Quelques notes pour le cours

Outils mathématiques – Optimisation

Peter Sturm
INRIA Rhône-Alpes, Projet MOVI
Peter.Sturm@inrialpes.fr
http://www.inrialpes.fr/movi/people/Sturm/

Contents

1 Généralités

1.1 Bibliographie générale

- P.G. Ciarlet, "Introduction à l'analyse matricielle et à l'optimisation," Dunod, Paris, 1998.
- J. Nocedal et S.J. Wright, "Numerical Optimization," Springer Verlag, 1999.
- P.E. Gill, W. Murray et M.H. Wright, "Practical Optimization," Academic Press, 1981.
- W.H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery, "Numerical Recipes in C," Cambridge University Press, 1992. Livre disponible gratuitement "on-line" : `http://www.nr.com/`
- NEOS Guide Optimization Tree, `http://www-fp.mcs.anl.gov/otc/Guide/OptWeb/index.html`
- Computational Science Education Project, "Mathematical Optimization," `http://csep1.phy.ornl.gov/mo/mo.html`

2 Introduction

2.1 Formulation générale d'un problème d'optimisation (utilisée dans ce cours, il y en a d'autres)

trouver $\mathbf{x}$ qui minimise la valeur de la fonction	$f(\mathbf{x})$	$\mathbf{x} = (x_1, x_2, \ldots, x_n)^\mathsf{T}$
sous les contraintes	$c_i(\mathbf{x}) = 0$	$i = 1, 2, \ldots, m'$
	$c_i(\mathbf{x}) \geq 0$	$i = m'+1, m'+2, \ldots, m$

Remarques/notations.

- f est souvent appelée *fonction de coût* ou simplement *fonctionnelle* (en anglais : cost function, objective function).
- Les fonctions c_i sont des *contraintes* et les x_i les *variables* du problème.
- Maximiser f est équivalent à minimiser $-f$.
- Il peut y avoir des problèmes qui consistent à minimiser plusieurs fonctions de coût à la fois. Très souvent, ces problèmes peuvent être transformés en un problème avec une seule fonction de coût (par exemple, une somme pondérée des différentes fonctions).
- Dans ce cours, nous considérons surtout le cas $f : \mathcal{R}^n \rightarrow \mathcal{R}$.
- Nous n'allons considérer que des fonctions de coût qui sont au moins deux fois continûment dérivables.

2.2 Quelques classes de problèmes

Différents types de problèmes requièrent différentes méthodes d'optimisation. Voici quelques caractéristiques :

- nombre de variables x_i et de contraintes ;
- problèmes contraints ou non contraints ;
- problèmes avec fonction de coût linéaire, quadratique, non-linéaire ;
- contraintes linéaires, quadratiques, non-linéaires ;
- contraintes d'égalité ou d'inégalité (ou les deux) ;
- type des variables x_i : réels, entiers, permutations d'entiers, ...
- méthodes nécessitant le calcul des dérivées de la fonction de coût ou pas ;
- méthodes globales vs. méthodes locales ;
- méthodes déterministes vs. méthodes stochastiques.

2.3 Extrema

Les définitions suivantes concernent des minima – elles peuvent facilement être modifiées pour obtenir celles des maxima (ou, en général, des extrema).
Un *minimum global* $\mathbf{x}^*$ est $\mathbf{x}^*$ tel que :

$$f(\mathbf{x}^*) \leq f(\mathbf{y}) \quad \forall \mathbf{y}$$

Un *minimum global strict* $\mathbf{x}^*$ est $\mathbf{x}^*$ tel que :

$$f(\mathbf{x}^*) < f(\mathbf{y}) \quad \forall \mathbf{y} \neq \mathbf{x}^*$$

Un *minimum par rapport à un ensemble* $U \subset V$ de $f : V \rightarrow W$ est $\mathbf{x}^* \in U$ tel que :

$$f(\mathbf{x}^*) \leq f(\mathbf{y}) \quad \forall \mathbf{y} \in U$$

La définition d'un *minimum strict par rapport à un ensemble* est évidente.
Un *minimum relatif* ou *minimum local* est $\mathbf{x}^*$ tel qu'il existe un voisinage U de $\mathbf{x}^*$ avec :

$$f(\mathbf{x}^*) \leq f(\mathbf{y}) \quad \forall \mathbf{y} \in U$$

La définition d'un *minimum local strict* est évidente.

2.4 Gradient et Hessien

Le gradient et le Hessien (ou bien la matrice Hessienne) servent à décrire des conditions nécessaires et suffisantes des minima. Ils sont également utilisés dans beaucoup de méthodes d'optimisation.

Le *gradient* est le vecteur des dérivées partielles premières :

$$\mathbf{g}(\mathbf{x}) = \begin{pmatrix} \partial_1 f(\mathbf{x}) \\ \partial_2 f(\mathbf{x}) \\ \vdots \\ \partial_n f(\mathbf{x}) \end{pmatrix}$$

où nous utilisons la notation :

$$\partial_i f(\mathbf{x}) = \frac{\partial f}{\partial x_i}(\mathbf{x})$$

Le *Hessien* est la matrice des dérivées partielles secondes :

$$\mathsf{H}(\mathbf{x}) = \begin{pmatrix} \partial_{11} f(\mathbf{x}) & \cdots & \partial_{1n} f(\mathbf{x}) \\ \partial_{21} f(\mathbf{x}) & \cdots & \partial_{2n} f(\mathbf{x}) \\ \vdots & & \vdots \\ \partial_{n1} f(\mathbf{x}) & \cdots & \partial_{nn} f(\mathbf{x}) \end{pmatrix}$$

avec :

$$\partial_{ij} f(\mathbf{x}) = \frac{\partial^2 f}{\partial x_i \partial x_j}(\mathbf{x})$$

Remarques.

- D'autres notations qui sont souvent utilisées : $\bigtriangledown f(\mathbf{x})$ pour le gradient et $\bigtriangledown^2 f(\mathbf{x})$ pour le Hessien.
- Le Hessien est une matrice symétrique.
- Une matrice A est *définie positive* si : $\mathbf{u}^\mathsf{T}\mathsf{A}\mathbf{u} > 0$ pour tout $\mathbf{u} \neq \mathbf{0}$.

2.5 Conditions d'extrema

Une condition nécessaire pour un minimum (ou maximum) local $\mathbf{x}^*$ est :

$$\mathbf{g}(\mathbf{x}^*) = \mathbf{0} \ .$$

Cette équation est parfois appelée *équation d'Euler*.

Ceci implique que $\mathbf{x}^*$ est un point stationnaire de f.

Le point $\mathbf{x}^*$ est un minimum local strict si en plus le Hessien de f en $\mathbf{x}^*$ est défini positif.

2.6 Schéma des méthodes dites locales

Très souvent, on a une vague idée de l'endroit où se trouve le minimum recherché. Des méthodes locales procèdent alors typiquement ainsi : étant donné un point initial, on met à jour, de manière itérative, sa position, afin de converger vers un minimum de la fonction de coût. La mise à jour de la position du point s'effectue souvent en deux étapes : choix d'une direction et choix de la longueur du pas à faire dans cette direction.

Le processus est arrêté si un ou plusieurs critères de convergence sont satisfaits, par exemple si la norme du gradient au point actuel est inférieure à un seuil (le gradient étant nul étant une condition nécessaire pour un minimum local).

Différentes méthodes se distinguent par différentes approches pour la détermination de la direction de recherche pour le nouveau point et de la longueur de pas. Une fois la direction donnée, la détermination de la longueur de pas est un problème d'optimisation en une seule variable, typiquement plus facile à résoudre que le problème original.

Dans les sections suivantes, nous décrivons de telles méthodes. Jusqu'au §**??**, nous ne considérons que des problèmes non contraints, ensuite nous donnons un aperçu de comment traiter des problèmes avec contraintes.

2.7 Vitesse de convergence asymptotique

Il est très intéressant d'étudier la vitesse de convergence *locale* des différentes méthodes d'optimisation. Par vitesse de convergence on comprend la vitesse de décroissance vers 0 de l'erreur $e_k = \|\mathbf{x}_k - \mathbf{x}^*\|$, où $\mathbf{x}^*$ est le minimum et $\mathbf{x}_k$ l'estimation à l'itération k.

On définit alors comme **ordre de convergence** de la suite e_k vers 0, le plus grand $p > 0$ tel qu'il existe une limite finie β avec :

$$\lim_{k\to\infty} \frac{e_{k+1}}{e_k^p} \leq \beta$$

Différents cas courants sont :

convergence linéaire ou géométrique de taux β si $p = 1$ et $\beta < 1$.

convergence superlinéaire si $p = 1$ et $\beta = 0$.

convergence quadratique si $p = 2$.

La convergence quadratique est plus rapide que celle superlinéaire qui, elle, est plus rapide que celle linéaire. Quant à la convergence quadratique, elle exprime que, en gros, le nombre de chiffres significatifs double à chaque itération. Avec une convergence linéaire, on gagne (en gros) un chiffre significatif par itération.

3 Méthode de relaxation

La méthode de relaxation consiste à changer la valeur d'une seule variable à la fois, en gardant les autres valeurs fixes. Si l'on s'imagine l'espace de recherche n-dimensionnel, dont les axes correspondent aux variables, les directions de recherche sont donc parallèles aux axes et orthogonales entre elles.

Le grand inconvénient de cette méthode est qu'elle ne prend pas du tout en compte la forme de la fonction de coût – les directions de recherche sont préscrites à l'avance. Elle a donc de très mauvaises propriétés de convergence.

4 Méthodes de gradient

Puisque nous cherchons un minimum de la fonction de coût, il est naturel de chercher le nouveau point dans une direction descendante . La direction la plus descendante (localement) est donnée par le gradient. Les méthodes de gradient (en anglais : *steepest descent methods*) adoptent alors la direction donnée par le gradient comme direction de recherche.

Différentes méthodes de gradient se distinguent d'après le choix de la longueur de pas :

méthode de gradient à pas optimal : on essaie de déterminer la longueur de pas optimale, c'est-à-dire qui minimise la valeur de la fonction de coût, restreinte sur la droite donnée par le point actuel et la direction de recherche.

méthode de gradient à pas fixe : la définition est évidente.

méthode de gradient à pas variable : la longueur de pas varie au cours des itérations, d'après certains critères.

5 Méthode de Newton

La méthode de Newton (souvent aussi appelée méthode Newton-Raphson) est basée sur une approximation locale du deuxième ordre de la fonction de coût. Pour la fonction de coût approchée (quadratique), le minimum est – théoriquement – facile à déterminer. Autour du minimum trouvé, la fonction de coût originale est alors à nouveau approchée au deuxième ordre, et ainsi de suite.

Nous décrivons d'abord la méthode de Newton pour trouver une *racine* d'une fonction en une variable, puis la méthode analogue pour la recherche d'un *minimum*, d'abord en une, puis en n dimensions.

5.1 Méthode de Newton pour trouver une racine d'une fonction en une variable

Etant donnés une fonction $g : \mathcal{R} \to \mathcal{R}$ et un point x_0 initial, la question se pose comment s'approcher d'une racine de g, en partant de x_0. La méthode de Newton utilise une approximation linéaire (ou : au premier ordre) de g autour de x_0 pour ce faire :

$$g(x_0 + h) = g(x_0) + g'(x_0)h + \epsilon$$

où ϵ contient les termes d'ordres supérieurs, qui seront negligés dans la suite.
La racine de cette approximation est donnée par :

$$h^* = -\frac{g(x_0)}{g'(x_0)}$$

ou bien :

$$h^* = -\{g'(x_0)\}^{-1} g(x_0)$$

Nous obtenons alors un nouveau point $x_1 = x_0 + h^*$. Puisque la longueur de pas h^* a été déterminée seulement à l'aide d'une *approximation* (locale) de la fonction g, nous ré-itérons le processus. La méthode consiste alors en l'estimation d'une suite de points :

$$x_{k+1} = x_k - \{g'(x_k)\}^{-1} g(x_k) \tag{1}$$

Remarque. Il existe des pièges, par exemple si le point actuel est (proche d') un extremum ou d'un point stationnaire : $g'(x_k) \approx 0$. Dans ce cas, la recherche d'après l'équation (**??**) peut partir dans les choux (on divise par un nombre très petit, ce qui peut résulter en un pas très long).

