $f(x) < f(a)$ et $f(x) < f(b)$
et $\rho = \frac{\sqrt{5}-1}{2}$
tant que $b-a > \varepsilon$ **faire**
  $x^+ \leftarrow a+\rho(b-a)$
  $x^- \leftarrow b-\rho(b-a)$
  si $f(x^-) < f(x^+)$ **alors**
    $b \leftarrow x^+$
    $x \leftarrow x^-$
  sinon
    $a \leftarrow x^-$
    $x \leftarrow x^+$
  fin
fin
Résultat : x

Algorithme 1 : Algorithme de la section d'or

A l'itération suivante, si $f(x^+) < f(b)$ et $f(x^+) < f(x^-)$, le minimum est à droite et a doit prendre la valeur de x^-. On impose alors que x^+ devienne le nouveau x^-. Comme la proportion doit être conservé, on peut écrire :

$$\frac{(b-x^+)}{(b-x^-)} = \rho \tag{2.7}$$

En simplifiant les trois équations, il vient :

$$\rho^2 + \rho - 1 = 0 \tag{2.8}$$

Comme $0,5 < \rho < 1$, on a :

$$\rho = \frac{\sqrt{5}-1}{2} \approx 0.61803 \tag{2.9}$$

ρ est l'inverse du nombre d'or $\varphi = (1+\sqrt{5})/2$ d'où le nom de la méthode.

L'algorithme 1 présente une version naïve de la méthode de la section d'or effectuant deux évaluations de la fonction à chaque itération. Il est possible moyennant quelques permutation d'écrire un algorithme n'effectuant qu'une évaluation de la fonction à chaque itération.

La convergence de cette méthode est linéaire avec le taux ρ, ce qui est relativement lent. Le principal avantage de cette méthode est qu'elle ne nécessite pas de calcul complexe et peut être effectuée à la main.

2.3 Méthode de Newton

Il est également possible d'utiliser la méthode de Newton pour rechercher un minimiseur. Le principe est de calculer une approximation parabolique p de f autour d'un point x_k par son développement de Taylor du second ordre :

$$p(x) = f(x_k) + f'(x_k)(x-x_k) + \frac{1}{2}f''(x_k)(x-x_k)^2 \tag{2.10}$$

La dérivée de $p(x)$ est naturellement :

$$\frac{dp(x)}{dx} = f'(x_k) + f''(x_k)(x-x_k) \tag{2.11}$$

On calcule alors un nouveau point, x_{k+1}, qui minimise p et donc qui annule sa dérivée :

$$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)} \tag{2.12}$$

Si le point x_0 est assez proche d'un minimiseur local de f, alors on montre que la suite x_k converge de façon quadratique ($|x_{k+1} - x^*| \leq \beta |x_k - x^*|^2$).

D'un point de vue pratique, la méthode de Newton est efficace proche d'un minimum mais souffre de nombreux inconvénients. En effet, elle n'est applicable que si $f''(x_k) > 0$ (la parabole est tournée vers le haut). De plus, même si $f''(x_k) > 0$, la méthode peut diverger si le point de départ est trop éloigné de la solution.
Cette méthode est donc peu utilisée sous cette forme. Son intérêt majeur est d'illustrer dans $\mathbb{R}$ le fonctionnement des algorithmes de minimisation multivariables.

2.4 Méthode de Brent

Plutôt que de calculer l'équation de la parabole à l'aide de la dérivée et de la dérivée seconde, il est préférable d'utiliser trois points $a < x < b$ tels que $f(x) < f(a)$ et $f(x) < f(b)$.
On calcule alors la position x' du minimum de la parabole. Tous calculs faits, on montre que :

$$x' = b - \frac{1}{2} \frac{(b-a)^2[f(b)-f(x)] - (b-x)^2[f(b)-f(a)]}{(b-a)[f(b)-f(x)] - (b-x)[f(b)-f(a)]} \tag{2.13}$$

Cette méthode est connue sous le nom d'interpolation quadratique inverse.

L'interpolation quadratique inverse converge en une itération si la fonction objectif est quadratique. Comme la méthode de Newton, elle converge très rapidement si x est suffisamment proche d'un minimum. Dans la pratique, on utilise rarement cette méthode seule car elle trop instable.

En 1973, Brent a proposé une méthode conciliant la robustesse de la méthode de la section d'or avec la vitesse de l'interpolation quadratique inverse. L'idée est d'utiliser l'interpolation quadratique inverse quand sa solution est acceptable et la méthode de la section d'or sinon. L'algorithme nécessite pas mal d'astuces pour imbriquer les deux méthodes [3, 19].

2.5 En pratique

La méthode de Brent est devenue la méthode de référence pour la minimisation de fonctions continues monovariables. Elle est notamment implantée dans la plupart des langages (C, Python, Mathematica, Maple, Matlab, etc.). Les conditions d'arrêts sont généralement le nombre d'itérations et la taille de l'intervalle.

On peut se demander avec quelle précision ultime peut-on espérer approcher un minimiseur ? On pourrait penser que si ε est la précision de la machine et x^* un minimiseur, on pourra obtenir une valeur entre $(1-\varepsilon)x^*$ et $(1+\varepsilon)x^*$. En réalité, cela est généralement impossible.
Au voisinage de x^*, la fonction objectif peut être estimée par son développement de Taylor au second ordre, sachant que $f'(x^*) = 0$, c'est-à-dire :

$$f(x) = f(x^*) + \frac{1}{2} f''(x^*)(x-x^*)^2 + o((x-x^*)^2) \tag{2.14}$$

Avec la méthode de la section d'or, il faut comparer les valeurs de la fonction. Si $|x-x^*|$ est de l'ordre de $o(\alpha)$ alors $|f(x)-f(x^*)|$ est de l'ordre de $o(\alpha^2)$ (en supposant que $f''(x^*)$ est de l'ordre de 1). Donc, si ε est la précision de la machine, alors on pourra obtenir une précision sur x de l'ordre de $o(\sqrt{\varepsilon})$.

3. Minimisation de fonctions multivariables

On s'intéresse maintenant à l'optimisation de fonctions scalaires de plusieurs variables (fonctions de $\mathbb{R}^n$ dans $\mathbb{R}$).

3.1 Définitions

Comme dans le cas monovariable, on peut définir les notions de pentes, courbures, points critiques et une condition suffisante de minimalité locale à partir de la « dérivée première », le gradient, et de la « dérivée seconde », la matrice hessienne.

Definition 3.1 — gradient. Soit $f : \mathbb{R}^n \mapsto \mathbb{R}$ une fonction différentiable. La fonction vectorielle notée $\nabla f(X) : \mathbb{R}^n \mapsto \mathbb{R}^n$ est appelée le *gradient* de f et est définie par :

$$\nabla f(X) = \begin{pmatrix} \frac{\partial f(X)}{\partial x_1} \\ \vdots \\ \frac{\partial f(X)}{\partial x_n} \end{pmatrix} \tag{3.1}$$

Pour une fonction multivariable, la notion de pente n'a de sens qu'en fixant une direction pour la calculer. Par exemple, en topographie, les lignes de niveau représentent les directions selon lesquelles la pente est nulle. Le gradient indique la direction qui a la pente la plus forte (montée). Cette direction est normale à la ligne de niveau passant en un point.

Definition 3.2 — pente. Soit $f : \mathbb{R}^n \mapsto \mathbb{R}$ une fonction différentiable. Soient $X, D \in \mathbb{R}^n$. La quantité :

$$\frac{D^{\mathrm{T}} \nabla f(X)}{\|D\|} \tag{3.2}$$

représente la *pente* de la fonction f en X dans la direction D.

Comme dans le cas monovariable, les points critiques correspondent aux zéros de la dérivée première et donc aux zéros du gradient.

