

Algorithmes de quasi-Newton

The key ideas that led me to the development of variable-metric algorithms were 1) to update a metric in the space of gradients during the search for an optimum, rather than waiting until the search was over, and 2) to accelerate convergence by using each updated metric to choose the next search direction.

W.C. DAVIDON, dans la nouvelle introduction de son article fondateur des méthodes de quasi-Newton, écrit en 1959, mais qui ne fut publié qu'en 1991, lors de la parution du premier numéro de la revue *SIAM Journal on Optimization* [155, 156].

I was able to minimize some nonquadratic functions of 100 variables by Davidon's method in 1962. I mentioned this fact at a meeting that was held at Imperial College then, but this remark was impromptu, it was made in a discussion that included the view from a senior person that 10-variable problems were usually too difficult and I was nervous.

M.J.D. POWELL [495; 2003].

Dans ce chapitre, on s'intéresse à la résolution du problème d'optimisation sans contrainte

$$\begin{cases} \min f(x) \\ x \in \mathbb{R}^n \end{cases}$$

par des algorithmes à directions de descente particuliers. On note $\{x_k\}_{k \geq 1}$ la suite des itérés et $g_k = \nabla f(x_k)$ le gradient de f en x_k . Les itérés sont donc générés par la récurrence

$$x_{k+1} = x_k + \alpha_k d_k, \quad k \geq 1,$$

où le pas $\alpha_k > 0$ est déterminé par recherche linéaire (section 6.3) et d_k est une direction de descente. Dans les méthodes de quasi-Newton, d_k est de la forme

$$d_k = -M_k^{-1} g_k. \tag{11.1}$$

Si $M_k = \nabla^2 f(x_k)$, on retrouve la méthode de Newton, si bien que le nom donné aux méthodes décrites ci-dessous apparaît naturel. Cependant, tout algorithme ayant une direction de descente de la forme (11.1) ne se retrouve pas pour autant dans cette classe de méthodes. Le qualificatif *quasi-Newton* fait en effet référence à un ensemble

de techniques mises au point à partir des années 1960 permettant de générer les matrices M_k à partir des gradients g_k .

Dans beaucoup de problèmes, on cherche à éviter le calcul des dérivées secondes, pour les raisons suivantes :

- l'évaluation de $\nabla^2 f(x_k)$ ou le produit de cette hessienne par un vecteur comme dans l'algorithme de Newton tronqué (section 10.3.1) peut demander trop de temps de calcul,
- on ne dispose pas toujours des dérivées secondes et leur calcul peut demander un investissement humain trop important, alors que l'on voudrait obtenir un résultat rapidement par des algorithmes peut-être plus lents que l'algorithme de Newton mais ne demandant que le calcul des dérivées premières,
- la fonction peut ne pas être deux fois dérivable,
- on ne dispose pas de place mémoire pour stocker les $O(n^2)$ éléments d'une matrice (c'est la moins bonne des raisons, voir section 10.3.1).

Ce sont des situations dans lesquelles les algorithmes de quasi-Newton sont très utiles. Dans ceux-ci, la matrice M_k dans (11.1) n'est pas égale à $\nabla^2 f(x_k)$, mais est générée par des formules qui cherchent à ce que cette matrice soit proche de la hessienne de f . On parle de *formules de mise à jour*. Celles-ci ne font intervenir que les dérivées premières de f . C'est en utilisant la variation du gradient de f d'une itération à l'autre que ces formules permettent d'engranger de l'information sur la hessienne.

Dans ce chapitre, on se propose d'introduire et d'étudier ces formules de mise à jour et les algorithmes de quasi-Newton qui les utilisent. Voici quelques propriétés de ces algorithmes.

- Localement (proche d'une solution), les algorithmes de quasi-Newton convergent moins rapidement que l'algorithme de Newton. Dans les implémentations correctes, les itérés convergent toutefois q-superlinéairement.
- Chaque itération demande moins de calcul au simulateur que dans l'algorithme de Newton : il ne faut pas évaluer les dérivées secondes.
- Dans leur version standard, ces algorithmes peuvent être utilisés pour un nombre de variables qui n'est pas trop grand, disons $n \leq 500$ pour fixer les idées. Cette borne sur n vient d'une part du coup de l'itération (de l'ordre de $O(n^2)$ opérations dans l'optimiseur) et du fait que les algorithmes deviennent plus lents lorsque n augmente. Notons qu'il existe des adaptations de ces algorithmes quasi-newtoniens pouvant être utilisées pour résoudre des problèmes de très grande taille, avec plusieurs millions de variables, par exemple. Ces méthodes sont dites à *mémoire limitée* car elles ne demandent pas que l'on garde M_k en mémoire. L'algorithme ℓ -BFGS de la section 11.2.5 est de ce type. Il est très utilisé.

Hypothèse générale à ce chapitre. Nous développerons la théorie lorsque le produit scalaire que l'on se donne sur \mathbb{R}^n est le produit scalaire euclidien : $(u, v) \mapsto u^\top v$. Dans ce cas, $\nabla f(x)$ est le vecteur des dérivées partielles de f . On peut se placer dans un cadre plus général en ne faisant aucune hypothèse sur le produit scalaire $(u, v) \mapsto \langle u, v \rangle$ utilisé. Il faut alors utiliser le produit tensoriel associé $(u, v) \mapsto u \otimes v$, où $(u \otimes v)$ est la matrice d'ordre n définie par

$$(u \otimes v)d := \langle v, d \rangle u, \quad \text{pour tout } d \in \mathbb{R}^n.$$

Pour obtenir des formules de mise à jour tenant compte du produit scalaire utilisé (et donc du préconditionnement que cela implique), il suffit le plus souvent de remplacer dans les formules obtenues une matrice de la forme uv^T par la matrice $u \otimes v$. Voir par exemple [244] et ses références pour plus détails.

11.1 Système d'équations

11.1.1 Formules de mise à jour

11.1.2 Convergence linéaire locale

Lemme 11.1 (détérioration bornée)

DÉMONSTRATION.

□

11.2 Optimisation

11.2.1 Formules de mise à jour

Principes

Supposons que M_k soit connue et cherchons quelle valeur donner à M_{k+1} pour que cette matrice approche $\nabla^2 f(x_{k+1})$. On cherche à ce que M_{k+1} soit proche de la hessienne de f pour que l'algorithme hérite des bonnes propriétés de convergence locale de l'algorithme de Newton.

Soient

$$s_k := x_{k+1} - x_k \quad \text{et} \quad y_k := g_{k+1} - g_k,$$

où $g_k = \nabla f(x_k)$ comme d'habitude. Le développement de Taylor avec reste intégral s'écrit

$$y_k = \left(\int_0^1 \nabla^2 f(x_k + ts_k) dt \right) s_k.$$

Si on veut que la nouvelle matrice M_{k+1} approche la hessienne de f , il semble raisonnable de lui imposer de satisfaire l'équation vérifiée par la hessienne *moyenne* $\int_0^1 \nabla^2 f(x_k + ts_k) dt$, à savoir

$$y_k = M_{k+1} s_k. \tag{11.2}$$

Cette relation porte le nom d'*équation de quasi-Newton*. D'autre part, comme la hessienne de f est symétrique, il est normal d'imposer également cette propriété à M_{k+1} :

$$M_{k+1} = M_{k+1}^T. \tag{11.3}$$

Rien n'assure a priori que cette idée simple d'imposer à M_{k+1} de vérifier des propriétés facilement exprimable de la hessienne moyenne donnera une matrice avec de bonnes propriétés, mais l'expérience a montré que l'inspiration est excellente.

Si $n > 1$, les conditions (11.2) et (11.3) ne suffisent pas pour déterminer M_{k+1} : il y a $n(n+1)/2$ inconnues et n équations seulement. Ce manque de conditions laisse une liberté dont ne se sont pas privé les spécialistes en optimisation numérique. Les années 1960-80 ont vu ainsi naître de nombreuses formules de mise à jour. Les plus importantes en optimisation sont la formule SR1 et la formule de BFGS.