5.2 Méthode de Newton pour trouver un minimum d'une fonction en une variable

Etant donnés une fonction $f : \mathcal{R} \to \mathcal{R}$ et un point x_0 initial, nous appliquons la méthode de Newton, cette fois-ci pour trouver un *minimum local* de f. Puisque la condition nécessaire d'un minimum local est que la dérivée première est nulle, nous allons utiliser la méthode précédente pour chercher une racine de la dérivée de f. Remplacer g par f' dans l'équation (**??**) donne alors :

$$x_{k+1} = x_k - \{f''(x_k)\}^{-1} f'(x_k) \tag{2}$$

Une autre manière d'obtenir la même formule est de partir d'une approximation au *deuxième* ordre de f :

$$f(x_k + h) = f(x_k) + f'(x_k)h + f''(x_k)\frac{h^2}{2} + \epsilon$$

Le but est alors de trouver h^* qui *minimise* $f(x_k) + f'(x_k)h + f''(x_k)\frac{h^2}{2}$. La dérivée par h de cette expression doit alors vérifier :

$$f'(x_k) + f''(x_k)h^* = 0$$

d'où on obtient :

$$h^* = -\frac{f'(x_k)}{f''(x_k)}$$

On retrouve alors le deuxième terme de la formule (**??**).

Remarque. Cette méthode hérite du piège mentionné dans le paragraphe précédent : si le point actuel est (proche d') un point stationnaire ($f''(x_k) \approx 0$), l'équation (**??**) peut donner un nouveau point très (trop) éloigné.

5.3 Méthode de Newton en n dimensions

La méthode de Newton peut être généralisée au cas de fonctions $f : \mathcal{R}^n \to \mathcal{R}$.
Une approximation au deuxième ordre peut s'écrire sous la forme :

$$f(\mathbf{x}_0 + \mathbf{h}) = f(\mathbf{x}_0) + \mathbf{h}^\mathsf{T}\mathbf{g} + \frac{1}{2}\mathbf{h}^\mathsf{T}\mathsf{H}\mathbf{h} + \epsilon$$

Notation. Ici, $\mathbf{x} = (x_1, \ldots, x_n)^\mathsf{T}$ est le vecteur des variables, $\mathbf{g}$ le gradient de f et H le Hessien, donc une matrice symétrique de dimension $n \times n$. Le vecteur $\mathbf{h}$ est tout comme $\mathbf{x}$ de longueur n. L'expression $\cdot^\mathsf{T}$ désigne la transposition (de matrices ou de vecteurs) et $\mathbf{v}^\mathsf{T}\mathbf{w}$ est donc le produit scalaire des deux vecteurs $\mathbf{v}$ et $\mathbf{w}$.
Le gradient de $f(\mathbf{x}_0 + \mathbf{h})$ par rapport aux coefficients de $\mathbf{h}$ est :

$$\mathbf{g} + \mathsf{H}\mathbf{h}$$

Le minimum de l'approximation quadratique de f est atteint si le gradient est le vecteur nul, donc il est donné par :

$$\mathbf{h}^* = -\mathsf{H}^{-1}\mathbf{g}$$

La formule de récursion est essentiellement la même que pour le cas uni-dimensionnel.
Remarques.

- Si la fonction de coût est parfaitement quadratique, la méthode de Newton trouve le minimum global en 1 itération.
- L'un des inconvénients de la méthode est la nécessité de calculer l'inverse du Hessien, ou bien de résoudre le système d'équations $\mathsf{H}\mathbf{h}^* = -\mathbf{g}$. Pour des problèmes avec beaucoup de variables, ceci devient vite impraticable. Il y a plusieurs remèdes à ce problème. Par exemple, les méthodes de gradient ou de gradient conjugué (expliquées plus loin). Il existe aussi plusieurs variantes approximatives de la méthode de Newton, qui seront expliquées plus loin.
- La convergence de la méthode de Newton est quadratique.

Figure 1: La fonction deux-dimensionnelle de Rosenbrock, souvent utilisée pour illustrer des caractéristiques de convergence de méthodes d'optimisation, définie par $f(x, y) = (1 - x)^2 + 100\,(y - x^2)^2$. En bas : lignes de niveau de la fonction et minimum global à $(1, 1)$.

6 Quelques variantes de la méthode de Newton

Les deux gros inconvénients de la méthode de Newton sont :

- Coût de calcul (et de mémoire) élevé (calcul et inverse du Hessien).
- Il n'y a aucune garantie que le vecteur de mise à jour $\mathbf{h}^*$ calculé par $\mathbf{h}^* = -\mathsf{H}^{-1}\mathbf{g}$ donne lieu à un nouveau point où la valeur de f est plus petite qu'avant la mise à jour ! Il existe donc un risque de divergence.

Dans la suite, quelques remèdes à ces problèmes sont présentés.

6.1 Quasi-Newton

Le principe de base des méthodes quasi-Newton est de ne pas calculer explicitement le Hessien (et son inverse), mais d'utiliser une approximation qui est mise à jour au cours des itérations. L'algorithme général peut alors être décrit comme suit :

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mu_k \mathbf{d}_k \qquad \text{avec } \mathbf{d}_k = -\mathsf{S}_k^{-1}\mathbf{g}(\mathbf{x}_k) \tag{3}$$

Avec $\mathsf{S}_k = \mathsf{H}(\mathbf{x}_k)$ et $\mu_k = 1$, on retrouve la méthode de Newton. Quant aux méthodes quasi-Newton, elles imposent souvent, lors du calcul des approximations S_k du Hessien, qu'elles soient des matrices définies positives. Cette propriété garantit que la direction $\mathbf{d}_k$ est une direction de *descente* :

$$\begin{aligned} f(\mathbf{x}_k + \mu_k \mathbf{d}_k) &= f(\mathbf{x}_k) + \mu_k \mathbf{g}(\mathbf{x}_k)^\mathsf{T} \mathbf{d}_k + O(\mu_k^2) \\ &= f(\mathbf{x}_k) - \mu_k \underbrace{\mathbf{g}(\mathbf{x}_k)^\mathsf{T} \mathsf{S}_k^{-1} \mathbf{g}(\mathbf{x}_k)}_{>0 \text{ car } \mathsf{S}_k \text{ définie positive}} + O(\mu_k^2) \\ &< f(\mathbf{x}_k) \end{aligned}$$

si μ_k est suffisamment petit.

Supposons maintenant qu'on se trouve à la $k$itération, i.e. le point actuel est $\mathbf{x}_k$ et l'approximation du Hessien actuelle est S_k (nous discutons plus loin comment initialiser le processus, i.e. comment calculer S_0). Pour calculer le nouveau point, la direction de recherche $\mathbf{d}_k$ est calculée d'après l'équation (**??**), puis la longueur de pas μ_k par exemple par une optimisation uni-dimensionnelle. Le calcul de la direction de recherche utilise le gradient, évalué au point actuel $\mathbf{x}_k$. A la prochaine itération, le gradient sera aussi calculé au nouveau point $\mathbf{x}_{k+1}$.

D'une part, de manière intuitive, les deux gradients, pour les points $\mathbf{x}_k$ et $\mathbf{x}_{k+1}$, nous donnent des informations sur la courbure de la fonction f, le long de la direction de recherche $\mathbf{d}_k$. D'autre part, c'est le Hessien d'une fonction qui nous donne sa courbure. On peut donc espérer que l'information acquise par le calcul des gradients nous dise quelque chose sur le Hessien, et par conséquent soit utile pour calculer une nouvelle (et peut-être meilleure) approximation S_{k+1} du Hessien.

En principe, on accumule alors, au cours des itérations, des informations (des gradients) permettant de mieux approcher le Hessien. En même temps pourtant, du moins pour des fonctions plus compliquées que quadratiques, le Hessien change au cours des itérations (puisqu'il sera calculé pour des points différents). Donc, en quelque sorte, on fait une course-poursuite ...

Regardons maintenant comment utiliser les gradients aux points $\mathbf{x}_k$ et $\mathbf{x}_{k+1}$ pour calculer la nouvelle approximation S_{k+1} du Hessien. Pour ce faire, nous effectuons l'approximation au premier ordre du gradient au point $\mathbf{x}_{k+1}$. Soit $\mathbf{s}_k = \mathbf{x}_{k+1} - \mathbf{x}_k$. Alors :

$$\begin{aligned} \mathbf{g}(\mathbf{x}_{k+1}) &= \mathbf{g}(\mathbf{x}_k + \mathbf{s}_k) \\ &\approx \mathbf{g}(\mathbf{x}_k) + \mathsf{H}(\mathbf{x}_k)\mathbf{s}_k \end{aligned}$$

Le vrai Hessien vérifie donc :

$$\mathsf{H}(\mathbf{x}_k)\mathbf{s}_k \approx \mathbf{g}(\mathbf{x}_{k+1}) - \mathbf{g}(\mathbf{x}_k)$$

Il est donc raisonnable d'imposer cette condition sur la nouvelle matrice S_{k+1} :

$$\mathsf{S}_{k+1}\mathbf{s}_k = \mathsf{S}_{k+1}(\mathbf{x}_{k+1} - \mathbf{x}_k) = \mathbf{g}(\mathbf{x}_{k+1}) - \mathbf{g}(\mathbf{x}_k) \tag{4}$$

Cette équation est souvent appelée, en anglais, *quasi-Newton condition* ou bien *secant equation*.

Cette seule équation entre vecteurs ne définit bien entendu pas entièrement la matrice S_{k+1}. Typiquement on essaie alors de déterminer une S_{k+1} qui satisfait cette équation et qui n'est pas trop différente de l'estimation précédente S_k. Il existe différentes méthodes pour ce faire. Souvent, S_{k+1} est déterminée via une matrice de mise à jour , U_k de *petit rang* :

$$\mathsf{S}_{k+1} = \mathsf{S}_k + \mathsf{U}_k$$

Nous examinons maintenant le cas de base d'une matrice U_k de rang 1. Toute matrice de rang 1 peut être obtenue via deux vecteurs de longueurs appropriées (afin de simplifier la lecture, nous omettons l'indice k) :

$$\mathsf{U} = \mathbf{u}\mathbf{v}^\mathsf{T}$$

Si nous notons $\mathbf{s}_k = \mathbf{x}_{k+1} - \mathbf{x}_k$ et $\mathbf{y}_k = \mathbf{g}(\mathbf{x}_{k+1}) - \mathbf{g}(\mathbf{x}_k)$, la condition (**??**) devient alors :

$$\mathsf{S}_k\mathbf{s}_k + \mathbf{u}\mathbf{v}^\mathsf{T}\mathbf{s}_k = \mathbf{y}_k$$

ou bien :

$$\mathbf{u}\left(\mathbf{v}^\mathsf{T}\mathbf{s}_k\right) = \mathbf{y}_k - \mathsf{S}_k\mathbf{s}_k$$

Si $\mathbf{y}_k = \mathsf{S}_k\mathbf{s}_k$, on peut directement utiliser $\mathsf{S}_{k+1} = \mathsf{S}_k$.

Dans le cas contraire, nous poursuivons :

$$\mathbf{u} = \frac{1}{\mathbf{v}^\mathsf{T}\mathbf{s}_k}\left(\mathbf{y}_k - \mathsf{S}_k\mathbf{s}_k\right)$$

et donc :

$$\mathsf{S}_{k+1} = \mathsf{S}_k + \frac{1}{\mathbf{v}^\mathsf{T}\mathbf{s}_k}\left(\mathbf{y}_k - \mathsf{S}_k\mathbf{s}_k\right)\mathbf{v}^\mathsf{T}$$

Cette matrice vérifiera la contrainte quasi-Newton (**??**), pour tout choix de $\mathbf{v}$. Nous disposons donc de quelques degrés de liberté qu'on peut tenter d'utiliser afin de garantir que la matrice S_{k+1} ait des propriétés désirables. Puisque S_{k+1} est utilisée comme approximation du Hessien, qui est une matrice symétrique, on voudra naturellement imposer qu'elle soit symétrique, elle aussi. Ceci n'est pas le cas pour n'importe quel vecteur $\mathbf{v}$ comme on verra.

On suppose ici que l'estimation précédente, S_k, est une matrice symétrique. La symétrie de S_{k+1} s'exprime par $\mathsf{S}_{k+1} = \mathsf{S}_{k+1}^\mathsf{T}$, alors :

$$\mathsf{S}_k + \frac{1}{\mathbf{v}^\mathsf{T}\mathbf{s}_k}\left(\mathbf{y}_k - \mathsf{S}_k\mathbf{s}_k\right)\mathbf{v}^\mathsf{T} = \mathsf{S}_k^\mathsf{T} + \frac{1}{\mathbf{v}^\mathsf{T}\mathbf{s}_k}\mathbf{v}(\mathbf{y}_k - \mathsf{S}_k\mathbf{s}_k)^\mathsf{T}$$

Puisque S_k est symétrique, on obtient :

$$\left(\mathbf{y}_k - \mathsf{S}_k\mathbf{s}_k\right)\mathbf{v}^\mathsf{T} = \mathbf{v}(\mathbf{y}_k - \mathsf{S}_k\mathbf{s}_k)^\mathsf{T}$$

Si $\mathbf{v} = \lambda\left(\mathbf{y}_k - \mathsf{S}_k\mathbf{s}_k\right)$, pour un λ réel, cette équation est vérifiée (il peut être montré qu'en général, c'est la seule solution possible).