Definition 3.3 — point critique. Soit $f : \mathbb{R}^n \mapsto \mathbb{R}$ une fonction différentiable. Tout vecteur $X \in \mathbb{R}^n$ tel que $\nabla f(X) = \mathbf{0}$ est appelé *point critique* ou *point stationnaire* de f.

Un point critique peut correspondre à un maximum de la fonction, à un minimum ou à un point selle. Pour distinguer ces différentes possibilités, il faut utiliser la dérivée seconde qui en multivariable

correspond à la matrice hessienne.

Definition 3.4 — hessien. Soit $f : \mathbb{R}^n \mapsto \mathbb{R}$ une fonction deux fois différentiable. La fonction notée $\nabla^2 f(X) : \mathbb{R}^n \mapsto \mathbb{R}^{n\times n}$ est appelée *matrice hessienne* ou *hessien* de f et est définie par :

$$\nabla^2 f(X) = \begin{pmatrix} \frac{\partial^2 f(X)}{\partial x_1^2} & \frac{\partial^2 f(X)}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f(X)}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f(X)}{\partial x_2 \partial x_1} & \frac{\partial^2 f(X)}{\partial x_2^2} & \cdots & \frac{\partial^2 f(X)}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f(X)}{\partial x_n \partial x_1} & \frac{\partial^2 f(X)}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f(X)}{\partial x_n^2} \end{pmatrix} \tag{3.3}$$

La matrice hessienne est symétrique par définition. Elle permet notamment de calculer la courbure qui comme la pente est calculée dans un direction donnée.

Definition 3.5 — courbure. Soit $f : \mathbb{R}^n \mapsto \mathbb{R}$ une fonction deux fois différentiable. Soient $X, D \in \mathbb{R}^n$. La quantité :

$$\frac{D^{\mathrm{T}} \nabla^2 f(X) D}{\|D\|^2} \tag{3.4}$$

représente la *courbure* de la fonction f en X dans la direction D.

La matrice hessienne permet également de savoir si un point critique correspond à un minimum de la fonction.

Theorem 3.1 — condition suffisante de minimalité locale. Soit $f : \mathbb{R}^n \mapsto \mathbb{R}$ une fonction deux fois différentiable dans un sous-ensemble ouvert V de $\mathbb{R}^n$. Si $X^* \in V$ vérifie les conditions :

$$\begin{cases} \nabla f(X^*) = 0 \\ \nabla^2 f(X^*) \text{ est définie positive (i.e. } \forall D \in \mathbb{R}^n \; D^{\mathrm{T}} \nabla^2 f(X^*) D > 0) \end{cases} \tag{3.5}$$

Alors, X^* est un minimiseur local strict de f.

Autrement dit, un minimiseur local strict de f est un point où la pente est nulle et la courbure positive dans toutes les directions.

3.2 Méthodes à directions de descente

Pour minimiser une fonction, il est naturel de chercher une direction dans le domaine de recherche qui la réduise progressivement. On appelle direction de descente, une direction dans laquelle la pente est négative. Si l'on avance dans une direction de descente la fonction va diminuer (au moins au début !).

Definition 3.6 — direction de descente. Soit $f : \mathbb{R}^n \mapsto \mathbb{R}$ une fonction différentiable. Soient $X, D \in \mathbb{R}^n$. La direction D est une *direction de descente* en X si

$$D^{\mathrm{T}} \nabla f(X) < 0. \tag{3.6}$$

Theorem 3.2 — descente. Soit $f : \mathbb{R}^n \mapsto \mathbb{R}$ une fonction différentiable. Soient $X \in \mathbb{R}^n$ tel que $f(X) \neq 0$ et $D \in \mathbb{R}^n$. Si D est une direction de descente en X alors il existe η tel que :

$$\forall \alpha \in [0; \eta[, \qquad f(X + \alpha D) < f(X) \tag{3.7}$$

Les algorithmes de descente exploitent ce résultat (cf. algorithme 2). Le principe consiste à chaque

Données : f une fonction différentiable et X_0 un point initial
$k \leftarrow 0$
tant que *condition d'arrêt non satisfaite* **faire**
  Trouver une direction de descente D_k telle que ${D_k}^{\mathrm{T}} \nabla f(X_k) < 0$
  Trouver un pas α_k tel que $f(X_k + \alpha_k D_k) < f(X_k)$ (recherche en ligne)
  $X_{k+1} \leftarrow X_k + \alpha_k D_k$
  $k \leftarrow k+1$
fin

Algorithme 2 : Algorithme à directions de descente

itération à trouver une direction de descente puis d'avancer un peu dans cette direction pour réduire la fonction.

Nous détaillerons plus loin les différentes méthodes de choix de la direction de descente. Une fois la direction déterminée, il reste à calculer une longueur de pas. Si l'on choisit naïvement un pas fixe tel que :

$$\|\alpha_k D_k\| = \text{constante} \tag{3.8}$$

La précision de la recherche est alors limitée à $\|\alpha_k D_k\|$. S l'on choisit $\|\alpha_k D_k\|$ petit la convergence est lente.

Il faut donc trouver à chaque itération une valeur de pas adaptée. Cette étape est appelée *recherche en ligne* (*line search*).

Comme on cherche à minimiser f et que l'on dispose de méthodes efficaces pour minimiser une fonction monovariable, on peut chercher un pas α_k diminuant le plus f dans la direction d_k, c'est-à-dire :

$$\forall \alpha \in \mathbb{R}^+, \quad f(X_k + \alpha D_k) \geq f(X_k + \alpha_k D_k) \tag{3.9}$$

Pour ce faire, on applique une méthode de minimisation à la fonction $g(\alpha) = f(X_k + \alpha D_k)$ [5, 19].

D'autres méthodes consistent à trouver rapidement un pas « acceptable », c'est la notion de progrès suffisant. On cherche un pas α diminuant *suffisamment* la fonction $g(\alpha)$ par rapport à $g(0)$.
Pour déterminer si un pas est « acceptable », il existe des tests comme le test d'Armijo, le test de Wolfe ou celui de Goldstein. Pour chaque test, il existe un algorithme ayant une convergence linéaire permettant de trouver une valeur qui passe le test sous la condition naturelle que $g'(0) < 0$ [2, 12].

3.3 Méthode de la descente de gradient

L'idée la plus simple consiste à utiliser l'opposée du gradient comme direction de descente, soit :

$$\boxed{D_k = -\nabla f(X_k)} \tag{3.10}$$

En effet, la direction du gradient est celle dans laquelle la fonction a la plus forte pente. La direction opposée au gradient est donc celle dans laquelle la fonction a la plus forte descente.

Dans le cas général, sous des hypothèses assez faibles de régularité, la descente de gradient converge vers un point critique de la fonction objectif [12]. La méthode est résumée par l'algorithme 3.
Si cette méthode est simple à mettre en oeuvre, les résultats sont décevants car sa vitesse de convergence est relativement basse et très sensible à la « forme » de la fonction objectif. *Grosso*

Données : f une fonction différentiable et X_0 un point initial
$k \leftarrow 0$
tant que *condition d'arrêt non satisfaite* **faire**
  $D_k \leftarrow -\nabla f(X_k)$
  Trouver un pas α_k qui minimise $f(X_k + \alpha D_k)$ (recherche en ligne)
  $X_{k+1} \leftarrow X_k + \alpha_k D_k$
  $k \leftarrow k+1$
fin

Algorithme 3 : Algorithme de la descente de gradient avec recherche en ligne optimale.

modo, la vitesse de convergence est liée au conditionnement de la matrice hessienne. Si le hessien est mal conditionnée, la fonction objectif ressemble à une « vallée » étroite. La pente est beaucoup plus forte dans la direction orthogonale à la vallée. Dans ces conditions, la direction du gradient est presque orthogonale à la direction de la vallée pour des points même proches du fond. L'algorithme suit donc une trajectoire en zigzag qui converge très lentement vers un minimiseur.
De plus, même si l'étape de recherche en ligne est optimale, la méthode n'est pas plus efficace car, dans ce cas, comme la pente est nulle en X_{k+1} dans la direction D_k, le gradient en X_{k+1} est normal à D_k soit :

$$D_k{}^{\mathrm{T}} \nabla f(X_{k+1}) = 0 \tag{3.11}$$

L'algorithme suit donc une trajectoire en zigzag à angles droits.