Avant d'introduire ces formules, notons que lorsque $n = 1$, les conditions (11.2) et (11.3) déterminent M_{k+1} , qui est le scalaire y_k/s_k (on suppose $x_{k+1} \neq x_k$). L'algorithme de quasi-Newton avec pas unité associé donne (en décalant les indices d'une unité et en supposant $g_{k-1} \neq g_k$ et $g_k \neq 0$) :

$$x_{k+1} = x_k - \frac{s_{k-1}}{y_{k-1}} g_k \quad \text{ou} \quad \frac{x_k - x_{k+1}}{g_k - 0} = \frac{x_{k-1} - x_k}{g_{k-1} - g_k}.$$

On reconnaît l'*algorithme de la sécante*. Les méthodes de quasi-Newton portent d'ailleurs parfois ce nom. La figure 11.1 rappelle ce qu'est l'algorithme de la sécante en

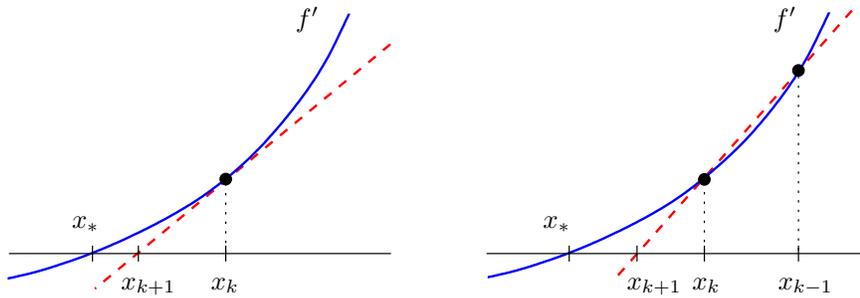


Fig. 11.1. Comparaison des algorithmes de Newton (à gauche) et de quasi-Newton (à droite) lorsque $n = 1$

dimension 1 et le compare à l'algorithme de Newton lorsqu'on cherche à minimiser une fonction $x \in \mathbb{R} \mapsto f(x) \in \mathbb{R}$ en annulant sa dérivée. Dans l'algorithme de Newton (à gauche), le nouvel itéré x_{k+1} est obtenu en calculant l'intersection avec l'axe des abscisses de la droite tangente à la courbe $x \mapsto f'(x)$ en $(x_k, f'(x_k))$ (c'est le zéro de la fonction f' linéarisée en x_k), tandis que dans l'algorithme de la sécante (à droite), on prend l'intersection du même axe avec la droite passant par $(x_{k-1}, f'(x_{k-1}))$ et $(x_k, f'(x_k))$. Dans ce dernier algorithme, on ne doit pas linéariser f' (c'est-à-dire calculer la dérivée seconde de f), mais on doit utiliser la valeur de f' aux deux points x_{k-1} et x_k .

Formule SR1

Une première idée est de rechercher parmi les matrices vérifiant (11.2) et (11.3), une matrice M_{k+1} suffisamment proche de M_k en imposant que la différence entre les deux matrices soit de rang 1. On prend une correction de faible rang de manière à assurer une certaine stabilité à la suite $\{M_k\}$. Il semble en effet raisonnable de souhaiter que ces matrices n'oscillent pas trop au cours des itérations.

On cherche donc une mise à jour de la forme

$$M_{k+1} = M_k + uv^\top,$$

où u et v sont deux vecteurs de \mathbb{R}^n à déterminer. Si on veut que M_{k+1} vérifie l'équation de quasi-Newton (11.2), il faut que u et v satisfassent

$$y_k = M_k s_k + (v^\top s_k)u.$$

Si $v^\top s_k = 0$, soit M_k vérifie déjà l'équation de quasi-Newton, soit une correction de rang 1 de M_k ne permet pas d'obtenir une matrice vérifiant cette équation. Si $v^\top s_k \neq 0$, la relation ci-dessus permet de déterminer u , ce qui conduit à

$$M_{k+1} = M_k + \frac{(y_k - M_k s_k)v^\top}{v^\top s_k}.$$

Pour que M_{k+1} soit symétrique (on suppose que M_k l'est), il faut donc prendre $v = y_k - M_k s_k$, ce qui donne

$$\boxed{M_{k+1} = M_k + \frac{(y_k - M_k s_k)(y_k - M_k s_k)^\top}{(y_k - M_k s_k)^\top s_k}}. \quad (11.4)$$

Cette formule porte le nom de *formule SR1* (Symétrique de Rang 1). Elle n'est bien définie que si $(y_k - M_k s_k)^\top s_k \neq 0$. En pratique, cela nécessite l'utilisation de garde-fous corrigeant le dénominateur dans (11.4) s'il est trop petit. La formule SR1 est souvent utilisée lorsqu'il n'est pas possible ou qu'il n'est pas nécessaire que M_{k+1} soit définie positive. Si cette propriété est souhaitable, on recourt à la formule de BFGS décrite ci-après, laquelle a de meilleures propriétés que la formule SR1.

Formule de BFGS

Une autre manière d'imposer à M_{k+1} d'être proche de M_k est de minimiser l'«écart» entre M_{k+1} et M_k , toujours en requérant que M_{k+1} soit symétrique et vérifie l'équation de quasi-Newton (11.2). On est donc conduit à considérer le problème en la variable matricielle $M \in \mathbb{R}^{n \times n}$ suivant :

$$\begin{cases} \min \text{ «écart»}(M, M_k) \\ y_k = M s_k \\ M = M^\top. \end{cases} \quad (11.5)$$

On dit alors que la matrice est obtenue par une *approche variationnelle*. La fonction «écart» utilisée dans ce problème est spécifiée ci-dessous.

Il est souvent intéressant d'imposer également la définie positivité des matrices M_k . En effet, dans ce cas, d_k donnée par (11.1) est une direction de descente de f en x_k . Cette exigence n'est pas dépourvue de fondement puisque M_k doit approcher $\nabla^2 f(x_*)$ qui est *semi-définie positive* en la solution et définie positive en des solutions fortes (celles vérifiant les conditions d'optimalité du second ordre).

Pour obtenir des matrices ayant cette propriété de définie positivité, il ne suffit pas d'ajouter cette contrainte au problème (11.5). En effet, le cône \mathcal{S}_{++}^n des matrices symétriques définies positives est un ouvert dans l'espace vectoriel des matrices $\mathbb{R}^{n \times n}$, si bien qu'avec cette contrainte additionnelle, (11.5) peut ne pas avoir de solution.

Imposer la **semi-définie positivité** (on a alors un fermé) n'est pas satisfaisant non plus, car M_{k+1} ne serait pas nécessairement inversible et d_k ne serait pas bien défini par (11.1). On va donc chercher à ce que ce soit le critère dans (11.5) qui impose la définie positivité de la matrice solution.

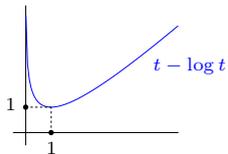
Dans ce but, on commence par introduire une fonction $\psi : \mathcal{S}^n \rightarrow \mathbb{R} \cup \{+\infty\}$, dont le **domaine** est \mathcal{S}_{++}^n et qui forme une « barrière » au bord du cône \mathcal{S}_{++}^n (elle tend vers l'infini lorsque son argument se rapproche du bord de \mathcal{S}_{++}^n) ainsi qu'à l'infini :

$$\psi(M) = \text{tr } M + \text{ld } M, \tag{11.6}$$

où la fonction log-déterminant $\text{ld} : \mathcal{S}^n \rightarrow \mathbb{R} \cup \{+\infty\}$ est définie en $M \in \mathcal{S}^n$ par

$$\text{ld}(M) = \begin{cases} -\log \det M & \text{si } M \in \mathcal{S}_{++}^n \\ +\infty & \text{sinon} \end{cases}$$

Les propriétés annoncées de ψ peuvent se voir sur son expression suivante. Si on note $\{\lambda_i\}$ les valeurs propres de M , on a $\text{tr } M = \sum_{i=1}^n \lambda_i$, $\det M = \prod_{i=1}^n \lambda_i$ et donc

$$\psi(M) = \sum_{i=1}^n (\lambda_i - \log \lambda_i), \quad \text{si } M \in \mathcal{S}_{++}^n. \tag{11.7}$$


Étant donné l'allure de la fonction $t \in \mathbb{R}_{++} \mapsto t - \log t$, $\psi(M)$ tend vers l'infini si l'une des valeurs propres de M tend vers zéro ou vers l'infini.