La matrice S_{k+1}, garantie symétrique, est alors :

$$\begin{aligned}\mathsf{S}_{k+1} &= \mathsf{S}_k + \frac{1}{\lambda(\mathbf{y}_k - \mathsf{S}_k\mathbf{s}_k)^\mathsf{T}\mathbf{s}_k}\left(\mathbf{y}_k - \mathsf{S}_k\mathbf{s}_k\right)\lambda(\mathbf{y}_k - \mathsf{S}_k\mathbf{s}_k)^\mathsf{T} \\ &= \mathsf{S}_k + \frac{1}{(\mathbf{y}_k - \mathsf{S}_k\mathbf{s}_k)^\mathsf{T}\mathbf{s}_k}\left(\mathbf{y}_k - \mathsf{S}_k\mathbf{s}_k\right)\left(\mathbf{y}_k - \mathsf{S}_k\mathbf{s}_k\right)^\mathsf{T}\end{aligned}$$

Cette méthode de mise à jour de l'approximation du Hessien est appelée *symmetric rank-one update* en anglais.

Remarques.

- Il existe d'autres méthodes de mise à jour, utilisant des matrice de rang 2 : PSB (Powell-Symmetric-Broyden), DFP (Davidon-Fletcher-Powell) et BFGS (Broyden-Fletcher-Goldfarb-Shanno), qui est actuellement considérée comme la meilleure méthode.
- Il existe des méthodes adaptées à la mise à jour de l'*inverse* du Hessien, qui rendent donc le calcul de l'inverse dans $\mathbf{d}_k = -\mathsf{S}_k^{-1}\mathbf{g}$ obsolète.
- Encore d'autres méthodes garantissent que les approximations du Hessien sont *définies positives*, en plus d'être symétriques (comme le vrai Hessien). Puisque toute matrice définie positive peut être décomposée en un produit d'une matrice triangulaire avec sa transposée : $\mathsf{S}_k = \mathsf{L}_k\mathsf{L}_k^\mathsf{T}$, c'est typiquement la matrice L qui est mise à jour, et non S.
- Les méthodes quasi-Newton sont aussi appelées, en anglais, des *variable metric methods*.

Résumé. Les avantages des méthodes quasi-Newton par rapport à la méthode de Newton sont :

- le Hessien n'est pas calculé explicitement (ce qui est souvent très coûteux en temps de calcul).
- si l'inverse du Hessien est approché, le calcul de la direction $\mathbf{d}_k$ ne nécessite qu'une multiplication matrice-vecteur (en dehors du calcul du gradient).
- les méthodes gérant des approximations définies positives du Hessien, garantissent que les directions de recherche sont descendantes.

L'inconvénient est évidemment que l'on utilise moins d'informations sur la forme de la fonction de coût.

Nous n'avons pas encore décrit comment initialiser le processus itératif d'une méthode quasi-Newton. Ce qui est fait très souvent en pratique et semble marcher assez bien, est de choisir S_0 simplement comme étant la matrice d'identité (ça revient à effectuer l'itération initiale avec la méthode du gradient).

6.2 Méthodes de Newton discrètes

Parfois, il est coûteux, ou simplement difficile de calculer les dérivées secondes de la fonction de coût analytiquement et donc de calculer le Hessien directement. Les méthodes de Newton discrètes procèdent alors par une approximation du Hessien, en utilisant des différences finies. Considérons par exemple l'approximation au premier ordre du gradient au point $\mathbf{x}$, auquel est ajouté un déplacement le long de l'axe correspondant à la $i$variable :

$$\mathbf{g}(\mathbf{x} + \lambda \mathbf{e}_i) = \mathbf{g}(\mathbf{x}) + \lambda \mathsf{H}(\mathbf{x})\mathbf{e}_i + \epsilon$$

où $\mathbf{e}_i$ est le vecteur ne contenant que des zéros, sauf que le $i$coefficient est égal à 1. Alors, en négligeant ϵ :

$$\mathsf{H}(\mathbf{x}) \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} = \frac{1}{\lambda} \left(\mathbf{g}(\mathbf{x} + \lambda \mathbf{e}_i) - \mathbf{g}(\mathbf{x}) \right)$$

Le terme sur la gauche n'est rien d'autre que la $i$colonne du Hessien. De cette manière on peut donc calculer des approximations pour toutes ses colonnes. Un facteur critique est le choix fait pour λ qui doit être ni trop grand (l'approximation ne sera pas valide) ni trop petit (les gradients évalués à des points dont la distance est λ ne fournissent pas assez d'informations sur le Hessien et l'erreur d'arrondi peut invalider l'estimation) ...

L'approximation S du Hessien obtenue colonne-par-colonne n'est en général pas une matrice symétrique. On utilise alors typiquement la matrice symétrique obtenue ainsi :

$$\frac{1}{2} \left(\mathsf{S} + \mathsf{S}^\mathsf{T} \right)$$

7 Méthodes du gradient conjugué

Une des raisons pour utiliser les méthodes quasi-Newton peut être un temps de calcul élevé ou une implémentation très difficile du Hessien, donc des dérivées secondes de la fonction de coût. Les inconvénients que ces méthodes partagent avec la méthode de Newton sont :

- la nécessité de stocker le Hessien ou son approximation ;
- la nécessité de résoudre un système d'équations (ou bien d'inverser une matrice) à chaque itération (ce qui peut être évité par quelques variantes quasi-Newton).

Pour de très grands problèmes d'optimisation (avec beaucoup de variables), ces méthodes peuvent donc devenir impraticables.

Une solution populaire est alors la famille des méthodes du gradient conjugué. Tout comme les méthodes du gradient (voir §**??**) ou les méthodes quasi-Newton (voir §**??**), elles ne requièrent pas le calcul du Hessien. Et contrairement aux méthodes quasi-Newton, elles ne stockent pas une matrice de la taille du Hessien ($n \times n$), mais seulement un ou quelques vecteurs (de longueur n). Contrairement aux méthodes du gradient (voir §**??**), les méthodes du gradient *conjugué* tentent d'utiliser les informations acquises au cours des itérations : de toute façon, chaque itération requiert le calcul du gradient au point actuel ; l'information qu'il donne sur la courbure de la fonction de coût est alors prise en compte. Par rapport aux méthodes quasi-Newton pourtant, on dispose de moins de mémoire pour stocker les informations acquises – la forme de la fonction de coût sera donc moins bien représentée. Néanmoins, avec un temps de calcul légèrement plus élevé que pour les méthodes du gradient, on gagne en général beaucoup en vitesse (meilleur ordre de convergence), surtout dans les cas difficiles tels que les « vallées élongées » de la fonction de coût.

7.1 Méthode du gradient conjugué linéaire

Originalement, les méthodes du gradient conjugué ont été dévelopées dans les années 50 pour la résolution de systèmes d'équations *linéaires* avec matrices *symétriques et définies positives* A :

$$\mathsf{A}\mathbf{x} - \mathbf{b} = \mathbf{0}$$

Elles sont une bonne alternative à par exemple la méthode d'élimination de Gauß, si A est large.

La *solution* $\mathbf{x}$ du système *linéaire* est également l'*extremum* de la fonction *quadratique* suivante :

$$f(\mathbf{x}) = -\mathbf{b}^\mathsf{T}\mathbf{x} + \frac{1}{2}\mathbf{x}^\mathsf{T}\mathsf{A}\mathbf{x} + \text{const} \tag{5}$$

d'où l'intérêt des méthodes du gradient conjugué pour l'optimisation. Dans ce paragraphe nous ne considérons que des fonctions parfaitement quadratiques, dans le paragraphe suivant on se penchera sur des fonctions non-linéaires générales. Notons pour plus tard que le gradient de f est donné par :

$$\mathbf{g}(\mathbf{x}) = -\mathbf{b} + \mathsf{A}\mathbf{x} \tag{6}$$

Les méthodes du gradient conjugué procèdent, tout comme d'autres méthodes locales, par la détermination successive de directions de recherche et de longueurs de pas. Quant aux méthodes du gradient, elles ne regardent que la structure locale de la fonction de coût, et ce qui se produit souvent est un phénomène de zigzague : l'après-prochaine direction de recherche peut coïncider plus au moins avec la direction actuelle. Les méthodes du gradient conjugué, quant à elles, tentent d'éviter celà, par la détermination de directions de recherche qui soient différentes des directions précédentes.

Supposons que $\mathbf{x}_k$ est une approximation du minimum de f. Soient $k+1$ vecteurs linéairement indépendants, $\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_k$ (les directions de recherche). Soit $\mathbf{x}_k + \sum_i \omega_i \mathbf{p}_i$ le sous-espace des points qui sont atteignables par les directions de recherche en partant de $\mathbf{x}_k$. Ceci peut être écrit de manière plus compacte comme :

$$\mathbf{x}_k + \mathsf{P}_k \omega$$

où P_k est la matrice dont les colonnes sont les $\mathbf{p}_i$ et ω le vecteur de longueur $k+1$ composé des ω_i. Nous cherchons maintenant le minimum de f dans ce sous-espace :

$$\underset{\omega}{\text{minimiser}} \quad f(\mathbf{x}_k + \mathsf{P}_k \omega)$$

Si nous plaçons ceci dans l'équation (**??**), il s'agit alors de trouver ω qui minimise :

$$f(\mathbf{x}_k + \mathsf{P}_k\omega) = -\mathbf{b}^\mathsf{T}\mathbf{x}_k - \mathbf{b}^\mathsf{T}\mathsf{P}_k\omega + \frac{1}{2}\mathbf{x}_k^\mathsf{T}\mathsf{A}\mathbf{x}_k + \frac{1}{2}\mathbf{x}_k^\mathsf{T}\mathsf{A}\mathsf{P}_k\omega + \frac{1}{2}\omega^\mathsf{T}\mathsf{P}_k^\mathsf{T}\mathsf{A}\mathbf{x}_k + \frac{1}{2}\omega^\mathsf{T}\mathsf{P}_k^\mathsf{T}\mathsf{A}\mathsf{P}_k\omega + \text{const}$$

Les termes où ω n'apparaît pas, sont constants, donc on peut les négliger pour la minimisation. Notons également que $\mathbf{x}_k^\mathsf{T}\mathsf{A}\mathsf{P}_k\omega = \omega^\mathsf{T}\mathsf{P}_k^\mathsf{T}\mathsf{A}^\mathsf{T}\mathbf{x}_k = \omega^\mathsf{T}\mathsf{P}_k^\mathsf{T}\mathsf{A}\mathbf{x}_k$ (puisque $\mathbf{x}_k^\mathsf{T}\mathsf{A}\mathsf{P}_k\omega$ est un scalaire et A est symétrique). Le problème se simplifie alors :

$$\underset{\omega}{\text{minimiser}} \quad -\mathbf{b}^\mathsf{T}\mathsf{P}_k\omega + \mathbf{x}_k^\mathsf{T}\mathsf{A}\mathsf{P}_k\omega + \frac{1}{2}\omega^\mathsf{T}\mathsf{P}_k^\mathsf{T}\mathsf{A}\mathsf{P}_k\omega$$

En utilisant l'expression donnée en (**??**), ceci peut s'écrire :

$$\underset{\omega}{\text{minimiser}} \quad \mathbf{g}(\mathbf{x}_k)^\mathsf{T}\mathsf{P}_k\omega + \frac{1}{2}\omega^\mathsf{T}\mathsf{P}_k^\mathsf{T}\mathsf{A}\mathsf{P}_k\omega$$

Puisque la dérivee par ω est égale à $\mathsf{P}_k^\mathsf{T}\mathbf{g}(\mathbf{x}_k) + \mathsf{P}_k^\mathsf{T}\mathsf{A}\mathsf{P}_k\omega$, le minimum est atteint pour

$$\omega^* = -\big(\mathsf{P}_k^\mathsf{T}\mathsf{A}\mathsf{P}_k\big)^{-1}\mathsf{P}_k^\mathsf{T}\mathbf{g}(\mathbf{x}_k)$$

Le minimum $\mathbf{x}_{k+1}$ du sous-espace décrit ci-dessus est alors donné par :

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{x}_k + \mathsf{P}_k\omega^* \\ &= \mathbf{x}_k - \mathsf{P}_k\big(\mathsf{P}_k^\mathsf{T}\mathsf{A}\mathsf{P}_k\big)^{-1}\mathsf{P}_k^\mathsf{T}\mathbf{g}(\mathbf{x}_k) \end{aligned} \tag{7}$$