3.4 Méthode de Newton

La méthode de Newton pour les fonctions multivariables est identique à celle des fonctions univariées. Comme précédemment, une estimation quadratique $p(X)$ de f au point X_k est donnée par le développement de Taylor du second ordre, qui s'écrit pour une fonction multivariée :

$$p(X) = f(X_k) + (X - X_k)^{\mathrm{T}} \nabla f(X_k) + \frac{1}{2}(X - X_k)^{\mathrm{T}} [\nabla^2 f(X_k)](X - X_k) \tag{3.12}$$

Si la matrice hessienne est inversible, on choisit X_{k+1} qui minimise p, soit :

$$X_{k+1} = X_k - \left[\nabla^2 f(X_k)\right]^{-1} \nabla f(X_k) \tag{3.13}$$

La direction de Newton est donc :

$$\boxed{D_k = -\left[\nabla^2 f(X_k)\right]^{-1} \nabla f(X_k)} \tag{3.14}$$

Pour des raisons d'efficacité des calculs, la direction de descente D_k est calculée sans inverser $\left[\nabla^2 f(X_k)\right]$ mais en résolvant :

$$\nabla^2 f(X_k).D_k = -\nabla f(X_k) \tag{3.15}$$

L'intérêt de cette méthode est sa convergence quadratique vers un minimiseur local à la condition que X_0 soit assez proche d'un minimiseur. Néanmoins d'un point de vue pratique, cette méthode comporte les mêmes inconvénients que dans le cas monovariable. La direction de Newton n'est définie que si la matrice hessienne est strictement définie positive ($\nabla^2 f(X_k) > 0$) car dans ce cas la paraboloïde est tournée vers le haut et admet un minimum. Dans le cas contraire, on peut toujours suivre la direction de la plus forte descente. On effectue ensuite une recherche en ligne dans la direction choisie afin de « robustifer » la méthode comme l'illustre l'algorithme 4.

Données : f une fonction différentiable et X_0 un point initial
$k \leftarrow 0$
tant que *condition d'arrêt non satisfaite* **faire**
  si $\nabla^2 f(X_k) > 0$ **alors**
    Calculer D_k solution de $\nabla^2 f(X_k) D = -\nabla f(X_k)$
  sinon
    $D_k \leftarrow -\nabla f(X_k)$
  fin
  Trouver un pas α_k qui minimise $f(X_k + \alpha D_k)$ (recherche en ligne)
  $X_{k+1} \leftarrow X_k + \alpha_k D_k$
  $k \leftarrow k+1$
fin

Algorithme 4 : Algorithme de Newton avec recherche en ligne optimale

3.5 Méthode quasi-Newton (DFP et BFGS)

A partir de la méthode de Newton, on peut en déduire des algorithme plus efficaces. En effet, le calcul direct du Hessien est trop coûteux. Une astuce permet de le calculer indirectement à partir de quelques évaluations du gradient.

Si on écrit le développement de Taylor du gradient à l'ordre 1 en X_{k-1}, on a :

$$\nabla f(X_k) \approx \nabla f(X_{k-1}) + \nabla^2 f(X_{k-1})(X_k - X_{k-1}) \tag{3.16}$$

On pose :

$$Y_k = \nabla f(X_k) - \nabla f(X_{k-1}) \tag{3.17}$$

On a par ailleurs de par la définition de l'algorithme de descente :

$$\bar{D}_{k-1} = X_k - X_{k-1} \tag{3.18}$$

Donc, si la fonction est quadratique, $\nabla^2 f$ est constante et est une solution de :

$$Y_k = \nabla^2 f . \bar{D}_{k-1} \tag{3.19}$$

A l'itération précédente, $\nabla^2 f$ était aussi solution de :

$$Y_{k-1} = \nabla^2 f . \bar{D}_{k-2} \tag{3.20}$$

Cette équation est également vrai pour toutes les itérations précédentes. Si les points X_k sont bien choisis alors il est possible d'écrire suffisamment d'équations pour identifier les inconnues de la matrice hessienne.

Naturellement, cette astuce ne fonctionne que pour une fonction quadratique. Dans le cas général, on utilise une idée similaire pour définir des suites B_k qui vont converger vers $\left(\nabla^2 f(X_k)\right)^{-1}$ en fonction de B_{k-1}, Y_{k-1} et D_{k-1}. On trouve notamment deux méthodes de calcul de cette suite :

— la formule de Davidon-Fletcher-Powell (DFP) [7, 9] :

$$B_{k+1} = B_k - \frac{B_k Y_k {Y_k}^{\mathrm{T}} B_k}{{Y_k}^{\mathrm{T}} B_k Y_k} + \frac{\bar{D}_k \bar{D}_k^{\mathrm{T}}}{\bar{D}_k^{\mathrm{T}} Y_k} \tag{3.21}$$

Données : f une fonction différentiable et X_0 un point initial
$k \leftarrow 0$
$B_0 \leftarrow I$
tant que *condition d'arrêt non satisfaite* **faire**
  $D_k \leftarrow -B_k \nabla f(X_k)$
  Trouver un pas α_k qui minimise $f(X_k + \alpha D_k)$ (recherche en ligne)
  $X_{k+1} \leftarrow X_k + \alpha_k D_k$
  $\bar{D}_k \leftarrow X_{k+1} - X_k$
  $Y_k \leftarrow \nabla f(X_{k+1}) - \nabla f(X_k)$
  $B_{k+1} \leftarrow B_k + \left(1 + \frac{Y_k{}^{\mathrm{T}} B_k Y_k}{Y_k{}^{\mathrm{T}} \bar{D}_k}\right) \frac{\bar{D}_k \bar{D}_k^{\mathrm{T}}}{Y_k{}^{\mathrm{T}} \bar{D}_k} - \frac{B_k Y_k \bar{D}_k^{\mathrm{T}} + \bar{D}_k Y_k{}^{\mathrm{T}} B_k}{Y_k{}^{\mathrm{T}} \bar{D}_k}$
  $k \leftarrow k+1$
fin

Algorithme 5 : Algorithme de quasi-Newton avec mise à jour BFGS

— la formule de Broyden-Fletcher-Goldfarb-Shanno (BFGS) [4, 8, 11, 20] :

$$B_{k+1} = B_k + \left(1 + \frac{Y_k{}^{\mathrm{T}} B_k Y_k}{Y_k{}^{\mathrm{T}} \bar{D}_k}\right) \frac{\bar{D}_k \bar{D}_k^{\mathrm{T}}}{Y_k{}^{\mathrm{T}} \bar{D}_k} - \frac{B_k Y_k \bar{D}_k^{\mathrm{T}} + \bar{D}_k Y_k{}^{\mathrm{T}} B_k}{Y_k{}^{\mathrm{T}} \bar{D}_k} \tag{3.22}$$

Cerise sur le gâteau, ces formules donnent toujours des matrices définies positives qui convergent vers l'inverse de la matrice hessienne si elle est définie positive et vers l'identité sinon. A l'aide de ces formules, on établit un algorithme à convergence très efficace et particulièrement robuste. La direction de descente est alors toujours définie par :

$$D_k = -B_k \nabla f(X_k) \tag{3.23}$$

Ainsi on prendra automatiquement la direction de Newton quand c'est possible et sinon on suivra la direction de la plus forte descente. Dans la pratique, on utilise la formule BFGS qui est plus efficace. L'unique inconvénient de cette méthode est la nécessité du stockage en mémoire d'une matrice de taille $n \times n$.