La formule (11.7) montre aussi que l'unique minimiseur de ψ est la matrice identité ($\lambda_i = 1$ pour tout i). Afin de minimiser l'écart entre M et M_k , on va chercher à ce que $M_k^{-1/2} M M_k^{-1/2}$ soit proche de I . Ceci peut être obtenu en minimisant $\psi(M_k^{-1/2} M M_k^{-1/2})$. On est donc conduit à résoudre le problème suivant sur l'espace vectoriel \mathcal{S}^n des matrices symétriques :

$$\begin{cases} \min \psi(M_k^{-1/2} M M_k^{-1/2}) \\ y_k = M s_k \\ M \in \mathcal{S}_{++}^n \quad (\text{contrainte implicite}). \end{cases} \tag{11.8}$$

Si $s_k = 0$, de deux choses l'une : soit $y_k \neq 0$ et le problème ci-dessus n'a pas de solution (son ensemble admissible est vide), soit $y_k = 0$ et sa solution est M_k (ψ prend alors sa valeur minimale n sur \mathcal{S}_{++}^n). Le cas non trivial où $s_k \neq 0$ est examiné dans la proposition suivante.

Proposition 11.2 *Supposons que M_k soit symétrique définie positive et que $s_k \neq 0$. Alors, le problème (11.8) a une solution si, et seulement si, $y_k^\top s_k > 0$. Sous cette condition la solution M_{k+1} de (11.8) est unique et est donnée par l'une des formules suivantes :*

$$M_{k+1} = M_k + \frac{y_k y_k^\top}{y_k^\top s_k} - \frac{M_k s_k s_k^\top M_k}{s_k^\top M_k s_k}, \quad (11.9)$$

$$W_{k+1} = \left(I - \frac{s_k y_k^\top}{y_k^\top s_k} \right) W_k \left(I - \frac{y_k s_k^\top}{y_k^\top s_k} \right) + \frac{s_k s_k^\top}{y_k^\top s_k}, \quad (11.10)$$

où on a noté $W_k := M_k^{-1}$ et $W_{k+1} := M_{k+1}^{-1}$.

DÉMONSTRATION. D'abord, il est clair que si (11.8) a une solution M_{k+1} , qui est une matrice symétrique définie positive, on a $y_k^\top s_k = s_k^\top M_{k+1} s_k > 0$. On suppose dorénavant que $y_k^\top s_k > 0$.

Montrons d'abord que le problème (11.8) a au plus une solution. L'ensemble admissible de (11.8) étant clairement convexe, il suffit de vérifier que ψ est strictement convexe, ce que l'on fait en calculant ses dérivées secondes. Rappelons que, pour une matrice inversible A , $\det'(A) \cdot H = (\det A) \operatorname{tr}(A^{-1}H)$. Alors, pour tout $M \in \mathcal{S}_{++}^n$ et tout $H \in \mathcal{S}^n$ non nul, on a avec $W := M^{-1}$:

$$\psi'(M) \cdot H = \operatorname{tr} H - \operatorname{tr}(WH),$$

$$\psi''(M) \cdot H^2 = \operatorname{tr}(W^{1/2} H W^{1/2})(W^{1/2} H W^{1/2}) > 0.$$

On achève la démonstration en calculant explicitement la solution $M = M_{k+1}$ de (11.8) au moyen des conditions d'optimalité du premier ordre. On travaille dans l'espace vectoriel \mathcal{S}^n des matrices symétriques d'ordre n , muni du produit scalaire $\langle A, B \rangle = \operatorname{tr} AB$. On note $\lambda \in \mathbb{R}^n$ le multiplicateur de Lagrange associé à la contrainte affine de (11.8), si bien que le lagrangien s'écrit

$$\ell(M, \lambda) = \psi(W_k^{1/2} M W_k^{1/2}) + \lambda^\top (y_k - M s_k).$$

On cherche $M \in \mathcal{S}_{++}^n$ telle que pour tout $H \in \mathcal{S}^n$ la dérivée directionnelle de ℓ en M dans la direction H soit nulle, c'est-à-dire

$$\operatorname{tr}(W_k^{1/2} H W_k^{1/2}) - \operatorname{tr}(M_k^{1/2} W M_k^{1/2})(W_k^{1/2} H W_k^{1/2}) - \lambda^\top H s_k = 0.$$

En utilisant la relation $\operatorname{tr}(AB) = \operatorname{tr}(BA)$, on obtient pour tout $H \in \mathcal{S}^n$:

$$0 = \operatorname{tr}(W_k H) - \operatorname{tr}(W H) - \operatorname{tr}(s_k \lambda^\top H) = \operatorname{tr} \left(\left[W_k - W - \frac{s_k \lambda^\top + \lambda s_k^\top}{2} \right] H \right).$$

Comme le facteur de H ci-dessus est dans \mathcal{S}^n , on a

$$W = W_k - \frac{s_k \lambda^\top + \lambda s_k^\top}{2}.$$

Pour calculer λ , on utilise l'équation de quasi-Newton, qui donne

$$s_k = W_k y_k - \frac{y_k^\top \lambda}{2} s_k - \frac{y_k^\top s_k}{2} \lambda.$$

En prenant le produit scalaire avec y_k et en utilisant le fait que $y_k^\top s_k \neq 0$, on trouve $y_k^\top \lambda = y_k^\top (W_k y_k - s_k) / (y_k^\top s_k)$ qui, injecté dans la relation ci-dessus, fournit la valeur du multiplicateur

$$\lambda = \frac{-2}{y_k^\top s_k} (s_k - W_k y_k) + \frac{y_k^\top (s_k - W_k y_k)}{(y_k^\top s_k)^2} s_k.$$

En utilisant celle-ci dans la formule de W ci-dessus, on trouve l'expression de W_{k+1} suivante

$$W_{k+1} = W_k + \frac{(s_k - W_k y_k) s_k^\top + s_k (s_k - W_k y_k)^\top}{y_k^\top s_k} - \frac{y_k^\top (s_k - W_k y_k)}{(y_k^\top s_k)^2} s_k s_k^\top. \quad (11.11)$$

Par un calcul laborieux, mais mécanique, on vérifie que cette formule est équivalente à (11.10) et que M_{k+1} donné par (11.9) est bien l'inverse de W_{k+1} donné par (11.10) (il suffit de vérifier que $M_{k+1} W_{k+1} = I$).

Il reste à montrer que la matrice obtenue est bien dans \mathcal{S}_{++}^n (le problème (11.8) a des points stationnaires en dehors de cet ensemble), si $y_k^\top s_k > 0$. Pour un vecteur $v \in \mathbb{R}^n$ arbitraire, la formule (11.9) donne

$$v^\top M_{k+1} v = v^\top M_k v - \frac{(s_k^\top M_k v)^2}{s_k^\top M_k s_k} + \frac{(y_k^\top v)^2}{y_k^\top s_k}.$$

Le dernier terme est positif. On voit qu'il en est de même de la somme des deux premiers termes, car d'après l'inégalité de Cauchy-Schwarz

$$(s_k^\top M_k v)^2 \leq \|M_k^{1/2} s_k\|_2^2 \|M_k^{1/2} v\|_2^2 = (s_k^\top M_k s_k) (v^\top M_k v).$$

Donc $v^\top M_{k+1} v \geq 0$ et ce produit ne peut être nul que si $v \perp y_k$ et $v \parallel s_k$, ce qui n'est vrai que si $v = 0$, car $y_k^\top s_k > 0$. \square

Les formules (11.9) et (11.10) portent le nom de *formules de BFGS* directe et inverse. L'appellation BFGS vient des initiales des auteurs (Broyden, Fletcher, Goldfarb et Shanno) qui ont proposé cette formule dans des articles parus en 1970 [96, 209, 257, 545]. On notera les formules (11.9) et (11.10) de la manière suivante :

$$M_{k+1} = \text{BFGS}(M_k, y_k, s_k) \quad \text{et} \quad W_{k+1} = \overline{\text{BFGS}}(W_k, y_k, s_k).$$

Notons que la première de ces formules peut aussi s'écrire sous la forme de produits de matrices [92, 258] :

$$M_{k+1} = (I + v_k \bar{s}_k^\top) M_k (I + \bar{s}_k v_k^\top),$$

où $v_k := -M \bar{s}_k + \bar{y}_k$, $\bar{s}_k := s_k / (s_k^\top M_k s_k)^{1/2}$ et $\bar{y}_k := y_k / (y_k^\top s_k)^{1/2}$. Observons enfin que les corrections $M_{k+1} - M_k$ et $W_{k+1} - W_k$ apportées par la formule de BFGS sont des matrices de rang 2.