Nous énonçons dans la suite quelques propriétés remarquables de cette suite $\mathbf{x}_0, \ldots$ (notons que l'équation ci-dessus ne peut pas être simplifiée directement, puisque la matrice P_k n'est pas carrée en général, et donc non inversible). Premièrement, si l'on calcule le gradient

au point $\mathbf{x}_{k+1}$, on constate qu'il est orthogonal à tous les vecteurs $\mathbf{p}_0, \ldots, \mathbf{p}_k$ dans P_k. Le gradient est donné par (cf. l'équation (**??**)) :

$$\begin{aligned} \mathbf{g}(\mathbf{x}_{k+1}) &= -\mathbf{b} + \mathsf{A}\mathbf{x}_k - \mathsf{A}\mathsf{P}_k\left(\mathsf{P}_k^\mathsf{T}\mathsf{A}\mathsf{P}_k\right)^{-1}\mathsf{P}_k^\mathsf{T}\mathbf{g}(\mathbf{x}_k) \\ &= \mathbf{g}(\mathbf{x}_k) - \mathsf{A}\mathsf{P}_k\left(\mathsf{P}_k^\mathsf{T}\mathsf{A}\mathsf{P}_k\right)^{-1}\mathsf{P}_k^\mathsf{T}\mathbf{g}(\mathbf{x}_k) \end{aligned}$$

Alors :

$$\begin{aligned} \mathsf{P}_k^\mathsf{T}\mathbf{g}(\mathbf{x}_{k+1}) &= \mathsf{P}_k^\mathsf{T}\mathbf{g}(\mathbf{x}_k) - \mathsf{P}_k^\mathsf{T}\mathsf{A}\mathsf{P}_k\left(\mathsf{P}_k^\mathsf{T}\mathsf{A}\mathsf{P}_k\right)^{-1}\mathsf{P}_k^\mathsf{T}\mathbf{g}(\mathbf{x}_k) \\ &= \mathsf{P}_k^\mathsf{T}\mathbf{g}(\mathbf{x}_k) - \mathsf{P}_k^\mathsf{T}\mathbf{g}(\mathbf{x}_k) \\ &= \mathbf{0} \end{aligned}$$

ce qui implique l'orthogonalité : $\mathbf{p}_i^\mathsf{T}\mathbf{g}(\mathbf{x}_{k+1}) = 0, \forall i = 0, \ldots, k$.

Et plus généralement :

$$\mathbf{p}_i^\mathsf{T}\mathbf{g}(\mathbf{x}_j) = 0 \quad \text{si } i < j$$

Le gradient au point actuel est donc orthogonal aux directions de recherche précédentes, ce qui veut dire qu'il est possible de trouver une nouvelle direction de recherche qui soit descendante (sauf si le point actuel est déjà un minimum local strict).

Regardons de plus près le terme $\mathsf{P}_k^\mathsf{T}\mathbf{g}(\mathbf{x}_k)$ dans l'équation (**??**) :

$$\begin{aligned} \mathsf{P}_k^\mathsf{T}\mathbf{g}(\mathbf{x}_k) &= \begin{pmatrix} \mathbf{p}_0^\mathsf{T} \\ \mathbf{p}_1^\mathsf{T} \\ \vdots \\ \mathbf{p}_{k-1}^\mathsf{T} \\ \mathbf{p}_k^\mathsf{T} \end{pmatrix} \mathbf{g}(\mathbf{x}_k) \\ &= \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ \gamma_k \end{pmatrix} \\ &= \gamma_k \mathbf{e}_k \end{aligned} \tag{8}$$

où $\gamma_k = \mathbf{p}_k^\mathsf{T}\mathbf{g}(\mathbf{x}_k)$ et $\mathbf{e}_k$ est le vecteur nul, en dehors du $k$coefficient, qui est égal à 1 (la numérotation des coefficients commence à 0 ici).

Supposons maintenant que les vecteurs $\mathbf{p}_i, i = 0, \ldots, k$ sont *mutuellement conjugués* par rapport à la matrice A :

$$\mathbf{p}_i^\mathsf{T}\mathsf{A}\mathbf{p}_j = 0 \ \text{ si } i \neq j$$

Alors, la matrice $\mathsf{P}_k^\mathsf{T}\mathsf{A}\mathsf{P}_k$ (et aussi son inverse) est *diagonale*. Par conséquent :

$$\left(\mathsf{P}_k^\mathsf{T}\mathsf{A}\mathsf{P}_k\right)^{-1}\mathbf{e}_k = \rho_k \mathbf{e}_k \tag{9}$$

pour un scalaire ρ_k.

Et finalement, l'équation (**??**) devient, en utilisant les relations (**??**) et (**??**) :

$$\begin{aligned}
\mathbf{x}_{k+1} &= \mathbf{x}_k - \mathsf{P}_k\left(\mathsf{P}_k^\mathsf{T}\mathsf{A}\mathsf{P}_k\right)^{-1}\mathsf{P}_k^\mathsf{T}\mathbf{g}(\mathbf{x}_k) \\
&= \mathbf{x}_k - \gamma_k\mathsf{P}_k\left(\mathsf{P}_k^\mathsf{T}\mathsf{A}\mathsf{P}_k\right)^{-1}\mathbf{e}_k \\
&= \mathbf{x}_k - \gamma_k\rho_k\mathsf{P}_k\mathbf{e}_k \\
&= \mathbf{x}_k - \alpha_k\mathbf{p}_k
\end{aligned}$$

où $\alpha_k = \gamma_k\rho_k$.

Il peut être montré qu'on peut obtenir des directions $\mathbf{p}_i$ mutuellement conjuguées par une itération de la forme :

$$\mathbf{p}_k = -\mathbf{g}(\mathbf{x}_k) + \beta_k\mathbf{p}_{k-1} \tag{10}$$

Comme première direction, on adopte normalement le gradient au point initial : $\mathbf{p}_0 = -\mathbf{g}(\mathbf{x}_0)$ (le signe $-$ est nécessaire pour *descendre* le long du gradient). Nous renvoyons au prochain paragraphe pour une description du calcul des β_i.

Avec cette construction des $\mathbf{p}_i$, il peut être montré que :

$$\mathbf{g}(\mathbf{x}_k)^\mathsf{T}\mathbf{g}(\mathbf{x}_i) = 0 \quad \forall i < k$$

c'est-à-dire que le gradient au nouveau point est orthogonal aux gradients des points précédents. Intuitivement, ceci veut dire que, au cours des itérations, on ne retourne pas dans la proximité de regions déjà explorées – la convergence est plus rapide que pour les méthodes du gradient.

Remarques.

- Pour des fonctions de coût quadratiques, les β_i du schéma (**??**) peuvent être déterminés analytiquement ; pour les fonctions non-linéaires générales, il existe plusieurs méthodes pour les calculer (voir le prochain paragraphe).
- Modulo des erreurs d'arrondi dans les calculs, la méthode du gradient conjugué trouve le minimum d'une fonction quadratique en n variables, en n itérations au plus.
- Le succès de la méthode dépend fortement du conditionnement de la matrice A. En gros, plus la matrice est diagonale et plus elle a des valeurs propres pas trop différentes, plus vite et plus fiable sera la convergence. En pratique, les méthodes du gradient conjugué sont donc souvent précédées par une étape de *pré-conditionnement* de A : on la multiplie avec des matrices appropriées, résoud le système ainsi modifié, et modifie la solution pour obtenir la solution du système original.

Schéma algorithmique. La méthode du gradient conjugué linéaire procède ainsi :

0. Initialisation (première direction de recherche) : $\mathbf{p}_0 = -\mathbf{g}(\mathbf{x}_0)$, $\mathsf{P}_0 = (\mathbf{p}_0)$ et $k = 0$.

1. Calcul des scalaires γ_k et ρ_k en utilisant les équations (**??**) et (**??**), puis $\alpha_k = \gamma_k\rho_k$.

2. Calcul du nouveau point : $\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k\mathbf{p}_k$.

3. Calcul du scalaire β_{k+1}.

4. Calcul de la nouvelle direction de recherche $\mathbf{p}_{k+1}$ d'après l'équation (**??**).

5. Former la matrice P_{k+1} à partir P_k, en rajoutant le vecteur $\mathbf{p}_{k+1}$ dans une nouvelle colonne.

6. $k \to k+1$, puis aller à l'étape 1.

7.2 Méthodes du gradient conjugué non-linéaires

La méthode décrite dans le paragraphe précédent peut être appliquée pour minimiser des fonctions de coût générales. L'algorithme générique est de la forme :

0. Direction de recherche initiale : $\mathbf{p}_0 = -\mathbf{g}(\mathbf{x}_0)$.

1. Résoudre le problème d'optimisation uni-dimensionnel en α_k :

$$\text{minimiser} \quad f(\mathbf{x}_{k+1}) \quad \text{avec } \mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k$$

2. Déterminer β_{k+1} (voir plus bas).

3. Déterminer la nouvelle direction de recherche :

$$\mathbf{p}_{k+1} = -\mathbf{g}(\mathbf{x}_{k+1}) + \beta_{k+1}\mathbf{p}_k$$

4. Aller à l'étape 1 pour $k = k+1$.

Quant au calcul de β_{k+1} dans l'étape 2, il existe plusieurs formules, proposés par des chercheurs dans le domaine. Les formules les plus connues sont celles de : Fletcher-Reeves (le travail original) et de Polak-Ribière. Leur point commun est qu'elles fournissent la solution optimale si la fonction de coût est parfaitement quadratique.

Résumons quelques propriétés des méthodes du gradient conjugué :

- Aucun calcul de dérivées secondes n'est nécessaire, seulement le gradient (et la fonction du coût elle-même) est calculé à chaque itération.
- En gros, un seul vecteur doit être stocké, contrairement au Hessien entier pour d'autres méthodes.
- La détermination de directions de recherche conjuguées permet une meilleure convergence que pour les méthodes du gradient.
- Tout comme pour les méthodes quasi-Newton, l'information acquise au cours des itérations (notamment les gradients) est prise en compte pour calculer les nouvelles directions de recherche.

Remarque. Il existe des méthodes hybrides entre quasi-Newton et gradient conjugué : au lieu de stocker une matrice entière (l'approximation du Hessien) ou un seul vecteur, un certain nombre de vecteurs est stocké, qui fait un compromis entre la taille de mémoire requise et le taux d'informations digérées , sur la forme de la fonction du coût. De telles méthodes sont nommées *Limited memory quasi-Newton methods* (LMQN).

8 Problèmes de moindres carrés – Introduction

Pour un nombre important de problèmes d'optimisation, la fonction de coût se présente sous la forme d'une somme de carrés de fonctions :

$$f(\mathbf{x}) = \frac{1}{2}\sum_{i=1}^{m}\left(r_i(\mathbf{x})\right)^2 \tag{11}$$

(le facteur $1/2$ est juste introduit pour avoir des expressions plus simples pour les dérivées). On appelle problèmes de moindres carrés des problèmes d'optimisation de ce type (*least squares problems* en anglais).

On rencontre cette formulation par exemple pour des problèmes d'ajustement de (paramètres de) modèles à des données observées ou mesurées. Typiquement, chaque fonction r_i correspond à une observation et est construite comme la différence entre une valeur prédite par le modèle théorique (dont les paramètres sont à estimer) et la valeur mesurée. On appelle les r_i donc aussi des *résidus*.

Il y aurait plusieurs possibilités de définir des fonctions de coût, par exemple on pourrait adopter la somme des valeurs absolues des r_i. Pourtant, la somme des carrés est normalement utilisées pour des raisons statistiques : il peut être prouvé que, si les valeurs des r_i (donc en gros, les erreurs faites dans les mesures) sont distribuées d'après une loi normale (ou Gaussienne), de manière identique et indépendante, alors le minimum de la somme des carrés donne la solution la plus probable (maximum de vraisemblance, *maximum likelihood*).

Les problèmes de la forme (**??**) pourraient être résolus avec les mêmes méthodes que d'autres problèmes d'optimisation. Pourtant, cette forme spécifique permet d'établir des méthodes faites sur mesure , comme on verra dans la suite.

Soit $\mathbf{r}(\mathbf{x})$ définie comme

$$\mathbf{r} : \mathcal{R}^n \rightarrow \mathcal{R}^m$$
$$\mathbf{r}(\mathbf{x}) = \begin{pmatrix} r_1(\mathbf{x}) \\ r_2(\mathbf{x}) \\ \vdots \\ r_m(\mathbf{x}) \end{pmatrix}$$

La fonction de coût f peut alors être écrite :

$$f(\mathbf{x}) = \frac{1}{2}\left(\|\mathbf{r}(\mathbf{x})\|_2\right)^2$$

où la norme vectorielle $\|\mathbf{v}\|_2$ (la norme L_2) est définie comme la racine carrée de la somme des carrés des coefficients du vecteur $\mathbf{v}$.