3.6 Méthode du gradient conjugué

La méthode du gradient conjugué est une méthode de descente à pas optimal permettant de minimiser une fonction quadratique de $\mathbb{R}^n$ dans $\mathbb{R}$ en au plus n itérations. [21].
Soit f une fonction quadratique :

$$f(X) = \frac{1}{2} X^{\mathrm{T}} Q X - B^{\mathrm{T}} X \tag{3.24}$$

avec Q une matrice définie positive. On a :

$$\nabla f(X) = (QX - B)^{\mathrm{T}} \qquad \nabla^2 f(x) = Q \tag{3.25}$$

On construit la suite :

$$X_{k+1} = X_k + \alpha_k D_k \tag{3.26}$$

avec α_k minimisant $f(X_k + \alpha D_k)$ (pas optimal).

La première idée de l'algorithme du gradient conjugué consiste à choisir chaque direction de descente conjuguée à la direction de descente précédente par rapport à Q.

Definition 3.7 — directions conjuguées. Soit une matrice $Q \in \mathbb{R}^{n\times n}$ définie positive. Deux vecteurs U et V de $\mathbb{R}^n$ sont *conjugués* par rapport à Q si $U^{\mathrm{T}}QV = 0$. Noter que, si Q est la matrice identité, les directions conjuguées sont orthogonales.

En dimension $n > 2$, il existe une infinité de directions conjuguées à une direction donnée. Pour trouver à l'itération k, une direction D_k conjuguée à D_{k-1}, la seconde idée consiste à chercher D_k sous forme d'une combinaison linéaire de D_{k-1} et de la direction de la plus forte descente au point X_k, soit :

$$D_k = -\nabla f(X_k) + s_k D_{k-1} \tag{3.27}$$

en choisissant le coefficient s_k tel que les directions successives soient conjuguées par rapport à Q :

$${D_{k-1}}^{\mathrm{T}} Q D_k = 0 \tag{3.28}$$

c'est-à-dire :

$$-{D_{k-1}}^{\mathrm{T}} Q \nabla f(X_k) + s_k {D_{k-1}}^{\mathrm{T}} Q D_{k-1} = 0 \tag{3.29}$$

Ceci est toujours possible si D_{k-1} est non nul. On montre alors que le réel s_k se calcule par la formule suivante (Fletcher-Reeves) :

$$s_k = \frac{\|\nabla f(X_k)\|^2}{\|\nabla f(X_{k-1})\|^2} \tag{3.30}$$

L'équation de récurrence permettant de calculer une nouvelle direction est donc :

$$D_k = -\nabla f(X_k) + \frac{\|\nabla f(X_k)\|^2}{\|\nabla f(X_{k-1})\|^2} D_{k-1} \tag{3.31}$$

Par définition, cette direction est toujours une direction de descente puisque :

$$\nabla f(X_k)^{\mathrm{T}} D_k = -\|\nabla f(X_k)\| + s_k \nabla f(X_k)^{\mathrm{T}} D_{k-1} = -\|\nabla f(X_k)\| \tag{3.32}$$

A chaque itération, le pas est déterminé comme dans tout algorithme de descente par une recherche en ligne. On obtient ainsi un algorithme de minimisation appelé *algorithme de Fletcher-Reeves* (cf. algorithme 6) [10].
Lorsque la fonction objectif est quadratique, cet algorithme génère une suite de directions de recherche deux à deux conjuguées qui converge vers le minimiseur en au plus n itérations quel que soit x_0 !
Dans les autres cas, on peut raisonnablement penser que dans un voisinage de X^*, la fonction objectif n'est pas très différente d'une fonction quadratique et qu'on peut donc appliquer cette méthode. L'efficacité de l'algorithme repose essentiellement sur deux points : la recherche en ligne doit être exacte, les relations de conjugaisons doivent être précises.

La recherche en ligne peut être facilement rendu assez précise. En revanche, si la fonction objectif n'est pas une fonction quadratique, la notion de conjugaison relative n'a pas de signification. Ainsi, il peut être judicieux de réinitialiser régulièrement la direction courante D par la direction de la plus grande descente $-\nabla f(X)$ (tous les m itérations avec m de l'ordre de n).
Par rapport aux algorithmes de quasi-Newton, l'algorithme du gradient conjugué ne nécessite le stockage que de deux vecteurs de taille n. Il est donc particulièrement adapté aux problèmes de grandes dimensions.
Un variante a été proposée par Polak et Ribière en 1969. Le principe de l'algorithme est identique, seule la mise à jour de la direction est légèrement modifiée :

$$D_k = -\nabla f(X_k) + \frac{(\nabla f(X_k) - \nabla f(X_{k-1}))^{\mathrm{T}} \nabla f(X_k)}{\|\nabla f(X_{k-1})\|^2} D_{k-1} \tag{3.33}$$

Cette variante est réputée plus performante dans certaines applications.

Données : f une fonction différentiable et X_0 un point initial
$k \leftarrow 0$
$D_0 \leftarrow -\nabla f(X_0)$
$s_0 \leftarrow \|\nabla f(X_0)\|^2$
tant que *condition d'arrêt non satisfaite* **faire**
  Trouver un pas α_k qui minimise $f(X_k + \alpha D_k)$
  $X_{k+1} \leftarrow X_k + \alpha_k D_k$
  $s_{k+1} \leftarrow \|\nabla f(X_{k+1})\|^2$
  $D_{k+1} \leftarrow -\nabla f(X_{k+1}) + \frac{s_{k+1}}{s_k} D_k$
  $k \leftarrow k+1$
fin

Algorithme 6 : Algorithme du gradient conjugué (Fletcher-Reeves)

3.7 Approximation du gradient par différences finies

Les algorithmes à directions de descente nécessitent l'évaluation du gradient à chaque itération. Si la fonction est connue, il est préférable de le calculer analytiquement. Dans les autres cas, on utilisera une approximation numérique par différences finies.

Pour une fonction monovariable, la dérivée peut être approximée par :

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h} \tag{3.34}$$

avec h une constante, qu'on appelle le pas. Cette approximation est qualifiée de *différences finies centrées*.
On peut également calculer une approximation par *différences finies avants* :

$$f'(x) \approx \frac{f(x+h) - f(x)}{h} \tag{3.35}$$

Ou par *différences finies arrières*.

Il est possible de calculer des différences finies dans le cas de fonctions à plusieurs variables ; elles sont l'analogue discret des dérivées partielles et donc des composantes du gradient.
Pour une fonction à deux variables, on a :

$$\frac{\partial}{\partial x} f(x,y) \approx \frac{f(x+h,y) - f(x-h,y)}{2h} \tag{3.36}$$

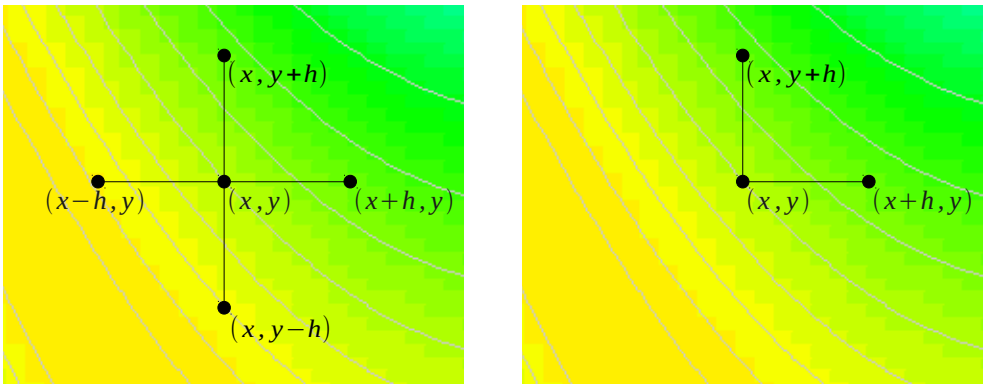
$$\frac{\partial}{\partial y} f(x,y) \approx \frac{f(x,y+h) - f(x,y-h)}{2h} \tag{3.37}$$

ou encore, dans le cas de différences finies avants :

$$\frac{\partial}{\partial x} f(x,y) \approx \frac{f(x+h,y) - f(x,y)}{h} \tag{3.38}$$

$$\frac{\partial}{\partial y} f(x,y) \approx \frac{f(x,y+h) - f(x,y)}{h} \tag{3.39}$$

La généralisation à une fonction à n variables est directe. L'estimation du gradient par différences finies centrées requiert $2n+1$ évaluations de la fonction. L'estimation du gradient par différences finies avant requiert $n+1$ évaluations de la fonction. Pour des raisons d'efficacité, on utilise généralement les différences finies avant même si les différences centrées donnent des approximations plus précises.