La proposition 11.2 a pour corollaire immédiat que si $s \neq 0$

$$\exists M \in \mathcal{S}_{++}^n : y = Ms \iff y^\top s > 0. \quad (11.12)$$

Cette proposition a une extension naturelle au cas où l'on impose plus d'une équation de quasi-Newton, qui doivent toutefois être compatibles : $M \in \mathcal{S}_{++}^n$ doit vérifier $Y = MS$, où $Y, S \in \mathbb{R}^{n \times p}$ et S est *injective* (voir l'exercice 11.1). L'équivalence (11.12) devient alors

$$\exists M \in \mathcal{S}_{++}^n : Y = MS \iff Y^\top S \in \mathcal{S}_{++}^n.$$

On utilise rarement une formule de mise à jour qui génère une matrice M vérifiant le système $Y = MS$ rassemblant plus d'une équation de quasi-Newton, car on ne sait pas comment satisfaire la condition $Y^\top S \succ 0$ qui assure la définie positivité de M . La situation est différente si l'on n'a qu'une seule équation de quasi-Newton à satisfaire, car nous verrons qu'il est aisé de vérifier la condition $y^\top s > 0$ apparaissant dans (11.12) (voir la section 11.2.2).

Propriétés algébriques

Voici une propriété de la formule de BFGS qui justifie l'utilisation de cette formule pour construire des préconditionneurs symétriques définis positifs de systèmes linéaires. Elle nous apprend que, si au cours de la minimisation d'une fonction quadratique strictement convexe sur \mathbb{R}^n par l'algorithme du gradient conjugué, on met à jour une matrice par le formule de BFGS (resp. par la formule de BFGS inverse), en utilisant les couples (Au_i, u_i) formés à partir des directions conjuguées u_i , la matrice obtenue après n itérations est la hessienne de la fonction minimisée (resp. son inverse).

Proposition 11.3 Soit A une matrice symétrique et u_1, \dots, u_p des directions non nulles, conjuguées par rapport à cette matrice (c'est-à-dire : $u_i^\top Au_j = 0$, pour $i \neq j$ et $u_i^\top Au_i > 0$). On se donne une matrice M_1 symétrique définie positive et on définit $M_{i+1} = \text{BFGS}(M_i, Au_i, u_i)$, pour $i = 1, \dots, p$. Alors,

- (i) $M_{k+1}u_i = Au_i$, pour $k = 1, \dots, p$ et $i = 1, \dots, k$,
- (ii) si $p = n$, on a $M_{n+1} = A$.

DÉMONSTRATION. La première propriété se démontre par récurrence. Elle est vérifiée pour $k = 1$: $M_2u_1 = Au_1$ (c'est l'équation de quasi-Newton vérifiée par M_2). Supposons qu'elle le soit pour un $k = 1, \dots, l-1$, avec $2 \leq l \leq p$, et démontrons la pour $k = l$. Soit $1 \leq i < l$. Par conjugaison et récurrence, $u_i^\top Au_i = 0$ et $u_i^\top M_l u_i = u_i^\top Au_i = 0$. Dès lors, la formule de BFGS (11.9) donne $M_{l+1}u_i = M_l u_i = Au_i$. Si $i = l$, on a $M_{l+1}u_i = Au_i$ (c'est l'équation de quasi-Newton vérifiée par M_{l+1}).

Si $p = n$, M_{n+1} prend la même valeur que A sur les n vecteurs linéairement indépendants u_1, \dots, u_n . Donc $M_{n+1} = A$. \square

Terminons cette section par deux formules qui nous seront utiles pour étudier la convergence de l'algorithme de BFGS, celles de la [trace](#) et du déterminant.

Proposition 11.4 Si M_{k+1} et M_k sont reliés par le formule de BFGS (11.9) avec M_k définie positive et $y_k^\top s_k > 0$, on a

$$\operatorname{tr} M_{k+1} = \operatorname{tr} M_k + \frac{\|y_k\|_2^2}{y_k^\top s_k} - \frac{\|M_k s_k\|_2^2}{s_k^\top M_k s_k} \quad \text{et} \quad \det M_{k+1} = \det M_k \left(\frac{y_k^\top s_k}{s_k^\top M_k s_k} \right).$$

DÉMONSTRATION. La formule de la *trace* se déduit directement de (11.9). Pour celle du déterminant, on écrit

$$\det M_{k+1} = \det(M_k) \det \left(I + \frac{M_k^{-1} y_k y_k^\top}{y_k^\top s_k} - \frac{s_k s_k^\top M_k}{s_k^\top M_k s_k} \right).$$

Pour évaluer le dernier déterminant, on utilise le point (iii) de l'exercice B.16. \square

Propriétés asymptotiques \blacktriangle

Supposons que l'on ait deux suites $\{y_k\}_{k \geq 1}$ et $\{s_k\}_{k \geq 1}$ de vecteurs de \mathbb{R}^n , vérifiant $y_k^\top s_k > 0$ pour tout $k \geq 1$, et une matrice d'ordre n symétrique définie positive M_1 . On génère alors la suite $\{M_k\}_{k \geq 1}$ par la formule de BFGS : $M_{k+1} = \text{BFGS}(M_k, y_k, s_k)$ pour tout $k \geq 1$. On s'intéresse ici aux propriétés que peuvent induire les vecteurs y_k et s_k sur la suite $\{M_k\}$. Ces propriétés seront utiles pour étudier le comportement asymptotique des suites générées par l'algorithme de BFGS.

On sait déjà, d'après la proposition 11.2, que les matrices M_k sont toutes symétriques définies positives.

Formule de mise à jour pour matrices creuses \blacktriangle

11.2.2 L'algorithme de BFGS

D'après la proposition 11.2, on aura un algorithme à directions de descente si dans (11.1), M_k est générée par la formule de BFGS à partir d'une matrice initiale M_1 définie positive et si à chaque itération on réalise l'inégalité $y_k^\top s_k > 0$. La formule de BFGS génère alors des matrices M_k définies positives. Il est tout à fait remarquable que l'inégalité $y_k^\top s_k > 0$ soit satisfaite lorsque la recherche linéaire détermine le pas $\alpha_k > 0$ par la règle de Wolfe (section 6.3.4; rappelons que l'on a choisit le produit scalaire euclidien $\langle u, v \rangle := u^\top v$). En effet, si l'on retranche $g_k^\top d_k$ dans les deux membres de l'inégalité (6.11b), on obtient

$$y_k^\top s_k \geq (\omega_2 - 1) g_k^\top s_k > 0,$$

car $\omega_2 < 1$ et $g_k^\top s_k = \alpha_k g_k^\top d_k < 0$ (d_k est une direction de descente). Pour minimiser une fonction non linéaire (non quadratique) au moyen de la formule de BFGS, on utilise donc *toujours* la recherche linéaire de Wolfe. Ceci conduit à l'algorithme parfaitement bien défini suivant.