La *matrice Jacobienne* de la fonction vectorielle $\mathbf{r}$ est définie ainsi :

$$\mathsf{J}(\mathbf{x}) = \begin{pmatrix} \frac{\partial r_1}{\partial x_1}(\mathbf{x}) & \frac{\partial r_1}{\partial x_2}(\mathbf{x}) & \cdots & \frac{\partial r_1}{\partial x_n}(\mathbf{x}) \\ \frac{\partial r_2}{\partial x_1}(\mathbf{x}) & \frac{\partial r_2}{\partial x_2}(\mathbf{x}) & \cdots & \frac{\partial r_2}{\partial x_n}(\mathbf{x}) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial r_m}{\partial x_1}(\mathbf{x}) & \frac{\partial r_m}{\partial x_2}(\mathbf{x}) & \cdots & \frac{\partial r_m}{\partial x_n}(\mathbf{x}) \end{pmatrix}$$

(les lignes de la matrice Jacobienne sont les transposées des gradients $\mathbf{g}_i(\mathbf{x})$ des r_i). Dans la suite, nous appellerons la matrice Jacobienne tout court le *Jacobien*, tout en sachant qu'il y a abus de langage (le Jacobien est en fait le déterminant de la matrice Jacobienne, si elle est carrée).

Soit $\mathsf{R}_i(\mathbf{x})$ le Hessien de $r_i(\mathbf{x})$. Alors, nous obtenons pour le gradient et le Hessien de la fonction de coût $f(\mathbf{x})$:

$$\begin{aligned} \mathbf{g}(\mathbf{x}) &= \mathsf{J}(\mathbf{x})^{\mathsf{T}}\, \mathbf{r}(\mathbf{x}) && (12) \\ \mathsf{H}(\mathbf{x}) &= \mathsf{J}(\mathbf{x})^{\mathsf{T}}\, \mathsf{J}(\mathbf{x}) + \mathsf{Q}(\mathbf{x}) && (13) \\ \text{avec } \mathsf{Q}(\mathbf{x}) &= \sum_{i=1}^{m} r_i(\mathbf{x}) \mathsf{R}_i(\mathbf{x}) \end{aligned}$$

On peut constater que le Hessien est une combinaison de dérivées premières et secondes. Les méthodes d'optimisation de problèmes de type moindres carrés s'appuient typiquement sur l'hypothèse que le terme de premier ordre, $\mathsf{J}(\mathbf{x})^{\mathsf{T}}\, \mathsf{J}(\mathbf{x})$, domine le terme de deuxième ordre, $\mathsf{Q}(\mathbf{x})$. Cette hypothèse est justifiée si les résidus r_i sont effectivement petits, proche du minimum (les coefficients de $\mathsf{Q}(\mathbf{x})$ seront alors petits), ou bien si les r_i sont des fonctions à peu près linéaires (dans ce cas, leurs Hessiens R_i sont des matrices à peu près nulles). Une bonne approximation du Hessien peut alors être calculée à partir des seules dérivées premières !

On retrouve le même avantage qu'ont par exemple les méthodes quasi-Newton ou du gradient conjugué : pas de nécessité d'effectuer les calculs, parfois lourds, et nombreux (de l'ordre du carré du nombre des variables) des dérivées secondes. Contrairement à ces méthodes-là, l'approximation du Hessien pour un problème de moindres carrés est immédiate , et non accumulée au cours des itérations, et donc a priori plus fiable (si l'hypothèse de petits résidus est correcte).

Dans la suite, nous considérons d'abord le cas où les r_i sont des fonctions linéaires, puis le cas général.

9 Moindres carrés linéaires

Considérons des fonctions $r_i(\mathbf{x})$ *linéaires* :

$$r_i(\mathbf{x}) = \mathbf{a}_i^{\mathsf{T}} \mathbf{x} + b_i \ .$$

Leurs gradients sont donc des vecteurs *constants* (ils ne dépendent pas de $\mathbf{x}$) :

$$\mathbf{g}_i = \begin{pmatrix} a_{i1} \\ a_{i2} \\ \vdots \\ a_{in} \end{pmatrix} \ ,$$

où a_{ij} est le $j$coefficient de $\mathbf{a}_i$. Puisque les gradients sont constants, les Hessiens (les dérivées secondes) sont nuls.

Le Jacobien de la fonction de coût est également constant :

$$\mathsf{J} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix} \quad .$$

Soit $\mathbf{b} = (b_1, \ldots, b_m)^\mathsf{T}$. Le vecteur des résidus est alors donné par :

$$\mathbf{r}(\mathbf{x}) = \mathsf{J}\mathbf{x} + \mathbf{b} \quad .$$

La fonction de coût s'écrit :

$$f(\mathbf{x}) = \frac{1}{2} \left(\|\mathbf{r}(\mathbf{x})\|_2 \right)^2 = \frac{1}{2} \left(\|\mathsf{J}\mathbf{x} + \mathbf{b}\|_2 \right)^2 \tag{14}$$

Quant au gradient et au Hessien de la fonction de coût, on obtient d'après l'équation (**??**) :

$$\begin{aligned} \mathbf{g}(\mathbf{x}) &= \mathsf{J}^\mathsf{T} (\mathsf{J}\mathbf{x} + \mathbf{b}) \\ \mathsf{H} &= \mathsf{J}^\mathsf{T}\mathsf{J} \end{aligned}$$

Un minimum de $f(\mathbf{x})$ doit satisfaire $\mathbf{g}(\mathbf{x}^*) = \mathbf{0}$, donc :

$$\mathsf{J}^\mathsf{T}\mathsf{J}\mathbf{x}^* = -\mathsf{J}^\mathsf{T}\mathbf{b} \tag{15}$$

On appelle ces équations aussi les *équations normales* pour le problème (**??**).

Il y a différentes méthodes pour déterminer $\mathbf{x}^*$. La méthode qui semble évidente à première vue, consiste à résoudre les équations normales (**??**) via le calcul de l'inverse de $\mathsf{J}^\mathsf{T}\mathsf{J}$. Ce calcul est relativement simple[1], grâce au fait que la matrice est, par construction, symétrique et définie positive (si J n'est pas singulière) : des méthodes spécifiques existent pour ce cas (par exemple la décomposition de Cholesky, qui ne sera pas détaillée ici). Cette méthode est donc simple (et rapide), pourtant son application pose des problèmes si le Jacobien est mal conditionné (voir §**??**). En effet, le conditionnement de $\mathsf{J}^\mathsf{T}\mathsf{J}$ est deux fois pire que celui de J, donc en gros on peut s'attendre à perdre deux fois plus de chiffres significatifs correctes dans la solution, par rapport à la méthode discutée dans la suite.

Au lieu de résoudre les équations normales, on peut directement estimer $\mathbf{x}^*$ qui minimise l'équation (**??**), donc l'expression $\frac{1}{2} \left(\|\mathsf{J}\mathbf{x} + \mathbf{b}\|_2 \right)^2$. Un moyen pour ce faire utilise la décomposition en valeurs singulières du Jacobien J ; cette méthode est décrite au §**??**. L'avantage de travailler directement avec le Jacobien au lieu du produit $\mathsf{J}^\mathsf{T}\mathsf{J}$ est le meilleur conditionnement. Aussi, la décomposition en valeurs singulières permet de raisonner sur le rang de la matrice : si le Jacobien est singulier, il y a plusieurs solutions pour $\mathbf{x}$. L'analyse du rang permet d'en déterminer celle de la plus petite norme (voir §**??**).

[1] En théorie – en pratique il faut veiller à la stabilité numérique des calculs. Par exemple, la stabilité peut dépendre de l'ordre d'exécution de différents calculs.

10 Moindres carrés non-linéaires

10.1 La méthode de Gauß-Newton

Dans le cas de fonctions non-linéaires, nous utilisons l'*approximation* du Hessien obtenue à partir de l'équation (**??**), en négligeant les dérivées secondes représentées par $\mathsf{Q}(\mathbf{x})$, c'est-à-dire la matrice $\mathsf{J}(\mathbf{x}_k)^\mathsf{T}\ \mathsf{J}(\mathbf{x}_k)$. Nous pouvons alors adapter la méthode de Newton (cf. §**??**) pour le calcul des directions de recherche ce qui résulte en :

$$\mathbf{d}_k = -\Big(\mathsf{J}(\mathbf{x}_k)^\mathsf{T}\ \mathsf{J}(\mathbf{x}_k)\Big)^{-1}\mathsf{J}(\mathbf{x}_k)^\mathsf{T}\mathbf{r}(\mathbf{x}_k) \tag{16}$$

Le calcul de la direction sera suivi par le calcul de la longueur de pas, comme vu à plusieurs reprises précédemment.

Cette méthode est appellée la *méthode Gauß-Newton*.

On s'aperçoit que (**??**) représente les *équations normales* du problème de moindres carrés linéaires (cf. les équations (**??**) et (**??**)) :

$$\min_{\mathbf{d}_k}\frac{1}{2}\left(\|\mathsf{J}(\mathbf{x}_k)\mathbf{d}_k + \mathbf{r}(\mathbf{x}_k)\|_2\right)^2$$

Les remarques faites en §**??**, concernant la résolution des équations normales, s'appliquent donc aussi ici. Pour résumer : la méthode de Gauß-Newton pour minimiser un système *non-linéaire* aux moindres carrés, contient, à chaque itération, la résolution d'un système *linéaire* aux moindres carrés.

10.2 La méthode de Levenberg-Marquardt

Une alternative populaire à la méthode de Gauß-Newton est celle de *Levenberg-Marquardt*. La direction de recherche y est calculée en résolvant le système :

$$\Big(\mathsf{J}(\mathbf{x}_k)^\mathsf{T}\ \mathsf{J}(\mathbf{x}_k) + \lambda_k\mathtt{I}\Big)\,\mathbf{d}_k = -\mathsf{J}(\mathbf{x}_k)^\mathsf{T}\mathbf{r}(\mathbf{x}_k) \tag{17}$$

où λ_k est un scalaire non negatif et $\mathtt{I}$ la matrice identité de la taille appropriée. La méthode de Levenberg-Marquardt ne nécessite pas de calcul de longueur de pas : le nouveau point est directement obtenu par $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{d}_k$.

Il peut être montré que, pour un certain scalaire Δ qui dépend de λ_k, le vecteur $\mathbf{d}_k$ est la solution du sous-problème contraint :

$$\begin{aligned}&\min_{\mathbf{d}_k}\frac{1}{2}\left(\|\mathsf{J}(\mathbf{x}_k)\mathbf{d}_k + \mathbf{r}(\mathbf{x}_k)\|_2\right)^2\\ &\text{sous la contrainte } \|\mathbf{d}_k\|_2 \leq \Delta\end{aligned}$$

Donc, $\mathbf{d}_k$ est le meilleur vecteur de mise à jour avec une longueur bornée par un seuil. Les scalaires λ_k ne sont pas identiques, mais typiquement modifiés selon le progrès fait au cours de la dernière itération (réduction de la valeur de la fonction de coût).

Cette méthode est en quelque sorte un mélange entre les méthodes du gradient et de Gauß-Newton : si λ_k est très petit, alors la direction calculée correspondra à celle de Gauß-Newton ; si λ_k est très grand, la direction correspondra au gradient (cf. l'équation (**??**)). Qualitativement, on peut expliquer ce mélange ainsi : loin de la solution, les résidus ne seront pas encore très petits. Donc, l'approximation du Hessien utilisée par la méthode Gauß-Newton n'est pas encore fiable et on va préférer la direction de descente donnée par le gradient (λ_k sera grand). Plus on converge, plus on peut faire confiance à la direction Gauß-Newton, ce que l'on obtient en diminuant λ_k.

Remarque. Cette méthode est un bon exemple (en fait, elle en est le précurseur) des méthodes du type *trust-region*. Ci-dessus, la trust-region est une sphère (en n dimensions) de diamètre Δ centré en $\mathbf{x}_k$. Des régions plus générales, adaptées à la forme de la fonction de coût, sont possibles (e.g. des ellipsoïdes).

Le deuxième type principal pour des méthodes d'optimisation basées sur une recherche, est constitué par les méthodes de type *line-search*. Toutes les méthodes discutées jusqu'ici, à l'exception de Levenberg-Marquardt, sont de ce type. En gros, ces méthodes cherchent la longueur de pas optimale pour une direction de recherche donnée, tandis que les méthodes trust-region cherchent la direction optimale, pour une longueur de pas (maximale) donnée.