(a) Différences finies centrées (b) Différences finies avant

FIGURE 3.1 – Approximation du gradient d'une fonction à deux variables par différences finies.

3.8 Méthode de Nelder et Mead

La méthode de Nelder et Mead [18, 13] souvent appelée « méthode du simplexe » [1] est une méthode heuristique qui n'utilise pas de direction de descente et donc ne nécessite pas d'évaluer le gradient.

Definition 3.8 — simplexe. Un simplexe de dimension k est l'enveloppe convexe de $k+1$ vecteurs $X_1, \ldots, X_{k+1}$ de $\mathbb{R}^n$, $k = n$, affinement indépendants, c'est-à-dire que les k vecteurs $X_1 - X_{k+1}, X_2 - X_{k+1}, \ldots, X_k - X_{k+1}$ sont linéairement indépendants.

Par exemple, trois points non alignés dans $\mathbb{R}^2$, ou quatre points non coplanaires dans $\mathbb{R}^3$ sont affinement indépendants, et définissent des simplexes de dimension 2 et 3, respectivement.
Le principe de l'algorithme de Nelder et Mead est de faire évoluer un simplexe vers un minimiseur de la fonction objectif par des expansions ou des contractions successives du simplexe en fonction de la topologie locale (cf. algorithme 7 et figure 3.2).
Pour déterminer un simplexe initial autour d'un point X_0, on pose $X_{n+1} = X_0$ et on calcule pour tout i de 1 à n :

$$X_i = x_0 + \lambda E_i \tag{3.40}$$

où E_i est le ième vecteur unité et λ une constante d'échelle.
L'algorithme de Nelder et Mead est souvent moins efficace en terme de nombre d'évaluations de la fonction objectif mais fonctionne bien même si la fonction n'est pas différentiable ce qui permet d'obtenir une bonne solution pour des fonctions pas très lisses.
En terme d'occupation mémoire, l'algorithme nécessite une place équivalente à un algorithme de quasi-newton, c'est-à-dire $n+1$ vecteurs de dimensions n.
En revanche, ses performances de dégradent lorsque la dimension de la fonction objectif augmente. De plus, dans certaines configurations, le simplexe peut dégénérer, les points ne sont alors plus affinement indépendants. Une parade consiste, dans ce cas, à réinitialiser la méthode à partir du meilleur point obtenu pour continuer la descente.

3.9 En pratique

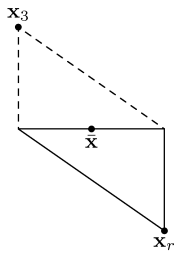
La méthode BFGS est la plus répandue car elle donne de bons résultats dans la plupart des cas. Pour les problèmes de très grandes dimensions, on peut utiliser la méthode du gradient conjugué. Enfin, si la fonction n'est pas différentiable voire discontinue, il est préférable d'utiliser la méthode de Nelder et Mead (en petites dimensions seulement).

1. à ne pas confondre avec l'algorithme du simplexe en programmation linéaire.

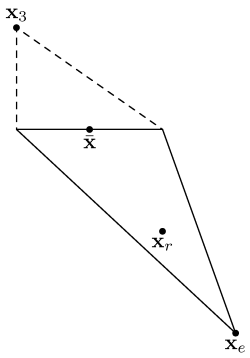
Tous les algorithmes progressent de façon itérative jusqu'à l'obtention d'un « bon » minimiseur. On utilise souvent plus d'une condition d'arrêt. Dans la plupart des logiciels, on trouvera la possibilité de spécifier :

- une condition sur le déplacement dans le domaine de recherche, si $\|X_{k+1} - X_k\|$ est très petit, c'est qu'on ne progresse plus beaucoup.
- une condition sur la progression de l'objectif, si $|f(X_{k+1}) - f(X_k)|$ est très petit, on peut être presque arrivé à un minimum,
- une condition sur la norme du gradient, si $\|\nabla f(X_k)\|$ est très petit alors on est proche d'un minimum,
- une condition sur le temps de calcul ou le nombre d'itérations.

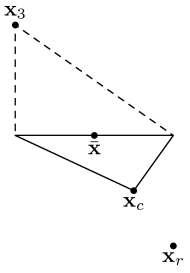
Enfin, il est très utile de tracer l'évolution de la fonction objectif et de la norme du gradient pour s'assurer qu'un minimum a bien été atteint. Naturellement, rien ne vaut une bonne connaissance de la fonction à optimiser, quand il faut décider des conditions d'arrêt.



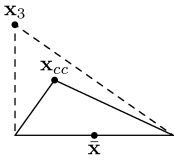
(a) Réflexion si $f(X_r) < f(X_1)$ et $f(X_r) \leq f(X_e)$ ou si $f(X_1) \leq f(X_r) < f(X_n)$



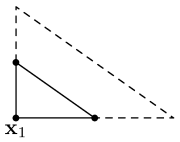
(b) Expansion si $f(X_e) < f(X_r) < f(X_1)$



(c) Contraction externe si $f(X_n) \leq f(X_r) < f(X_{n+1})$ et $f(X_c) < f(X_r)$



(d) Contraction interne si $f(X_{n+1}) \leq f(X_r)$ et $f(X_{cc}) < f(X_r)$



(e) Rétrécissement dans les autres cas

FIGURE 3.2 – Evolution du simplexe de Nelder et Mead (le simplexe original est en pointillé)

Données : f une fonction et $n+1$ points affinements indépendants $X_1, \ldots, X_{n+1}$ (simplexe)
tant que *condition d'arrêt non satisfaite* **faire**
  Renuméroter les points tels que : $f(X_1) \leq f(X_2) \leq \cdots \leq f(X_{n+1})$
  $\bar{X} \leftarrow \frac{1}{n} \sum_{i=1}^{n} X_i$
  $D \leftarrow \bar{X} - X_{n+1}$
  $X_r \leftarrow X_{n+1} + 2D$
  si $f(X_r) < f(X_1)$ **alors**
    $X_e \leftarrow X_{n+1} + 3D$
    si $f(X_e) < f(X_r)$ **alors**
      $X_{n+1} \leftarrow X_e$ `(expansion)`
    sinon
      $X_{n+1} \leftarrow X_r$ `(réflexion)`
    fin
  sinon
    si $f(X_1) \leq f(X_r) < f(X_n)$ **alors**
      $X_{n+1} \leftarrow X_r$ `(réflexion)`
    sinon
      si $f(X_n) \leq f(X_r) < f(X_{n+1})$ **alors**
        $X_c \leftarrow X_{n+1} + \frac{3}{2}D$
        si $f(X_c) < f(X_r)$ **alors**
          $X_{n+1} \leftarrow X_c$ `(contraction externe)`
        sinon
          pour i *de* 2 *à* $n+1$ **faire** $X_i \leftarrow X_i + \frac{1}{2}(X_i - X_1)$ `(rétrécissement)`
        fin
      sinon
        $X_{cc} \leftarrow X_{n+1} + \frac{1}{2}D$
        si $f(X_{cc}) < f(X_r)$ **alors**
          $X_{n+1} \leftarrow X_{cc}$ `(contraction interne)`
        sinon
          pour i *de* 2 *à* $n+1$ **faire** $X_i \leftarrow X_i + \frac{1}{2}(X_i - X_1)$ `(rétrécissement)`
        fin
      fin
    fin
  fin
fin
Résultat : x_1

Algorithme 7 : Algorithme de Nelder et Mead

4. Moindres carrés non linéaires

Cette partie est consacrée à la minimisation de fonctions scalaires de plusieurs variables pouvant s'écrire comme la somme m termes mis au carré, communément appelée moindres carrés.