Algorithme 11.5 (de BFGS)

-
0. On se donne deux constantes ω_1 et ω_2 pour la recherche linéaire de Wolfe :
 $0 < \omega_1 < \frac{1}{2}$ et $\omega_1 < \omega_2 < 1$.
 Choix d'un itéré initial $x_1 \in \mathbb{R}^n$ et d'une matrice initiale W_1 définie positive
 (approximation de l'inverse de la hessienne $\nabla^2 f(x_1)$).
 Initialisation : $k := 1$.
 1. *Test d'arrêt* : si $\nabla f(x_k) = 0$, arrêt de l'algorithme.
 2. *Calcul de la direction de descente* : $d_k = -W_k g_k$.
 3. *Recherche linéaire de Wolfe* : trouver un pas $\alpha_k > 0$ tel que l'on ait (6.11a)
 et (6.11b).
 4. $x_{k+1} := x_k + \alpha_k d_k$.
 5. Mettre à jour la matrice W_k par la formule (11.10) dans laquelle on prend
 $s_k = x_{k+1} - x_k$ et $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$.
 6. Accroître k de 1 et aller en 1.
-

En pratique, on arrête l'algorithme à l'étape 1 lorsque $\nabla f(x_k)$ est suffisamment petit. On notera $M_k = W_k^{-1}$ l'inverse des matrices générées par l'algorithme ci-dessus, qui approchent donc la hessienne $\nabla^2 f$. On trouvera à la section 11.2.4 plus de détails sur la mise en œuvre de l'algorithme de BFGS. Analysons d'abord ses principales propriétés.

11.2.3 Propriétés de l'algorithme de BFGS*Convergence globale*

My favorite results are his proof of the global convergence of the BFGS method for convex functions, and the introduction of the least change approach for derivative-free optimization.

J. MOREÉ (2015), dans un hommage à M.J.D. Powell [439].

Un des plus beaux résultats de convergence en optimisation sans contrainte est le théorème 11.6 ci-dessous. Il donne des conditions assurant la convergence globale de l'algorithme de BFGS lorsque la fonction à minimiser est convexe. On notera qu'il n'est pas nécessaire de supposer la forte convexité du critère. La beauté du résultat tient en particulier au fait qu'il est rare de pouvoir démontrer la convergence d'un algorithme quasi-newtonien, avec si peu d'hypothèses sur les objets générés par celui-ci. On suppose seulement que l'algorithme génère une suite $\{x_k\}$ (ce qui signifie qu'il ne s'arrête pas à l'étape 1 parce que le gradient y est nul ; auquel cas, il n'y a rien à démontrer) et que la suite $f(x_k)$ est bornée inférieurement (il existe une constante C telle que pour tout indice k , $f(x_k) \geq C$). On ne fait aucune hypothèse sur la suite des matrices $\{M_k\}$.

Théorème 11.6 (convergence globale de BFGS) *Supposons que $f : \mathbb{R}^n \rightarrow \mathbb{R}$ soit convexe et $\mathcal{C}^{1,1}$ dans un voisinage convexe de $\{x \in \mathbb{R}^n : f(x) \leq f(x_1)\}$, où $x_1 \in \mathbb{R}^n$. On considère l'algorithme de BFGS démarrant en x_1 avec une matrice M_1 symétrique définie positive et on suppose que celui-ci génère une suite $\{x_k\}$ telle que $\{f(x_k)\}$ soit bornée inférieurement. Alors, $\liminf_{k \rightarrow \infty} \|g_k\| = 0$.*

DÉMONSTRATION. On note C_1, C_2, \dots des constantes strictement positives. L'idée de la démonstration est de contrôler le comportement des matrices $\{M_k\}$ en obtenant une majoration de leur [trace](#) et une minoration de leur déterminant, ce qui permettra d'avoir une borne inférieure sur le pas α_k . On note

$$q_k := \frac{s_k^\top M_k s_k}{\|s_k\|_2^2} \quad \text{et} \quad \cos \theta_k := \frac{-g_k^\top d_k}{\|g_k\|_2 \|d_k\|_2} = \frac{s_k^\top M_k s_k}{\|M_k s_k\|_2 \|s_k\|_2}$$

le quotient de Rayleigh de M_k dans la direction s_k et le cosinus de l'angle entre d_k et l'opposé du gradient.

Première étape. On utilise la formule de la [trace](#) pour montrer que pour tout $k \geq 1$:

$$\text{tr } M_{k+1} \leq C_1 k \quad \text{et} \quad \sum_{i=1}^k \frac{\|M_i s_i\|_2^2}{s_i^\top M_i s_i} = \sum_{i=1}^k \frac{q_i}{\cos^2 \theta_i} \leq C_1 k. \quad (11.13)$$

En effet, comme $f \in \mathcal{C}^{1,1}$, on a $\|y_k\|_2^2 / (y_k^\top s_k) \leq C_2$ (proposition 3.69) et donc la formule de la [trace](#) (proposition ??) donne

$$\text{tr } M_{k+1} \leq \text{tr } M_k + C_2 - \frac{\|M_k s_k\|_2^2}{s_k^\top M_k s_k} \leq \text{tr } M_1 + C_2 k - \sum_{i=1}^k \frac{\|M_i s_i\|_2^2}{s_i^\top M_i s_i}.$$

On en déduit la première estimation de (11.13) avec $C_1 := \text{tr } M_1 + C_2$. D'autre part, comme M_{k+1} est définie positive, on a $\text{tr } M_{k+1} \geq 0$ et l'inégalité précédente donne la seconde estimation de (11.13).

Deuxième étape. En utilisant la majoration de la [trace](#) (11.13) et une minoration du déterminant, on obtient une minoration de la plus petite valeur propre de M_{k+1} , donc une minoration du pas. On montre en fait que cela conduit à la borne inférieure suivante sur le pas moyen : pour tout $k \geq 1$,

$$\left(\prod_{i=1}^k \alpha_i \right)^{\frac{1}{k}} \geq C_3. \quad (11.14)$$

En effet, par la formule du déterminant (proposition ??) et la règle de Wolfe (6.11b) :

$$\begin{aligned}
 \det M_{k+1} &= \frac{y_k^\top s_k}{s_k^\top M_k s_k} \det M_k \\
 &\geq \frac{(1-\omega_2)}{\alpha_k} \det M_k \\
 &\geq \frac{(1-\omega_2)^k}{\prod_{i=1}^k \alpha_i} \det M_1 \\
 &\geq \frac{C_4^k}{\prod_{i=1}^k \alpha_i},
 \end{aligned}$$

avec $C_4 := (1-\omega_2) \min(1, \det M_1)$. D'autre part, en utilisant l'inégalité géométrico-arithmétique (3.76), (11.13) et le fait que $k^n \leq (e^n)^k$:

$$\det M_{k+1} \leq \left(\frac{1}{n} \operatorname{tr} M_{k+1} \right)^n \leq \left(\frac{C_1}{n} k \right)^n \leq C_5^k,$$

avec $C_5 := \max(1, C_1/n)^n e^n$. Les deux estimations ci-dessus conduisent à (11.14) avec $C_2 := C_4/C_5$.