11 Conditionnement d'un système linéaire

Considérons le système linéaire suivant :

$$\begin{pmatrix} 10 & 7 & 8 & 7 \\ 7 & 5 & 6 & 5 \\ 8 & 6 & 10 & 9 \\ 7 & 5 & 9 & 10 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 32 \\ 23 \\ 33 \\ 31 \end{pmatrix} \quad , \quad \text{de solution} \quad \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

et considérons le système perturbé , où les seconds membres ont été légèrement modifiés, la matrice restant inchangée :

$$\begin{pmatrix} 10 & 7 & 8 & 7 \\ 7 & 5 & 6 & 5 \\ 8 & 6 & 10 & 9 \\ 7 & 5 & 9 & 10 \end{pmatrix} \begin{pmatrix} x_1 + \delta x_1 \\ x_2 + \delta x_2 \\ x_3 + \delta x_3 \\ x_4 + \delta x_4 \end{pmatrix} = \begin{pmatrix} 32.1 \\ 22.9 \\ 33.1 \\ 30.9 \end{pmatrix} \quad , \quad \text{de solution} \quad \begin{pmatrix} 9.2 \\ -12.6 \\ 4.5 \\ -1.1 \end{pmatrix} \tag{18}$$

Une erreur relative de l'ordre de $1/200$ sur les données entraîne une erreur relative de l'ordre de $10/1$ sur le résultat – les erreurs relatives sont amplifiées par un facteur de l'ordre de 2000 !

Considérons également le système perturbé où, cette fois, ce sont les éléments de la matrice qui ont été modifiés :

$$\begin{pmatrix} 10 & 7 & 8.1 & 7.2 \\ 7.08 & 5.04 & 6 & 5 \\ 8 & 5.98 & 9.89 & 9 \\ 6.99 & 4.99 & 9 & 9.98 \end{pmatrix} \begin{pmatrix} x_1 + \Delta x_1 \\ x_2 + \Delta x_2 \\ x_3 + \Delta x_3 \\ x_4 + \Delta x_4 \end{pmatrix} = \begin{pmatrix} 32 \\ 23 \\ 33 \\ 31 \end{pmatrix} \quad , \quad \text{de solution} \quad \begin{pmatrix} -81 \\ 137 \\ -34 \\ 22 \end{pmatrix}$$

Sans commentaire ...

Un système linéaire de ce type est dit mal conditionné . Il y a en effet beaucoup de définitions possibles du conditionnement, qui dépendent du problème à résoudre – de manière générale on peut dire qu'un problème est mal conditionné si de petites erreurs/perturbations dans les données provoquent de grands changements dans les résultats.

On peut essayer de quantifier le conditionnement d'un système linéaire. Considérons le système dans (**??**). Il peut être prouvé que la borne supérieure suivante existe sur l'erreur relative dans le résultat :

$$\frac{\|\delta \mathbf{x}\|}{\|\mathbf{x}\|} \leq \|\mathsf{A}\| \|\mathsf{A}^{-1}\| \frac{\|\delta \mathbf{b}\|}{\|\mathbf{b}\|}$$

où A est la matrice du système, $\mathbf{b}$ le vecteur du côté droit et $\delta\mathbf{b}$ le vecteur des perturbations sur $\mathbf{b}$. Comme norme $\|\cdot\|$, toute norme matricielle/vectorielle peut être choisie. Cette borne n'est que supérieure, donc l'erreur dans le résultat *peut* être aussi grande. Pourtant, cette borne est la meilleure possible qu'on puisse établir en général.

Pour une matrice A inversible, on peut alors définir le *conditionnement* (anglais : *condition number*) :

$$\text{cond}(\mathsf{A}) = \|\mathsf{A}\| \|\mathsf{A}^{-1}\| \tag{19}$$

On dit qu'un système linéaire est bien (respectivement mal) conditionné, selon que le conditionnement de sa matrice est petit (respectivement grand).

12 Décomposition en valeurs singulières

La *décomposition en valeurs singulières* (anglais : singular value decomposition – SVD) d'une matrice A de dimension $m \times n$ est un produit de matrices :

$$\mathsf{A}_{m\times n} = \mathsf{U}_{m\times n}\Sigma_{n\times n}\mathsf{V}^{\mathsf{T}}_{n\times n}$$

où :

- U est orthogonale : $\mathsf{U}^{\mathsf{T}}\,\mathsf{U} = \mathtt{I}$;
- V est orthogonale : $\mathsf{V}^{\mathsf{T}}\,\mathsf{V} = \mathtt{I}$;
- Σ est diagonale : $\Sigma = \text{diag}(\sigma_1, \ldots, \sigma_n)$. Normalement, on suppose que les *valeurs singulières* σ_i sont ordonnées : $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_n \geq 0$.

La décomposition en valeurs singulières est très utile pour analyser le rang d'une matrice : si A est de rang r, alors seules les r premières valeurs singulières seront non nulles. En outre, une base orthogonale du *noyau* de A (l'ensemble des vecteurs $\mathbf{v}$ avec $\mathsf{A}\mathbf{v} = \mathbf{0}$) est donnée par les $n - r$ derniers vecteurs colonne de V.

Si la précision des calculs n'est que finie (nombre flottants par exemple), le concept de rang doit être considéré avec prudence, puisque, en général, même une matrice parfaitement singulière n'aura, une fois stockée en flottants, que des valeurs singulières non nulles (à cause des erreurs d'arrondi). Par conséquent, on ne peut plus déterminer le vrai rang de la matrice, en comptant les valeurs singulières qui sont exactement nulles.

12.1 Résolution aux moindres carrés d'un système linéaire et homogène

Considérons la résolution aux moindres carrés d'un système linéaire et *homogène* :

$$\min_{\mathbf{x}} \left(\|\mathsf{A}\mathbf{x}\|_2\right)^2 \quad .$$

Notons tout d'abord qu'il faudra éviter la solution triviale $\mathbf{x} = \mathbf{0}$. Typiquement, ceci se fait en imposant une contrainte sur la norme de la solution :

$$\left(\|\mathbf{x}\|_2\right)^2 = 1 \quad .$$

La résolution de ce problème d'optimisation contrainte est particulièrement simple avec la décomposition en valeurs singulières de A. Soit $\mathsf{A} = \mathsf{U}\Sigma\mathsf{V}^{\mathsf{T}}$. La solution du système est alors donnée par :

$$\mathbf{x}^* = \mathbf{v}_n$$

où $\mathbf{v}_n$ est la dernière colonne de V (la contrainte $\|\mathbf{x}^*\|_2 = 1$ est satisfaite de part l'orthogonalité de V).

Nous en esquissons la preuve. La matrice V étant orthogonale, ses colonnes constituent une base de l'espace $\mathcal{R}^n$, donc :

$$\mathbf{x} = \sum_{i=1}^{n} \lambda_i \mathbf{v}_i$$

pour des λ_i réels. Afin d'éviter la solution triviale $\lambda_i = 0, \forall i$, on impose $\left(\|\mathbf{x}\|_2\right)^2 = 1$, ce qui revient (grâce à l'orthogonalité de V) à imposer :

$$\sum_{i=1}^{n} \lambda_i^2 = 1 \tag{20}$$

Donc,

$$\begin{aligned}
\mathsf{A}\mathbf{x} &= \begin{pmatrix}\mathbf{u}_1 & \mathbf{u}_2 & \cdots & \mathbf{u}_n\end{pmatrix} \begin{pmatrix}\sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_n\end{pmatrix} \begin{pmatrix}\mathbf{v}_1^\mathsf{T} \\ \mathbf{v}_2^\mathsf{T} \\ \vdots \\ \mathbf{v}_n^\mathsf{T}\end{pmatrix} \sum_{i=1}^{n} \lambda_i \mathbf{v}_i \\
&= \begin{pmatrix}\sigma_1\mathbf{u}_1 & \sigma_2\mathbf{u}_2 & \cdots & \sigma_n\mathbf{u}_n\end{pmatrix} \begin{pmatrix}\mathbf{v}_1^\mathsf{T} \sum_{i=1}^{n} \lambda_i \mathbf{v}_i \\ \mathbf{v}_2^\mathsf{T} \sum_{i=1}^{n} \lambda_i \mathbf{v}_i \\ \vdots \\ \mathbf{v}_n^\mathsf{T} \sum_{i=1}^{n} \lambda_i \mathbf{v}_i\end{pmatrix}
\end{aligned}$$

La matrice V étant orthogonale, les produits scalaires $\mathbf{v}_i^\mathsf{T}\mathbf{v}_j$ sont égaux à 1 si $i = j$ et à 0 si $i \neq j$. Il en découle que :

$$\begin{aligned}
\mathsf{A}\mathbf{x} &= \begin{pmatrix}\sigma_1\mathbf{u}_1 & \sigma_2\mathbf{u}_2 & \cdots & \sigma_n\mathbf{u}_n\end{pmatrix} \begin{pmatrix}\lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_n\end{pmatrix} \\
&= \sum_{i=1}^{n} \lambda_i \sigma_i \mathbf{u}_i
\end{aligned}$$

Le carré de la norme de ce vecteur, $\left(\|\cdot\|_2\right)^2$, est donné par (ceci est obtenu en utilisant le fait que la norme de chacune des colonnes $\mathbf{u}_i$ de U vaut 1, puisque U est orthogonale) :

$$\left(\|\mathsf{A}\mathbf{x}\|_2\right)^2 = \lambda_1^2\sigma_1^2 + \ldots + \lambda_n^2\sigma_n^2$$

En respectant la contrainte (**??**) et en prenant en compte que σ_n est la plus petite valeur singulière, il est clair que la plus petite norme de $\mathsf{A}\mathbf{x}$ est obtenue en adoptant $\lambda_i = 0, \forall i < n$ et $\lambda_n = \pm 1$. Donc $\mathbf{x}^*$ est donné par la dernière colonne de V.

12.2 Résolution des équations normales (??) en utilisant la décomposition en valeurs singulières

Illustrons les conséquences possibles de ce problème, en résolvant les équations normales (**??**) à l'aide de la décomposition en valeurs singulières de J. Soit

$$\mathsf{J} = \mathsf{U}\Sigma\mathsf{V}^{\mathsf{T}}$$

La solution directe des équations normales $\mathsf{J}^{\mathsf{T}}\mathsf{J}\mathbf{x}^* = -\mathsf{J}^{\mathsf{T}}\mathbf{b}$ est donnée par (dans ces calculs, nous utilisons le fait que l'inverse d'une matrice orthogonale carrée, est sa transposée) :

$$\begin{aligned}
\mathbf{x}^* &= -\left(\mathsf{J}^{\mathsf{T}}\mathsf{J}\right)^{-1}\mathsf{J}^{\mathsf{T}}\mathbf{b} \\
&= -\left(\mathsf{V}\Sigma\underbrace{\mathsf{U}^{\mathsf{T}}\mathsf{U}}_{\mathtt{I}}\Sigma\mathsf{V}^{\mathsf{T}}\right)^{-1}\mathsf{V}\Sigma\mathsf{U}^{\mathsf{T}}\mathbf{b} \\
&= -\left(\mathsf{V}\Sigma\Sigma\mathsf{V}^{\mathsf{T}}\right)^{-1}\mathsf{V}\Sigma\mathsf{U}^{\mathsf{T}}\mathbf{b} \\
&= -\mathsf{V}\Sigma^{-1}\Sigma^{-1}\underbrace{\mathsf{V}^{\mathsf{T}}\mathsf{V}}_{\mathtt{I}}\Sigma\mathsf{U}^{\mathsf{T}}\mathbf{b} \\
&= -\mathsf{V}\Sigma^{-1}\Sigma^{-1}\Sigma\mathsf{U}^{\mathsf{T}}\mathbf{b} \\
&= -\mathsf{V}\Sigma^{-1}\mathsf{U}^{\mathsf{T}}\mathbf{b}
\end{aligned}$$

Nous constatons que la solution n'a pas recours au produit $\mathsf{J}^{\mathsf{T}}\mathsf{J}$ – seule la décomposition de la matrice J est utilisée. Le vecteur $\mathbf{x}^*$ ainsi déterminé minimise alors $f(\mathbf{x}) = \frac{1}{2}\left(\|\mathsf{J}\mathbf{x} + \mathbf{b}\|_2\right)^2$. Si l'on note par $\mathbf{u}_i$ respectivement $\mathbf{v}_i$, la $i$colonne de U respectivement V, on peut écrire :

$$\mathbf{x}^* = -\sum_{i=1}^{n}\frac{1}{\sigma_i}\left(\mathbf{u}_i^{\mathsf{T}}\mathbf{b}\right)\mathbf{v}_i \tag{21}$$

Evidemment, si J était exactement singulière, au moins σ_n serait nul, et la solution ne serait pas définie (puisque nous avons inversé une matrice singulière (Σ) ou bien, effectué une division par zéro). Dans ce cas-là, il existe en effet plusieurs solutions pour $\mathbf{x}^*$, et il peut être montré que tous les vecteurs $\mathbf{x}^*$ de la forme :

$$\mathbf{x}^* = -\sum_{\sigma_i\neq 0}\frac{1}{\sigma_i}\left(\mathbf{u}_i^{\mathsf{T}}\mathbf{b}\right)\mathbf{v}_i - \sum_{\sigma_j=0}\tau_j\mathbf{v}_j$$

(pour toutes les valeurs possibles des τ_i) sont des solutions.