4.1 Définitions

Les problèmes des moindres carrés non linéaires visent à minimiser des fonctions objectifs de la forme :

$$f(X) = \frac{1}{2}\sum_{i=1}^{m} g_i^2(X) \tag{4.1}$$

Les fonctions $g_i(X)$ sont appelées coûts partiels.

On utilise généralement la forme vectorielle $F : \mathbb{R}^n \mapsto \mathbb{R}^m$ de la fonction objectif définie par :

$$F(X) = \begin{pmatrix} g_1(X) \\ \vdots \\ g_m(X) \end{pmatrix} \tag{4.2}$$

On a donc :

$$f(X) = \frac{1}{2}F(X)^{\mathrm{T}}F(X) \tag{4.3}$$

Le gradient de f est :

$$\nabla f(X) = \sum_{i=1}^{m} g_i(X)\nabla g_i(X) = J_F(X)^{\mathrm{T}}F(X) \tag{4.4}$$

avec $J_F(X)$ la matrice jacobienne de F telle que :

$$J_F(X) = \begin{pmatrix} \frac{\partial g_1(X)}{\partial x_1} & \frac{\partial g_1(X)}{\partial x_2} & \cdots & \frac{\partial g_1(X)}{\partial x_n} \\ \frac{\partial g_2(X)}{\partial x_1} & \frac{\partial g_2(X)}{\partial x_2} & \cdots & \frac{\partial g_2(X)}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial g_m(X)}{\partial x_1} & \frac{\partial g_m(X)}{\partial x_2} & \cdots & \frac{\partial g_m(X)}{\partial x_n} \end{pmatrix} \tag{4.5}$$

Données : f une fonction différentiable telle que $f(X) = \frac{1}{2}F(X)^{\mathrm{T}}F(X)$ et X_0 un point initial
$k \leftarrow 0$
tant que *condition d'arrêt non satisfaite* **faire**
  $H_k \leftarrow J_F(X_k)^{\mathrm{T}}J_F(X)$
  si $H_k > 0$ **alors**
    Calculer D_k solution de $H_k.D = -J_F(X_k)^{\mathrm{T}}F(X_k)$
  sinon
    $D_k \leftarrow -J_F(X_k)^{\mathrm{T}}F(X_k)$
  fin
  Trouver un pas α_k qui minimise $f(X_k + \alpha D_k)$ (recherche en ligne)
  $X_{k+1} \leftarrow X_k + \alpha_k D_k$
  $k \leftarrow k+1$
fin

Algorithme 8 : Algorithme de Gauss-Newton avec recherche en ligne

Le hessien de f s'écrit :

$$\nabla^2 f(X) = \sum_{i=1}^{m} \nabla g_i(X) \nabla g_i(X)^{\mathrm{T}} + \sum_{i=1}^{m} g_i(X) \nabla^2 g_i(X) \tag{4.6}$$

$$= J_F(X)^{\mathrm{T}} J_F(X) + \sum_{i=1}^{m} g_i(X) \nabla^2 g_i(X) \tag{4.7}$$

4.2 Méthode de Gauss-Newton

Le deuxième terme de la matrice hessienne est très coûteux à calculer et est relativement faible au voisinage d'un minimum. Il peut être négligé en pratique et le hessien approché par le premier terme qui est une matrice semi définie positive :

$$\nabla^2 f(X_k) = J_F(X_k)^{\mathrm{T}} J_F(X_k) \tag{4.8}$$

La méthode qui consiste utiliser cette approximation dans le cadre de la méthode de Newton est nommée méthode de Gauss-Newton (cf. algorithme 8). La direction de Gauss-Newton est alors :

$$\boxed{D_k = -\left[J_F(X_k)^{\mathrm{T}} J_F(X_k)\right]^{-1} J_F(X)^{\mathrm{T}} F(X)} \tag{4.9}$$

En pratique, la direction de descente D_k est calculée en résolvant :

$$J_F(X_k)^{\mathrm{T}} J_F(X_k).D = -J_F(X)^{\mathrm{T}} F(X) \tag{4.10}$$

Contrairement à l'algorithme de Newton, l'algorithme de Gauss-Newton est un algorithme robuste car la matrice $J_F(X)^{\mathrm{T}} J_F(X)$ est presque toujours définie positive et ainsi la direction calculée est une direction de descente [14].

4.3 Méthode de Levenberg-Marquardt

Une approche alternative aux méthodes à directions de descente avec recherche en ligne repose sur la notion de région de confiance. Une région de confiance désigne un sous-ensemble du domaine de recherche (généralement une boule) dans lequel la fonction objectif peut être approximée par un modèle simple (généralement quadratique).

A chaque itération, on effectue un déplacement qui est obtenu en minimisant le modèle de la fonction dans la région de confiance. Si le modèle est une bonne approximation de la fonction sur la région de confiance alors la valeur de la fonction au nouveau point doit correspondre à la valeur prédite par le modèle. Dans ce cas, le nouveau point est acceptable et peut être utilisé pour l'itération suivante. De plus, comme l'approximation était bonne, on peut agrandir la région de confiance.
Dans le cas contraire, si la valeur de la fonction ne correspond pas du tout à la prédiction, on réduit la taille de la région de confiance et recommence le calcul en repartant du même point.

La méthode de Levenberg-Marquardt est une méthode à région de confiance qui utilise un modèle quadratique de la fonction objectif :

$$p(X) = f(X_k) + (X - X_k)^{\mathrm{T}} \nabla f(X_k) + \frac{1}{2}(X - X_k)^{\mathrm{T}} [\nabla^2 f(X_k)](X - X_k) \tag{4.11}$$

Si la fonction est bien quadratique, alors les g_i sont linéaires et $\nabla^2 g_i(X) = 0$. Ainsi, le deuxième terme de la matrice hessienne est nulle et le modèle s'écrit :

$$p(X) = f(X_k) + (X - X_k)^{\mathrm{T}} J_F(X_k)^{\mathrm{T}} F(X_k) + \frac{1}{2}(X - X_k)^{\mathrm{T}} J_F(X_k)^{\mathrm{T}} J_F(X_k)(X - X_k) \tag{4.12}$$

Pour calculer le point suivant X_{k+1}, on cherche à annuler le gradient de $p(X)$ comme dans la méthode de Newton ce qui revient à trouver un pas $D_k = X_{k+1} - X_k$ solution de :

$$J_F(X)^{\mathrm{T}} J_F(X) D_k = -J_F(X)^{\mathrm{T}} F(X) \tag{4.13}$$

Cependant la solution de cette équation n'est intéressante que si le modèle est une bonne aproximation de la fonction objectif. Si ce n'est pas le cas, la solution peut correspondre à un pas très grand pouvant provoquer la divergence de la méthode. L'idée de Levenberg est de « charger » la diagonale de la matrice hessienne pour réduire la taille du pas en fonction de la fidélité du modèle [15]. Le pas D_k est alors solution de :

$$\boxed{\left[J_F(X_k)^{\mathrm{T}} J_F(X_k) + \mu I\right] D_k = -J_F(X_k)^{\mathrm{T}} F(X_k)} \tag{4.14}$$

avec $\mu > 0$. Quand μ est grand, la direction D_k se rapproche de la plus forte descente (opposée du gradient). Inversement quand μ est petit, D_k correspond à la direction Gauss-Newton.