Troisième étape. En exploitant la deuxième estimation de (11.13), on obtient une majoration du pas moyen pondéré. On se défera de la pondération dans le raisonnement par l'absurde de la quatrième étape. Pour tout $k \geq 1$:

$$\left(\prod_{i=1}^k \alpha_i \|g_i\|_2^2 \right)^{\frac{1}{k}} \leq \frac{C_6}{k}. \quad (11.15)$$

En effet,

$$\sum_{i=1}^k \frac{\|M_i s_i\|_2^2}{s_i^\top M_i s_i} = \sum_{i=1}^k \frac{\alpha_i \|g_i\|_2^2}{-g_i^\top s_i} \leq C_1 k. \quad (11.16)$$

On utilise ensuite deux fois l'inégalité géométrico-arithmétique (3.76) et la règle de Wolfe (6.11a)

$$\begin{aligned}
 \prod_{i=1}^k \alpha_i \|g_i\|_2^2 &\leq C_1^k \prod_{i=1}^k (-g_i^\top s_i) \quad [(11.16) \text{ et } (3.76)] \\
 &\leq \left(\frac{C_1}{k} \sum_{i=1}^k (-g_i^\top s_i) \right)^k \quad [(3.76)] \\
 &\leq \left(\frac{C_1}{\omega_1 k} \sum_{i=1}^k (f(x_i) - f(x_{i+1})) \right)^k \quad [(6.11a)] \\
 &= \left(\frac{C_1}{\omega_1 k} (f(x_1) - f(x_{k+1})) \right)^k.
 \end{aligned}$$

On en déduit (11.15), avec $C_6 := C_1(f(x_1) - f_{\min})/\omega_1$, où $f_{\min} \in \mathbb{R}$ est un minorant de $\{f(x_k)\}$.

Quatrième étape. On conclut par un raisonnement par l'absurde. Supposons que le résultat ne soit pas vrai. Alors $\|g_k\| \geq C_7 > 0$ et (11.15) donne

$$\left(\prod_{i=1}^k \alpha_i \right)^{\frac{1}{k}} \leq \frac{C_6}{C_7^2 k},$$

qui est en contradiction avec (11.14). \square

Observons que si l'on avait supposé f fortement convexe, on n'aurait pu utiliser de la recherche linéaire de Wolfe que ce qui est contenu dans la condition de Zoutendijk (6.21). En effet la formule du déterminant aurait donné pour une constante générique strictement positive C :

$$\det M_{k+1} \geq C \frac{\|s_k\|^2}{s_k^\top M_k s_k} \det M_k \geq \frac{C^k}{\prod_{i=1}^k q_i},$$

qui avec la formule de la **trace** aurait conduit à

$$\left(\prod_{i=1}^k q_i \right)^{\frac{1}{k}} \geq C.$$

Alors la seconde estimation de (11.13) et l'inégalité géométrico-arithmétique (3.76) auraient donné

$$\left(\prod_{i=1}^k \frac{q_i}{\cos^2 \theta_i} \right)^{\frac{1}{k}} \leq \frac{1}{k} \sum_{i=1}^k \frac{q_i}{\cos^2 \theta_i} \leq C.$$

Donc

$$C \leq \left(\prod_{i=1}^k q_i \right)^{\frac{1}{k}} \leq C \left(\prod_{i=1}^k \cos^2 \theta_i \right)^{\frac{1}{k}} \leq \frac{C}{k} \sum_{i=1}^k \cos^2 \theta_i.$$

On en aurait déduit que $\sum_{k \geq 0} \cos^2 \theta_i = \infty$ et, avec (6.21), que $\liminf \|g_k\| = 0$. Lorsque f est seulement convexe, nous avons vu que la règle de Wolfe intervenait à plusieurs endroits dans la démonstration.

Corollaire 11.7 *Sous les hypothèses du théorème 11.6, si la fonction f est fortement convexe, alors la suite $\{x_k\}$ converge vers l'unique minimum de f .*

DÉMONSTRATION. Si f est fortement convexe, il existe un unique point x_* minimisant f sur \mathbb{R}^n .

Montrons qu'il existe une sous-suite de $\{x_k\}$ qui converge vers x_* . La suite $\{x_k\}$ est bornée (elle est dans $\{x : f(x) \leq f(x_1)\}$ qui est borné). Dès lors, d'après le théorème, on peut trouver une sous-suite convergente $\{x_k\}_{k \in \mathcal{K}}$, telle que $g_k \rightarrow 0$. On en déduit que $x_k \rightarrow x_*$ lorsque $k \rightarrow \infty$ dans \mathcal{K} (ça ne peut pas être un autre point du fait de la continuité du gradient et de l'unicité du point annulant le gradient).

On a alors que $f(x_k) \rightarrow f(x_*)$. En effet, la suite $\{f(x_k)\}$ est décroissante et bornée inférieurement. Donc elle converge et ça ne peut être que vers $f(x_*)$, car une sous-suite converge vers cette valeur.

Il reste à conclure. De toute sous-suite $\{x_k\}_{k \in \mathcal{K}'}$, on peut extraire une sous-suite convergente (elle est bornée), qui ne peut converger que vers x_* (car $f(x_k)$ converge vers $f(x_*)$). Alors, toute la suite $\{x_k\}$ converge vers x_* . \square

Convergence locale**Terminaison quadratique**

Indeed for many years the only intrusion of theory on my research was the point of view that, if an algorithm does not perform well when the objective function is quadratic, then it is unlikely that it will be efficient in general use. This rule of thumb and trying to make good use of available information were the main considerations that helped me to develop several successful algorithms for unconstrained calculations.

M.J.D. POWELL [495; 1991].

L'algorithme de BFGS peut être interprété comme une extension de l'algorithme du gradient conjugué (GC) préconditionné. Si on l'utilise pour minimiser une fonction quadratique strictement convexe, donc de la forme

$$x \in \mathbb{R}^n \mapsto f(x) = \frac{1}{2}x^\top Ax - b^\top x,$$

où la matrice A est d'ordre n symétrique définie positive et $b \in \mathbb{R}^n$, en faisant de la recherche linéaire exacte, alors l'algorithme génère les mêmes itérés que l'algorithme du GC préconditionné par la matrice $W_1 = M_1^{-1}$. Cette propriété est connue sous le nom de *terminaison quadratique*.

Proposition 11.8 (terminaison quadratique) *Si $f : \mathbb{R}^n \rightarrow \mathbb{R}$ est quadratique strictement convexe. Alors l'algorithme de BFGS avec recherche linéaire exacte est bien défini et génère les mêmes itérés que l'algorithme du gradient conjugué préconditionné par la matrice $W_1 = M_1^{-1}$. En particulier, il converge en au plus n itérations.*

DÉMONSTRATION. On montre par récurrence que l'on a pour tout $k \geq 1$ tel que $g_k \neq 0$:

$$W_i g_k = W_1 g_k, \quad i = 1, \dots, k-1, \quad (11.17)$$

$$d_k = d_k^{\text{GC}} := \begin{cases} -W_1 g_1 & \text{si } k = 1 \\ -W_1 g_k + \frac{g_k^\top W_1 g_k}{g_{k-1}^\top W_1 g_{k-1}} d_{k-1} & \text{si } k \geq 2, \end{cases} \quad (11.18)$$

où d_k^{GC} est la direction du GC préconditionné par la matrice W_1 .

Si $g_1 \neq 0$ et $k = 1$, les deux identités sont clairement vérifiées puisque la première direction de l'algorithme de BFGS est $d_1 = -W_1 g_1$.

Supposons à présent que les identités (11.17) et (11.18) aient lieu jusqu'à un indice $k \geq 1$ et montrons qu'elles ont également lieu pour l'indice $k+1$ si $g_{k+1} \neq 0$. Sous l'hypothèse de récurrence, les itérés x_1, \dots, x_{k+1} de l'algorithme de BFGS sont identiques à ceux générés par le GC préconditionné par W_1 . Dès lors $s_i^\top g_{k+1} = 0$ pour $i = 1, \dots, k$ (exercice 8.3) et, par la formule (11.10),

$$W_{i+1} g_{k+1} = \left(I - \frac{s_i y_i^\top}{y_i^\top s_i} \right) W_i g_{k+1}, \quad \text{pour } i = 1, \dots, k.$$

On a aussi $g_i^\top W_1 g_{k+1} = 0$ pour $i = 1, \dots, k$ (exercice 8.3), et donc $y_i^\top W_1 g_{k+1} = 0$ pour $i = 1, \dots, k-1$. L'identité ci-dessus avec $i = 1, \dots, k-1$ permet alors d'obtenir (11.17) par récurrence sur i . L'identité ci-dessus avec $i = k$ donne

$$d_{k+1} = -W_{k+1} g_{k+1} = -W_1 g_{k+1} + \frac{y_k^\top W_1 g_{k+1}}{y_k^\top d_k} d_k.$$