Il peut également être montré que la solution avec la plus petite norme est donnée en adoptant $\tau_j = 0, \forall j$ avec $\sigma_j = 0$, alors :

$$\mathbf{x}^* = -\sum_{\sigma_i\neq 0}\frac{1}{\sigma_i}\left(\mathbf{u}_i^{\mathsf{T}}\mathbf{b}\right)\mathbf{v}_i$$

Considérons maintenant le cas où J aurait quelques valeurs singulières très petites – proches de la précision de la machine. Ces valeurs seront alors particulièrement affectées par les

erreurs d'arrondi. Pourtant, leur contribution à la solution $\mathbf{x}^*$ est énorme, puisqu'elle se fait par des termes $\frac{1}{\sigma_i}$. Les erreurs d'arrondi se trouveront alors multipliées, éventuellement à un tel point de rendre la solution complètement imprécise et inutilisable.

En pratique, on pourra alors tenter de négliger, dans l'équation (**??**), les termes correspondant à de petites valeurs singulières :

$$\mathbf{x}^* = -\sum_{i \leq \mathrm{rg}(\mathsf{J})} \frac{1}{\sigma_i} \left(\mathbf{u}_i^{\mathsf{T}} \mathbf{b}\right) \mathbf{v}_i$$

où $\mathrm{rg}(\mathsf{J})$ est le rang de J – le nombre de valeurs singulières qu'on considère non nulles (non petites).

Comment décider si une valeur singulière est petite ou non, est très délicat, et il n'y a effectivement pas de moyen parfait pour le faire. Par exemple, on ne peut pas utiliser de seuil : les valeurs singulières du système $\lambda \mathsf{A}\mathbf{x}' + \mathbf{b} = \mathbf{0}$ sont λ fois plus grandes que celles du système $\mathsf{A}\mathbf{x} + \mathbf{b} = \mathbf{0}$, mais le rang de A devrait être le même afin d'obtenir des solutions avec $\mathbf{x} = \lambda \mathbf{x}' \ldots$

12.3 Conditionnement et décomposition en valeurs singulières

Si la norme matricielle utilisée pour définir le conditionnement (**??**) est la norme de Frobenius (la racine de la somme des carrés des coefficients de la matrice), alors le conditionnement est donné par les valeurs singulières de A :

$$\mathrm{cond}(\mathsf{A}) = \sqrt{\left(\sum \sigma_i^2\right)\left(\sum \frac{1}{\sigma_i^2}\right)}$$

Si toutes les valeurs singulières sont identiques, le conditionnement est égal à n (le meilleur cas possible). Si par contre la plus grande respectivement la plus petite valeur singulière est égale à 10^a respectivement 10^{-a}, le conditionnement sera supérieur à $10^{2a} \ldots$

13 Optimisation sous contraintes linéaires – Conditions d'optimalité et multiplicateurs de Lagrange

Dans cette section et la suivante, nous considérons des problèmes d'optimisation qui sont *contraints* par des fonctions linéaires, e.g. :

trouver $\mathbf{x}$ qui minimise la valeur de la fonction	$f(\mathbf{x})$	$\mathbf{x} = (x_1, x_2, \ldots, x_n)^{\mathsf{T}}$
sous les contraintes	$c_i(\mathbf{x}) = 0$	$i = 1, 2, \ldots, m'$
	$c_i(\mathbf{x}) \geq 0$	$i = m'+1, m'+2, \ldots, m$

où les c_i sont linéaires en les x_j :

$$c_i(\mathbf{x}) = -b_i + \sum_{j=1}^{n} a_{ij} x_j = -b_i + \mathbf{a}_i^{\mathsf{T}} \mathbf{x} \tag{22}$$

pour des scalaires réels a_{ij} et b_i.

Des contraintes particulièrement simples sont bien sûr celles ne concernant qu'une seule variable :

$$\begin{aligned} x_j &= b_i \\ x_j &\geq b_i \\ x_j &\leq b_i \end{aligned}$$

Les deux derniers cas représentent alors des bornes sur les valeurs d'une variable.

Dans les sections suivantes, nous dérivons des conditions nécessaires et suffisantes pour des minima locaux du problème exposé ci-dessus. Notons U l'ensemble de tous les points qui satisfont toutes les contraintes ; nous supposons que $U \neq \{\}$. Les minima recherchés sont donc des minima locaux par rapport à U. Dans le cas présent, où U est défini par des contraintes sur les variables, on parle aussi de *minima locaux liés*.

13.1 Contraintes d'égalité

Dans cette section, nous ne considérons que des problèmes avec des contraintes d'égalité, i.e. :

$$\begin{array}{ll} \text{trouver } \mathbf{x} \text{ qui minimise la valeur de la fonction} & f(\mathbf{x}) \\ \text{sous les contraintes} & \mathsf{A}\mathbf{x} = \mathbf{b} \end{array} \tag{23}$$

où la matrice A et le vecteur $\mathbf{b}$ sont formés par les a_{ij} et b_i de l'équation (**??**).

Le but de cette section est de dériver des conditions d'optimalité. Un minimum local $\mathbf{x}^*$ du problème (**??**) doit bien sûr être dans U, et il faut que $f(\mathbf{x}^*) \leq f(\mathbf{y})$ pour tous les $\mathbf{y} \in U \cap V$, où V est un environnement de $\mathbf{x}^*$. Afin d'établir des conditions d'optimalité, nous considérons d'abord des moyens pour caractériser l'ensemble $U \cap V$.

Supposons que $\mathbf{x}$ et $\mathbf{y}$ soient deux points dans U, alors : $\mathsf{A}(\mathbf{y}-\mathbf{x}) = \mathsf{A}\mathbf{y} - \mathsf{A}\mathbf{x} = \mathbf{b} - \mathbf{b} = \mathbf{0}$. Donc, afin d'atteindre, en partant d'un point $\mathbf{x} \in U$, un autre point $\mathbf{y}$ dans U, il faut prendre une direction $\mathbf{d} = \mathbf{y} - \mathbf{x}$ avec :

$$\mathsf{A}\mathbf{d} = \mathbf{0} \ .$$

Généralement, si $\mathbf{x} \in U$:

$$\begin{aligned} \mathbf{x} + \mathbf{d} \in U &\Leftrightarrow \mathsf{A}\mathbf{d} = \mathbf{0} \\ \mathbf{x} + \mathbf{d} \notin U &\Leftrightarrow \mathsf{A}\mathbf{d} \neq \mathbf{0} \end{aligned}$$

Les directions acceptables forment donc le noyau de la matrice des contraintes A. Supposons que les vecteurs $\mathbf{z}_i$ (où $i = 1, \ldots, \dim(\ker(\mathsf{A}))$) constituent une base du noyau ($\ker(\mathsf{A})$ désigne le noyau de A). Soit Z la matrice dont les colonnes sont les $\mathbf{z}_i$. Toute direction acceptable peut alors être écrite ainsi :

$$\mathbf{d} = \mathsf{Z}\mathbf{s} \tag{24}$$

pour un vecteur $\mathbf{s}$ de coefficients de longueur appropriée.

Considérons maintenant l'environnement d'un point $\mathbf{x}$, en effet seulement la partie de l'environnement qui est contenue dans U, donc des points $\mathbf{x} + \epsilon\mathsf{Z}\mathbf{s}$. La valeur de la fonction de coût est, au deuxième ordre :

$$f(\mathbf{x} + \epsilon\mathsf{Z}\mathbf{s}) = f(\mathbf{x}) + \epsilon\, \mathbf{s}^\mathsf{T}\mathsf{Z}^\mathsf{T}\mathbf{g}(\mathbf{x}) + O(\epsilon^2)$$

S'il existe $\mathbf{s}$ tel que $\mathbf{s}^\mathsf{T}\mathsf{Z}^\mathsf{T}\mathbf{g}(\mathbf{x})$ est non nul, alors l'environnement de $\mathbf{x}$ contient des points (se trouvant dans la direction $\mathsf{Z}\mathbf{s}$) qui sont dans U et où la valeur de la fonction de coût est plus petite qu'à $\mathbf{x}$: si ϵ est suffisamment petit, et du signe opposé à celui de $\mathbf{s}^\mathsf{T}\mathsf{Z}^\mathsf{T}\mathbf{g}(\mathbf{x})$, alors $\epsilon\, \mathbf{s}^\mathsf{T}\mathsf{Z}^\mathsf{T}\mathbf{g}(\mathbf{x}) + O(\epsilon^2) < 0$.

Pour que $\mathbf{x}^*$ soit un minimum local (par rapport à U), il faut donc que $\mathbf{s}^\mathsf{T}\mathsf{Z}^\mathsf{T}\mathbf{g}(\mathbf{x}^*)$ soit nul pour tout vecteur $\mathbf{s}$, ce qui implique que :

$$\mathsf{Z}^\mathsf{T}\mathbf{g}(\mathbf{x}^*) = \mathbf{0} \tag{25}$$

Le vecteur $\mathsf{Z}^\mathsf{T}\mathbf{g}(\mathbf{x}^*)$ est aussi appelé le *gradient projeté* de f au point $\mathbf{x}^*$.

La condition (**??**) implique que $\mathbf{g}(\mathbf{x}^*)$ n'est pas dans le noyau de A (dont Z contient la base), alors $\mathbf{g}(\mathbf{x}^*)$ doit être une combinaison linéaire des lignes de A :

$$\exists\lambda^* = (\lambda_1^*, \ldots, \lambda_m^*)^\mathsf{T} \text{ avec } \quad \mathbf{g}(\mathbf{x}^*) = \sum_{i=1}^{m} \mathbf{a}_i\lambda_i^* = \mathsf{A}^\mathsf{T}\lambda^* \tag{26}$$

Le vecteur λ^* est appelé le vecteur des *multiplicateurs de Lagrange*. Les multiplicateurs de Lagrange λ_i^* sont uniques seulement si les lignes de A sont linéairement indépendantes.

Les équations (**??**) et (**??**) sont des conditions *nécessaires* pour un minimum local (en effet, elles indiquent que $\mathbf{x}^*$ est un point stationnaire). D'autres conditions nécessaires pour un minimum sont :

$$\begin{aligned} &\mathsf{A}\mathbf{x}^* = \mathbf{b} \\ &\mathsf{Z}^\mathsf{T}\mathsf{H}(\mathbf{x}^*)\mathsf{Z} \text{ est positive} \end{aligned}$$

(une matrice X est positive si $\mathbf{v}^\mathsf{T}\mathsf{X}\mathbf{v} \geq 0, \forall\mathbf{v}$). La matrice $\mathsf{Z}^\mathsf{T}\mathsf{H}(\mathbf{x}^*)\mathsf{Z}$ est aussi appelée le *Hessien projeté*.

Des **conditions suffisantes** pour un minimum local sont :

$$\begin{aligned} &\mathsf{Z}^\mathsf{T}\mathbf{g}(\mathbf{x}^*) = \mathbf{0} \quad \text{ ou bien } \quad \exists\lambda^* = (\lambda_1^*, \ldots, \lambda_m^*)^\mathsf{T} \text{ avec } \quad \mathbf{g}(\mathbf{x}^*) = \mathsf{A}^\mathsf{T}\lambda^* \\ &\mathsf{A}\mathbf{x}^* = \mathbf{b} \\ &\mathsf{Z}^\mathsf{T}\mathsf{H}(\mathbf{x}^*)\mathsf{Z} \text{ est } \textit{définie positive} \end{aligned}$$

13.2 Contraintes d'inégalité

Dans cette section, nous ne considérons que des problèmes avec des contraintes d'inégalité, i.e. :

$$\begin{aligned} &\text{trouver } \mathbf{x} \text{ qui minimise la valeur de la fonction} && f(\mathbf{x}) \\ &\text{sous les contraintes} && \mathsf{A}\mathbf{x} \geq \mathbf{b} \end{aligned} \tag{27}$$

où la relation $\mathbf{v} \geq \mathbf{w}$ signifie que $v_i \geq w_i, \forall i = 1, \ldots, m$.

Tout comme pour le cas de contraintes d'égalité, nous examinons des conditions pour qu'un point dans un environnement d'un point $\mathbf{x}^* \in U$, soit également dans U. Pour ce faire, il est important de distinguer les contraintes qui sont satisfaites *exactement* (il y a égalité) des contraintes qui ne le sont pas. Une contrainte c_i est *active* au point $\mathbf{x}^*$ si $\mathbf{a}_i^\mathsf{T}\mathbf{x}^* = b_i$, et *inactive* si $\mathbf{a}_i^\mathsf{T}\mathbf{x}^* > b_i$. Si $\mathbf{a}_i^\mathsf{T}\mathbf{x}^* < b_i$, la contrainte est *violée*.