Comme nous l'avons vu au chapitre précédent, la direction de la plus forte descente n'est pas la meilleure dans les problèmes mal conditionnés (en forme de vallées étroites). Marquardt a proposé une amélioration de la méthode de Levenberg qui exploite la diagonale de la matrice hessienne [17]. Le pas D_k est alors définit comme solution de :

$$\boxed{\left[J_F(X_k)^{\mathrm{T}} J_F(X_k) + \mu \operatorname{diag} J_F(X_k)^{\mathrm{T}} J_F(X_k)\right] D_k = -J_F(X_k)^{\mathrm{T}} F(X_k)} \tag{4.15}$$

Dans les deux cas, μ est calculé à chaque itération et permet d'ajuster la taille de la région de confiance. En effet, si l'approximation quadratique est bonne alors μ doit être petit et D_k correspond à un grand pas dans la direction de Gauss-Newton. Si l'approximation est mauvaise alors μ doit être grand et D_k correspond à un petit pas dans la direction de la plus forte descente.
Il existe plusieurs heuristiques pour adapter μ en fonction de la qualité de l'approximation quadratique. On utilise généralement le rapport entre les évolutions de la fonction et du modèle, définit par :

$$\rho = \frac{f(X_k) - f(X_k + D_k)}{p(X_k) - p(X_k + D_k)} \tag{4.16}$$

Données : f une fonction différentiable telle que $f(X) = \frac{1}{2}F(X)^{\mathrm{T}}F(X)$ et X_0 un point initial
$k \leftarrow 0$
$\nu \leftarrow 2$
$G_0 \leftarrow J_F(X_0)^{\mathrm{T}}F(X_0)$
$H_0 \leftarrow J_F(X_0)^{\mathrm{T}}J_F(X_0)$
$\mu \leftarrow \tau * \max\{h_{ii}\}$
tant que *condition d'arrêt non satisfaite* **faire**
  Calculer D_k solution de $[H_k + \mu I]D = -G_k$
  $Y \leftarrow X_k + D_k$
  $\rho \leftarrow (F(X_k) - F(Y)) \,/\, \left(\frac{1}{2}D_k{}^{\mathrm{T}}(\mu D_k - G_k)\right)$
  si $\rho > 0$ **alors**
    $X_{k+1} \leftarrow Y$
    $H_{k+1} \leftarrow J_F(X_{k+1})^{\mathrm{T}}J_F(X_{k+1})$
    $G_{k+1} \leftarrow J_F(X_{k+1})^{\mathrm{T}}F(X_{k+1})$
    $\mu \leftarrow \mu * \max\left\{\frac{1}{3}; 1 - (2\rho - 1)^3\right\}$
    $\nu \leftarrow 2$
  sinon
    $\mu \leftarrow \nu * \mu$
    $\nu \leftarrow 2 * \nu$
  fin
  $k \leftarrow k + 1$
fin

Algorithme 9 : Algorithme de Levenberg-Marquardt

Par définition le dénominateur est toujours positif et peut se simplifier en utilisant l'une des définitions de D_k. Par exemple, avec l'équation 4.14 de Levenberg, on obtient :

$$p(X_k) - p(X_k + D_k) = -D_k{}^{\mathrm{T}}J_F(X_k)^{\mathrm{T}}F(X_k) - \frac{1}{2}D_k{}^{\mathrm{T}}J_F(X_k)^{\mathrm{T}}J_F(X_k)D_k \tag{4.17}$$

$$= -\frac{1}{2}D_k{}^{\mathrm{T}}\left(2J_F(X_k)^{\mathrm{T}}F(X_k) + J_F(X_k)^{\mathrm{T}}J_F(X_k)D_k\right) \tag{4.18}$$

$$= \frac{1}{2}D_k{}^{\mathrm{T}}\left(\mu D_k - J_F(X_k)^{\mathrm{T}}F(X_k)\right) \tag{4.19}$$

Une valeur de ρ positive et proche de 1 indique que l'approximation est bonne et que l'on pourra réduire μ pour la prochaine itération. Une valeur négative indique que l'approximation est mauvaise et qu'il faut augmenter μ pour réduire la région de confiance. Plusieurs règles de mise à jour de μ ont été proposées. L'algorithme 9 qui synthétise la méthode de Levenberg-Marquardt utilise la mise à jour de Nielsen [16].

4.4 Méthode du gradient stochastique

Dans certain cas, on ne dispose pas de tous les coûts partiels $g_i(X)$ de la fonction objectif. C'est notamment le cas de l'identification en ligne des paramètres d'un système physique. A chaque instant k, on dispose de nouvelles données permettant de calculer un nouveau terme $\frac{1}{2}g_k^2(X)$ de la fonction objectif. On définit alors :

$$f_k(X) = \sum_{i=1}^{k} \frac{1}{2}g_i^2(X) \tag{4.20}$$

Données : f une fonction différentiable telle que $f_k(X) = \sum_{i=1}^{k} \frac{1}{2} g_i^2(X)$ et X_0 un point initial
$k \leftarrow 1$
tant que *condition d'arrêt non satisfaite* **faire**
  Recueillir g_k
  $X_k \leftarrow X_{k-1} - \alpha_k g_k(X_k) \nabla g_k(X_k)$
  $k \leftarrow k+1$
fin

Algorithme 10 : Algorithme du gradient stochastique

On souhaite trouver un minimiseur X_k^* à chaque instant k :

$$X_k^* = \arg\min_X f_k(X) \tag{4.21}$$

On utilise alors une méthode permettant de prendre en compte le nouveau coût partiel dans le calcul du minimiseur en effectuant un minimum de calculs. L'objectif est de parvenir à une mise à jour du minimiseur de la forme :

$$X_k^* = X_{k-1}^* + \delta_k(g_k) \tag{4.22}$$

Cette méthode est qualifiée de récursive, car on utilise le minimiseur X_{k-1}^* minimisant f_{k-1} pour calculer le nouveau minimiseur X_k^* minimisant f_k.

La méthode du gradient stochastique (appelé aussi gradient incrémental) est une méthode récursive du premier ordre introduite par Widrow et Hoff [22]. Son point de départ est de trouver une formulation récursive du gradient :

$$\nabla f_k(X) = \sum_{i=1}^{k} g_i(X) \nabla g_i(X) \tag{4.23}$$

$$= \nabla f_{k-1}(X) + g_k(X) \nabla g_k(X) \tag{4.24}$$

Partant de X_{k-1}, on effectue une descente du gradient pour trouver X_k, on écrit donc :

$$X_k = X_{k-1} - \alpha_k \nabla f_k(X_{k-1}) \tag{4.25}$$

$$= X_{k-1} - \alpha_k [\nabla f_{k-1}(X_{k-1}) + g_k(X_{k-1}) \nabla g_k(X_{k-1})] \tag{4.26}$$

En supposant que X_{k-1} minimisait bien f_{k-1} à l'instant $k-1$, on a $\nabla f_{k-1}(X_{k-1}) = 0$ et donc :

$$X_k = X_{k-1} - \alpha_k g_k(X_{k-1}) \nabla g_k(X_{k-1}) \tag{4.27}$$

La méthode est très simple à mettre en œuvre comme l'illustre l'algorithme 10. Sa convergence n'est pas toujours assurée et est très lente en pratique. Néanmoins, des conditions de convergence sont disponibles dans [1].