On en déduit (11.18) en notant que $y_k^\top W_1 g_{k+1} = g_{k+1}^\top W_1 g_{k+1}$ et $y_k^\top d_k = -g_k^\top d_k = g_k^\top W_1 g_k$ (on utilise l'hypothèse de récurrence et $g_k^\top d_{k-1} = 0$). \square

11.2.4 Mise en œuvre de l'algorithme de BFGS

Formules de mise à jour directe ou inverse

A priori, il paraît plus intéressant de mettre à jour W_k , l'inverse de M_k par (11.10), que M_k par (11.9), de manière à ne pas devoir résoudre le système linéaire $M_k d_k = -g_k$ pour déterminer la direction de descente d_k à l'étape 2 de l'algorithme. Dans cette approche, la direction $d_k = -W_k g_k$ est obtenue en $O(n^2)$ opérations, alors que la résolution d'un système linéaire en demande $O(n^3)$ en général. On notera aussi que la mise à jour de W_k par (11.10) peut se faire en $O(n^2)$, par la suite d'opérations suivantes :

$$\begin{aligned} \widetilde{W}_k &:= \left(I - \frac{s_k y_k^\top}{y_k^\top s_k} \right) W_k = W_k - \bar{s}_k (W_k \bar{y}_k)^\top \\ W_{k+1} &:= \widetilde{W}_k \left(I - \frac{y_k s_k^\top}{y_k^\top s_k} \right) + \frac{s_k s_k^\top}{y_k^\top s_k} = \widetilde{W}_k - (\widetilde{W}_k \bar{y}_k) \bar{s}_k^\top + \bar{s}_k \bar{s}_k^\top, \end{aligned}$$

dans lesquelles on a noté $\bar{s}_k := s_k / (y_k^\top s_k)^{1/2}$ et $\bar{y}_k := y_k / (y_k^\top s_k)^{1/2}$. Évidemment, \widetilde{W}_k et W_{k+1} peuvent être stockées dans W_k . Dans les problèmes avec contraintes, on doit mettre à jour la matrice directe M_k et cela se fait parfois par la formule suivante

$$M_{k+1} = M_k + u_k v_k^\top + v_k u_k^\top + v_k v_k^\top, \quad (11.19)$$

qui est équivalente à (11.9) si on prend $\bar{s}_k := s_k / (s_k^\top M_k s_k)^{1/2}$, $\bar{y}_k := y_k / (y_k^\top s_k)^{1/2}$, $u_k := M_k \bar{s}_k$ et $v_k := \bar{y}_k - M_k \bar{s}_k$.

En pratique il est préférable d'utiliser (11.10) plutôt que (11.11). Cette dernière formule est en effet instable, du fait des erreurs d'arrondi qui y sont moins bien contrôlées et qui peuvent rendre W_{k+1} indéfinie. Cette affirmation relève de notre propre expérience avec ces formules, mais nous ne connaissons pas d'étude précise sur le sujet. On observe toutefois que par le second terme de (11.11), on *retranche* une matrice **semi-définie positive**, alors que dans (11.10) on fait la *somme* de deux matrices semi-définies positives. Dans la procédure de calcul proposée ci-dessus, dans laquelle on évite de devoir former la matrice $(I - \bar{s}_k \bar{y}_k^\top)$ et de faire un produit de deux matrices (ce qui requiert $O(n^3)$ opérations), on ne bénéficie pas entièrement de cette bonne propriété, car on retrouve des différences de matrices pouvant entraîner de l'instabilité. De ce point de vue, on pourra aussi préférer la formule (11.19) pour la mise à jour de la matrice directe.

Certains auteurs [248, 258] préfèrent mettre à jour les facteurs de Cholesky L_k de $M_k = L_k L_k^\top$ de manière à contrôler la définie positivité de cette matrice qui peut être perdue du fait des erreurs d'arrondi (pour les problèmes difficiles seulement). Cette mise à jour des facteurs de Cholesky de M_k peut se faire en $O(n^2)$ opérations. Le calcul de la direction de descente d_k demande aussi $O(n^2)$ opérations (il y a deux systèmes linéaires triangulaires à résoudre), alors qu'il en faudrait $O(n^3)$ si M_k n'était pas connue par ses facteurs triangulaires.

D'autres auteurs enfin [259, 494, 550] utilisent la troisième approche qui consiste à mettre à jour des facteurs Z_k de $W_k = Z_k Z_k^\top$ (qui ne sont pas de Cholesky). Cela permet d'avoir un meilleur contrôle de la définie positivité de W_k . Cette approche jette un nouvel éclairage sur les liens avec l'algorithme du gradient conjugué (voir à ce sujet les propositions 11.3 et 11.8).

Choix de la matrice initiale

Le choix de la matrice symétrique définie positive initiale M_1 est crucial pour un bon fonctionnement de l'algorithme de BFGS. On conçoit en effet que prendre $M_1 = I$ peut avoir des effets désastreux, parce que la matrice identité peut être beaucoup trop grande ou trop petite, selon les cas, et que, $M_{k+1} - M_k$ étant une matrice de rang 2, la suite $\{M_k\}$ ne va pas évoluer rapidement (on tient d'ailleurs beaucoup à cette propriété de stabilité de la formule de BFGS).

En toute logique, il faudrait prendre comme matrice initiale M_1 une matrice symétrique définie positive approchant convenablement $\nabla^2 f(x_1)$. On dispose parfois d'une telle matrice. Par exemple, cela peut être une approximation définie positive de la diagonale de cette hessienne (on rappelle qu'en toute généralité, celle-ci est aussi coûteuse à évaluer que la hessienne complète, voir l'exercice ??) ou une partie définie positive de $\nabla^2 f(x_1)$ (comme $J_1^\top J_1$ dans les problèmes de moindres-carrés lorsque la jacobienne $J_1 = r'(x_1)$ du résidu au point initial est calculable à peu de frais et est injective, voir la section 19.3).

Parfois, il est intéressant de prendre pour M_1 , la matrice obtenue en fin de résolution par l'algorithme de BFGS d'un problème voisin. Dans ce dernier cas, on parle de *démarrage à chaud*.

Le plus souvent, cependant, on ne dispose d'aucune matrice « naturelle » M_1 et il faut *démarrer à froid*. On prend alors $M_1 = I$ ou $W_1 = I$. Ensuite $M_2 = \text{BFGS}(\sigma_1 I, y_1, s_1)$ ou $W_2 = \overline{\text{BFGS}}(\bar{\sigma}_1 I, y_1, s_1)$, avec

$$\sigma_1 := \frac{y_1^\top s_1}{\|s_1\|_2^2} \quad \text{et} \quad \bar{\sigma}_1 := \frac{y_1^\top s_1}{\|y_1\|_2^2}. \quad (11.20)$$

Pour $k \geq 2$, on prend comme d'habitude $M_{k+1} = \text{BFGS}(M_k, y_k, s_k)$ ou $W_{k+1} = \overline{\text{BFGS}}(W_k, y_k, s_k)$. Les facteurs scalaires σ_1 et $\bar{\sigma}_1$, connus sous le nom de *facteurs d'Oren-Luenberger*, sont déterminés par le souci d'avoir $\sigma_1 I$ et $\bar{\sigma}_1 I$ les plus proches possibles, dans des sens très variés, du *sous-espace affine* $\{M \in \mathcal{S}^n : y_1 = M s_1\}$ des matrices symétriques vérifiant l'équation de quasi-Newton (voir l'exercice 11.3). On notera que σ_1 et $\bar{\sigma}_1$ sont tous les deux strictement positifs (on suppose $y_1^\top s_1 > 0$). On notera également que, par l'inégalité de Cauchy-Schwarz, $\sigma_1 \leq (\bar{\sigma}_1)^{-1}$ sans que l'on ait nécessairement égalité, si bien que générer les matrices directes ou inverses avec les initialisations ci-dessus ne donnent pas le même algorithme. On pourrait d'ailleurs

aussi prendre $\sigma_1 = \|y_1\|_2^2/(y_1^\top s_1)$ et $\bar{\sigma}_1 = \|s_1\|_2^2/(y_1^\top s_1)$. On ne connaît pas de critère faisant l'unanimité permettant de faire le bon choix des facteurs σ_1 ou $\bar{\sigma}_1$; on utilise souvent (11.20).