Les contraintes actives jouent un rôle important puisqu'elles limitent le nombre de directions dans lesquelles on peut trouver des points dans U, en partant de $\mathbf{x}^*$. Si la contrainte c_i est *inactive* à $\mathbf{x}^*$, on peut trouver, dans n'importe quelle direction en partant de $\mathbf{x}^*$, des points qui satisfont cette contrainte (pour des longueurs de pas suffisamment petites).

Considérons donc de plus près le cas d'un point $\mathbf{x}^*$ où la contrainte c_i est active. Si l'on se déplace dans une direction $\mathbf{d}$ avec

$$\mathbf{a}_i^\mathsf{T}\mathbf{d} = 0$$

la contrainte reste active :

$$\mathbf{a}_i^\mathsf{T}(\mathbf{x}^* + \mathbf{d}) = \mathbf{a}_i^\mathsf{T}\mathbf{x}^* + \mathbf{a}_i^\mathsf{T}\mathbf{d} = \mathbf{0}$$

Si, par contre, la direction vérifie

$$\mathbf{a}_i^\mathsf{T}\mathbf{d} > 0$$

la contrainte devient inactive. Un mouvement dans toute autre direction mènera à la violation de la contrainte.

Afin de déterminer si le point $\mathbf{x}^*$ est un minimum local (par rapport à U), il ne faut donc considérer que les contraintes qui y sont actives. Supposons que les contraintes actives sont représentées par $\hat{\mathsf{A}}$ et $\hat{\mathbf{b}}$, sous-matrice respectivement sous-vecteur de A respectivement $\mathbf{b}$:

$$\hat{\mathsf{A}}\mathbf{x} = \hat{\mathbf{b}} \ .$$

Il faut prendre en compte les deux types de directions montrés ci-dessus. Quant au premier type (directions dans lesquelles toutes les contraintes actives restent actives), on trouve les mêmes conditions que pour le cas de contraintes d'égalité :

$$\exists \lambda^* = (\lambda_1^*, \ldots, \lambda_m^*)^\mathsf{T} \text{ avec } \quad \mathbf{g}(\mathbf{x}^*) = \sum_{i=1}^{m} \hat{\mathbf{a}}_i \lambda_i^* = \hat{\mathsf{A}}^\mathsf{T}\lambda^* \tag{28}$$

Considérons le deuxième type de direction, $\mathbf{d}$ avec $\hat{\mathsf{A}}\mathbf{d} \geq \mathbf{0}$. Pour qu'aucune telle direction ne permette d'atteindre une valeur de la fonction de coût plus basse qu'à $\mathbf{x}^*$, il faut que :

$$\mathbf{g}(\mathbf{x}^*)^\mathsf{T}\mathbf{d} \geq 0, \quad \forall \mathbf{d} \text{ avec } \hat{\mathsf{A}}\mathbf{d} \geq \mathbf{0}$$

La combinaison de cette contrainte avec (**??**) donne :

$$\mathbf{g}(\mathbf{x}^*)^\mathsf{T}\mathbf{d} = \lambda_1^*\hat{\mathbf{a}}_1^\mathsf{T}\mathbf{d} + \ldots + \lambda_m^*\hat{\mathbf{a}}_m^\mathsf{T}\mathbf{d} \geq 0, \quad \forall \mathbf{d} \text{ avec } \hat{\mathsf{A}}\mathbf{d} \geq \mathbf{0}$$

Puisque tous les termes $\hat{\mathbf{a}}_i^\mathsf{T}\mathbf{d}$ sont non négatifs ($\mathbf{d}$ vérifie $\hat{\mathsf{A}}\mathbf{d} \geq \mathbf{0}$), cette contrainte n'est satisfaite que si tous les multiplicateurs de Lagrange sont non négatifs eux aussi (dans le cas contraire, on trouverait un $\mathbf{d}$ tel que $\mathbf{g}(\mathbf{x}^*)^\mathsf{T}\mathbf{d} < 0$) :

$$\lambda_i^* \geq 0, \quad \forall i = 1, \ldots, m$$

Les **conditions nécessaires** pour un minimum local $\mathbf{x}^*$ par rapport à U sont alors :

$$\begin{aligned}
&\mathsf{A}\mathbf{x}^* \geq \mathbf{b}, \qquad \text{avec } \hat{\mathsf{A}}\mathbf{x}^* = \hat{\mathbf{b}} \\
&\exists \lambda_1^* \geq 0, \ldots, \lambda_m^* \geq 0 \text{ avec } \quad \mathbf{g}(\mathbf{x}^*) = \sum_{i=1}^{m} \hat{\mathbf{a}}_i \lambda_i^* = \hat{\mathsf{A}}^\mathsf{T}\lambda^* \\
&\mathsf{Z}^\mathsf{T}\mathsf{H}(\mathbf{x}^*)\mathsf{Z} \text{ est positive}
\end{aligned}$$

Notons la contrainte de positivité des multiplicateurs de Lagrange, qui n'existe pas dans le cas des contraintes d'égalité.

L'établissement de conditions suffisantes est plus délicat que dans le cas de contraintes d'égalité, et ne sera pas détaillé ici.

13.3 Contraintes mixtes

Les conditions caractérisant les minima dans le cas de contraintes mixtes, sont un mélange des conditions obtenues pour les cas individuels. Les contraintes actives comprennent alors aussi les contraintes d'égalité, mais il n'y a pas de restriction de signe sur les multiplicateurs de Lagrange y correspondant (contrairement aux contraintes d'inégalité actives).

14 Méthodes d'optimisation sous contraintes linéaires

14.1 Méthodes pour contraintes d'égalité

14.1.1 Fonction de coût quadratique

Considérons un problème avec seulement des contraintes d'égalité. Une méthode de le résoudre consiste à transformer le problème *primal* en un problème *dual* : on inclut les multiplicateurs de Lagrange dans l'ensemble des inconnues, et résoud le système suivant, basé sur deux des conditions nécessaires d'un minimum local :

$$\begin{aligned}
\mathsf{A}\mathbf{x} &= \mathbf{b} \\
\mathbf{g}(\mathbf{x}) - \mathsf{A}^\mathsf{T}\lambda &= \mathbf{0}
\end{aligned}$$

La solution du problème dual donne alors la solution du problème primal. Dans le cas d'une fonction de coût quadratique, le problème dual ne contient que des équations linéaires en les inconnues.

Exemple. Soit le problème original :

$$\begin{aligned} \mathbf{x} &= (x_1, x_2, x_3)^\mathsf{T} \\ f(\mathbf{x}) &= x_1^2 + x_2^2 + x_3^2 \\ c_1(\mathbf{x}) &= x_1 + x_2 + x_3 - 1 = 0 \end{aligned}$$

Donc :

$$\begin{aligned} \mathsf{A} &= \begin{pmatrix} 1 & 1 & 1 \end{pmatrix} \\ \mathbf{b} &= \begin{pmatrix} 1 \end{pmatrix} \\ \mathbf{g}(\mathbf{x}) &= \begin{pmatrix} 2x_1 \\ 2x_2 \\ 2x_3 \end{pmatrix} \end{aligned}$$

Le problème dual est alors :

$$\begin{aligned} x_1 + x_2 + x_3 &= 1 \\ \begin{pmatrix} 2x_1 - \lambda_1 \\ 2x_2 - \lambda_1 \\ 2x_3 - \lambda_1 \end{pmatrix} &= \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \end{aligned}$$

Ce qui est donc le système d'équations linéaires suivant :

$$\begin{pmatrix} 1 & 1 & 1 & 0 \\ 2 & 0 & 0 & -1 \\ 0 & 2 & 0 & -1 \\ 0 & 0 & 2 & -1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \lambda_1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

La résolution de ce système, avec une méthode quelconque, donne :

$$\mathbf{x}^* = \begin{pmatrix} 1/3 \\ 1/3 \\ 1/3 \end{pmatrix} \quad \lambda_1^* = 2/3$$

Remarque. Rappelons-nous que les calculs effectués ci-dessus prouvent seulement que le point $\mathbf{x}^*$ est un point stationnaire. Pour prouver qu'il est vraiment un minimum, il faudrait aussi vérifier la condition basée sur le Hessien de f.

14.1.2 Fonction de coût non-linéaire

Beaucoup de méthodes d'optimisation non contrainte peuvent être adaptées au cas présent, par exemple les méthodes du gradient, du gradient conjugué et de Newton.

Dans la suite, nous considérons la méthode de Newton. Il faut donc construire une approximation quadratique de la fonction de coût, qui soit restreinte sur U. Etant donné un point

$\mathbf{x} \in U$, nous savons que $\mathbf{x} + \mathsf{Z}\mathbf{s}$ est dans U, pour tout vecteur $\mathbf{s}$ (cf. équation (**??**)). Nous formulons alors l'approximation quadratique :

$$f(\mathbf{x} + \mathsf{Z}\mathbf{s}) = f(\mathbf{x}) + \mathbf{s}^\mathsf{T}\mathsf{Z}^\mathsf{T}\mathbf{g}(\mathbf{x}) + \frac{1}{2}\mathbf{s}^\mathsf{T}\mathsf{Z}^\mathsf{T}\mathsf{H}(\mathbf{x})\mathsf{Z}\mathbf{s} + \epsilon$$

Afin de trouver $\mathbf{s}$ qui minimise cette fonction, nous calculons les dérivées par rapport à $\mathbf{s}$ et imposons :

$$\mathsf{Z}^\mathsf{T}\mathbf{g}(\mathbf{x}) + \mathsf{Z}^\mathsf{T}\mathsf{H}(\mathbf{x})\mathsf{Z}\mathbf{s} = \mathbf{0}$$

La solution est alors donnée par (si l'on suppose que le Hessien projeté est inversible) :

$$\mathbf{s}^* = -\left(\mathsf{Z}^\mathsf{T}\mathsf{H}(\mathbf{x})\mathsf{Z}\right)^{-1}\mathsf{Z}^\mathsf{T}\mathbf{g}(\mathbf{x})$$

14.2 Méthodes pour contraintes d'inégalité

14.2.1 Méthodes de type ensemble actif

Dans la suite, nous esquissons quelques principes d'une famille de méthodes appelées *méthodes d'ensemble actif* (active set methods). Seulement des contraintes d'inégalité seront considérées ; l'inclusion de contraintes d'égalité est relativement directe.

Rappelons-nous que parmi les conditions d'optimalité se trouve :

$$\exists \lambda_1^* \geq 0, \ldots, \lambda_m^* \geq 0 \text{ avec } \quad \mathbf{g}(\mathbf{x}^*) = \sum_{i=1}^{m} \hat{\mathbf{a}}_i \lambda_i^* = \hat{\mathsf{A}}^\mathsf{T}\lambda^*$$

où $\hat{\mathsf{A}}$ et $\hat{\mathbf{b}}$ représentent les contraintes *actives* au point $\mathbf{x}^*$. Si l'on connaissait quelles sont les contraintes actives au minimum recherché, on pourrait le trouver en appliquant des méthodes pour des contraintes d'égalité (plus considération du signe des multiplicateurs de Lagrange). Pourtant, les contraintes actives à la solution ne sont justement pas connues, donc le problème est plus compliqué.

Les méthodes de type ensemble actif procèdent de manière itérative. Elles gèrent non seulement les estimées de la solution $\mathbf{x}^*$, mais aussi un *ensemble de travail* – l'ensemble des contraintes *supposées* actives à la solution. L'ensemble de travail contient typiquement seulement des contraintes qui sont actives au point actuel.

Typiquement, les méthodes procèdent de la manière suivante. Soit l'ensemble de travail l'ensemble des contraintes actives au point actuel. La direction de recherche est calculée avec une des méthodes pour contraintes d'égalité (e.g. la méthode de Newton modifiée, expliquée ci-dessus). Une longueur de pas qui ne viole aucune contrainte inactive est calculée (il s'agit donc d'un problème d'optimisation contrainte sur une variable). Des contraintes d'inégalité qui deviennent actives au nouveau point, sont inclues dans l'ensemble de travail. Puisque au minimum recherché, il se peut qu'aucune contrainte ne soit active, les méthodes doivent être capables d'effacer des contraintes de l'ensemble de travail, afin de pouvoir quitter le sous-espace des points défini par une contrainte d'égalité. Il y a une multitude de stratégies pour ce faire ...

14.2.2 Cas particuliers de fonctions de coût

Si la fonction de coût est linéaire respectivement quadratique, on parle de problèmes de *programmation linéaire* respectivement de *programmation quadratique*.

L'une des méthodes les plus populaires pour la programmation linéaire est la *méthode du simplex*.