De nombreuses méthodes d'adaptation du pas α_k ont été proposées (voir par exemple [6]). Une autre modification populaire de la méthode du gradient stochastique consiste à ajouter un terme inertiel dans la mise à jour :

$$X_k = X_{k-1} - \alpha_k g_k(X_{k-1}) \nabla g_k(X_{k-1}) + \beta_k (X_{k-1} - X_{k-2}) \tag{4.28}$$

Données : f une fonction différentiable telle que $f_k(X) = \sum_{i=1}^{k} \frac{1}{2} g_i^2(X)$ et X_0 un point initial
$H_0 \leftarrow cI$ avec c un grand nombre (typiquement entre 10^4 et 10^8)
$k \leftarrow 1$
tant que *condition d'arrêt non satisfaite* **faire**
  Recueillir g_k
  $B_k \leftarrow B_{k-1} - \frac{B_{k-1}\nabla g_k(X_k)\nabla g_k(X_k)^{\mathrm{T}} B_{k-1}}{1+\nabla g_k(X_k)^{\mathrm{T}} B_{k-1}\nabla g_k(X_k)}$
  $G_k \leftarrow g_k(X_k)\nabla g_k(X_k)$
  $X_k \leftarrow X_{k-1} - \alpha_k B_k G_k$
  $k \leftarrow k+1$
fin

Algorithme 11 : Version récursive de l'algorithme de Gauss-Newton

4.5 Version récursive de la méthode de Gauss-Newton

Une amélioration plus intéressante consiste à utiliser une formulation récursive du hessien. On note G_k le gradient de la fonction objectif f_k. On a :

$$G_k = \sum_{i=1}^{k} g_i(X_k)\nabla g_i(X_k) \tag{4.29}$$

$$= G_{k-1} + g_k(X_k)\nabla g_k(X_k) \tag{4.30}$$

Si on suppose que X_{k-1} minimisait bien f_{k-1} à l'instant $k-1$, alors $G_{k-1} = 0$ et donc :

$$G_k = g_k(X_k)\nabla g_k(X_k) \tag{4.31}$$

Pour le hessien, noté H_k et approché par la formule de Gauss-Newton, il vient :

$$H_k = \sum_{i=1}^{k} \nabla g_i(X_k)\nabla g_i(X_k)^{\mathrm{T}} \tag{4.32}$$

$$= H_{k-1} + \nabla g_k(X_k)\nabla g_k(X_k)^{\mathrm{T}} \tag{4.33}$$

On trouve alors un minimiseur X_k de f_k en effectuant une mise à jour de type Newton (ce qui suppose que l'on est assez près du minimum pour l'atteindre en un coup...) :

$$X_k = X_{k-1} - H_k^{-1} G_k \tag{4.34}$$

En pratique pour éviter l'inversion de H_k à chaque fois, on introduit :

$$B_k = H_k^{-1} \tag{4.35}$$

puis on utilise le lemme d'inversion qui stipule que :

$$[A+BCD]^{-1} = A^{-1} - A^{-1}B[DA^{-1}B + C^{-1}]^{-1}DA^{-1} \tag{4.36}$$

En posant $A = H_{k-1}$, $B = D^T = \nabla g_k(X)$ et $C = 1$, on obtient la mise à jour suivante :

$$B_k = B_{k-1} - \frac{B_{k-1}\nabla g_k(X_k)\nabla g_k(X_k)^{\mathrm{T}} B_{k-1}}{1+\nabla g_k(X_k)^{\mathrm{T}} B_{k-1}\nabla g_k(X_k)} \tag{4.37}$$

On obtient alors une version récursive de l'algorithme de Gauss-Newton présentée par l'algorithme 11. De nombreuses variantes de cet algorithme ont été proposées (avec facteur d'oubli, avec réinitialisation).

4.6 En pratique

La méthode de Levenberg-Marquardt est particulièrement robuste et efficace. Elle est devenue l'algorithme de référence pour la minimisation de problèmes des moindres carrés non linéaires. Néanmoins pour des problèmes de très grandes tailles ou quand les coûts partiels g_i ne sont pas tous connus au départ, il est possible d'effectuer une optimisation *en ligne* à l'aide de la méthode du gradient stochastique ou de la version récursive de l'algorithme de Gauss-Newton.

Bibliographie

[1] Dimitri Bertsekas. *Nonlinear Programming : 2nd Edition*. Athena Scientific, 1999.

[2] Michel Bierlaire. *Introduction à l'optimisation différentiable*. Presses polytechniques et universitaires romandes, 2006.

[3] Richard P. Brent. *Algorithms for Minimisation Without Derivatives*. Prentice Hall, 1973.

[4] C. G. Broyden. The convergence of a class of double-rank minimization algorithms 2 : the new algorithm. *Journal of the institute of Mathematics and its Applications*, 6 :222–231, 1970.

[5] Thomas Coleman, Mary Ann Branch, and Andrew Grace. *Optimization Toolbox User's Guide*. The Mathworks Inc., 1999.

[6] Christian Darken, Joseph Chang Z, and John Moody. Learning rate schedules for faster stochastic gradient search. In *Proc. of Advances in Neural Information Processing Systems*. IEEE Press, 1992.

[7] W. C. Davidon. Variable metric method for minimization. Technical Report ANL-5990, A.E.C. Research and Development Report, 1959.

[8] R. Fletcher. A new approach to variable metric algorithms. *Computer Journal*, 13 :317–322, 1970.

[9] R. Fletcher and M. J. D. Powell. A rapidly convergent descent method for minimization. *Computer Journal*, 6 :163–168, 1963.

[10] R. Fletcher and C. M. Reeves. Function minimization by conjugate gradients. *Computer Journal*, 7 :149–154, 1964.

[11] D. Goldfarb. A family of variable metric updates derived by variational means. *Mathematics of Computation*, 24 :23–26, 1970.

[12] Anatoli Juditsky. *Cours d'optimisation*. Magistère de Mathématiques, Université Joseph Fourier, Grenoble. http ://ljk.imag.fr/membres/Anatoli.Iouditski/, consulté en septembre 2006.

[13] J. C. Lagarias, J. A. Reeds, M. H. Wright, and P. E. Wright. Convergence properties of the nelder-mead simplex algorithm in low dimensions. *SIAM Journal on Optimization*, 9(1) :112–147, 1998.

[14] Gérard Lebourg. *Cours d'optimisation*. Licence Génie Mathématique et Informatique, Université Paris-Dauphine. http ://cours.ufrmd.dauphine.fr/lebourg/opti-iup1.html, consulté en septembre 2007.

[15] K. Levenberg. A method for the solution of certain non-linear problems in least squares. *Quartely Journal of Applied Mathematics II*, 2(2) :164–168, 1944.

[16] K. Madsen, H. B. Nielsen, and O. Tingleff. *Methods for Non-Linear Least Squares Problems (2nd ed.)*. Informatics and Mathematical Modelling, Technical University of Denmark, DTU, 2004.

[17] D. W. Marquardt. An algorithm for least squares estimation of non-linear parameters. *Journal of the Society of Industrial and Applied Mathematics*, 11(2) :431–441, 1963.

[18] J. A. Nelder and R. Mead. A simplex method for function minimization. *Computer Journal*, 7 :308–313, 1965.

[19] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes : The Art of Scientific Computing*. Cambridge University Press, 2007.

[20] D. F. Shanno. Conditioning of quasi-newton methods for function minimization. *Mathematics of Computation*, 24 :647–656, 1970.

[21] Jonathan Richard Shewchuk. An introduction to the conjugate gradient method without the agonizing pain, 1994. Available at http ://www.cs.cmu.edu/˜jrs/jrspapers.html.

[22] B. Widrow and M. E. Hoff. Adaptive switching circuits. *IRE Wescon Convention Record*, 4 :96–104, 1960. Reprinted in Anderson & Rosenfeld (eds), Neurocomputing : foundations of research, MIT Press, 1988.