Mise à l'échelle répétée

Nous avons dit que la suite $\{M_k\}$ était très stable. Hélas parfois trop ! Si l'initialisation n'est pas satisfaisante et que les itérés progressent rapidement, on aimerait que les matrices évoluent plus vite pour s'adapter à la hessienne courante. Il peut donc être intéressant de *catalyser* la mise à jour en multipliant M_k ou W_k par un facteur scalaire *avant* leur mise à jour, mais *après* leur utilisation. On prend donc $M_{k+1} = \text{BFGS}(\sigma_k M_k, y_k, s_k)$ ou $W_{k+1} = \overline{\text{BFGS}}(\bar{\sigma}_k W_k, y_k, s_k)$, avec des scalaires $\sigma_k > 0$ et $\bar{\sigma}_k > 0$ valant

$$\sigma_k := \frac{y_k^\top s_k}{s_k^\top M_k s_k} \quad \text{et} \quad \bar{\sigma}_k := \frac{y_k^\top s_k}{y_k^\top W_k y_k}. \quad (11.21)$$

Ceux-ci sont obtenus par un raisonnement analogue à celui conduisant à (11.20)

Ce qui peut ne pas fonctionner

L'algorithme de BFGS avec recherche linéaire en optimisation demande que le gradient soit calculé avec une bonne précision. Si ce n'est pas le cas, en particulier lorsque le gradient est estimé par différences finies, l'algorithme pourra s'interrompre prématurément parce que la direction générée ne sera pas une direction de descente. En effet, celle-ci s'écrit $d_k = -W_k g_k$, où g_k est le gradient *calculé* et W_k est une matrice définie positive dont on ne contrôle pas le conditionnement $\kappa_2(W_k)$. Comme, selon (6.4), l'inverse de ce dernier minore l'angle θ_k entre d_k et $-g_k$, on pourra avoir un angle θ_k arbitrairement proche de $\pi/2$. Si g_k n'est pas suffisamment précis, il se pourra que la direction d_k calculée par l'algorithme de BFGS ne soit pas de descente en x_k , entraînant l'échec de la recherche linéaire. Si la précision obtenue alors n'est pas suffisante, il n'y a qu'un seul remède : calculer le gradient avec plus de précision (éviter la discrétisation d'un gradient en dimension infinie, faire de la différentiation automatique ou manuelle, chasser les erreurs d'arrondi).

Mentionnons aussi, mais sans insister, que des exemples de non-convergence de l'algorithme de BFGS avec recherche linéaire ont été découverts [411 ; 2004], soit environ 35 ans après son introduction. Il y a peu de chance toutefois que cette non-convergence s'observe en pratique.

11.2.5 L'algorithme ℓ -BFGS

Si le problème est de grande taille, disons avec un nombre de variables n supérieur à 500 (ce seuil dépend en partie des performances des ordinateurs), les méthodes de quasi-Newton telles qu'on les a présentées ne sont plus adaptées, pour deux raisons au moins. D'abord parce qu'il peut être impossible de mettre en mémoire une matrice d'ordre n . Ensuite, même si le stockage d'une telle matrice est possible, sa mise à jour peut prendre beaucoup de temps (celle-ci requiert en effet de l'ordre de n^2 opérations) et son adaptation à la hessienne courante peut être lente (dans les formules de

mise à jour vues, la différence entre deux matrices successives est de faible rang, si bien qu'il faut normalement un nombre de mises à jour proportionnel à n pour que l'approximation courante devienne correcte et que l'algorithme devienne efficace).

Une belle découverte, très utile en pratique, a été de constater que l'on peut *adapter* beaucoup de méthodes de quasi-Newton à des problèmes de très grande taille en appliquant les principes suivants :

1. on ne forme l'approximation de la hessienne qu'en utilisant un nombre limité de couples (y_i, s_i) , disons m , avec $m \ll n$ (typiquement $m \simeq 5 \dots 10$) ;
2. on ne mémorise que ces couples, pas la matrice elle-même, et les mises à jour se font à partir d'une matrice initiale peu encombrante en mémoire (typiquement une matrice diagonale) ; la matrice approchant la hessienne (ou son inverse) est donc définie de manière implicite, sans représentation en mémoire ;
3. on calcule le produit de cette matrice (ou de son inverse) par un vecteur (comme cela est requis par la formule (11.1)) par un algorithme qui doit être efficace et peut se passer d'une représentation explicite de la matrice W_k .

Une technique de mise à jour reposant sur ces principes est dite à *mémoire limitée*. La sélection des bons couples (y_i, s_i) est un problème délicat qui n'a pas trouvé de réponse satisfaisante, si bien que l'on sélectionne en général les m plus récents, ceux sensés donner la meilleure information sur la hessienne courante. De même, le choix de la matrice initiale est encore aujourd'hui un sujet de recherche ; certains auteurs [399] choisissent la matrice identité multipliée par un des scalaires de (11.21) ; d'autres [244] utilisent des matrices diagonales, elles-mêmes mises à jour au cours des itérations. Enfin, nous verrons qu'une conséquence de la différentiation automatique en mode inverse (section 5.5.3) est que le produit de la matrice implicite par un vecteur pourra se faire de manière efficace si la forme quadratique associée à cette matrice s'évalue rapidement.

Soyons plus concret et décrivons la mise à jour par la formule de BFGS inverse (11.10) de l'approximation à *mémoire limitée* de l'inverse de la hessienne, qui suit les principes énoncés aux points 1-3 ci-dessus. L'algorithme correspondant est appelé l'algorithme ℓ -BFGS [458]. La matrice implicite (c.-à-d., non stockée) à l'itération k est toujours notée W_k . Au moment de la mise à jour, on suppose que x_k et x_{k+1} sont connus ; le premier itéré est noté x_1 . Comme mentionné ci-dessus, on utilise en général les couples les plus récents :

$$\{(y_{k-\bar{m}+i}, s_{k-\bar{m}+i}) : i = 1, \dots, \bar{m}\}, \quad (11.22)$$

où

$$\bar{m} := \min(m, k).$$

Partant d'une matrice initiale W_k^0 facilement mémorisable et qu'il faut spécifier, la matrice implicite suivante W_{k+1} est obtenue par

$$\begin{aligned} W_k^i &:= \overline{\text{BFGS}}(W_k^{i-1}, y_{k-\bar{m}+i}, s_{k-\bar{m}+i}), \quad \text{pour } i = 1, \dots, \bar{m} \\ W_{k+1} &= W_k^{\bar{m}}, \end{aligned}$$

où $\overline{\text{BFGS}}$ symbolise la formule de BFGS inverse (11.10). Il s'agit ici d'une description formelle de W_{k+1} , car cette matrice n'est pas stockée.

Notre but à présent est de montrer que le produit de la matrice implicite W_{k+1} , représentée par la matrice W_k^0 et les couples (11.22), par un vecteur v peut se faire aisément. Dans (11.1), $v = g_{k+1}$, mais ce qui suit est valable pour un vecteur v arbitraire, ce qui pourra être utile pour les problèmes avec contraintes. On note

$$\begin{aligned} \text{pour } i = 1, \dots, \bar{m} : \quad & \rho_i := (y_{k-\bar{m}+i}^\top s_{k-\bar{m}+i})^{-1} \quad \text{et} \quad V_i := I - \rho_i y_{k-\bar{m}+i} s_{k-\bar{m}+i}^\top, \\ \text{pour } i = 0, \dots, \bar{m} : \quad & \text{la fonction } v \in \mathbb{R}^n \mapsto \varphi_i(v) := (v^\top W_k^i v)/2 \end{aligned}$$

et on définit par récurrence

$$q_{\bar{m}} = v \quad \text{et} \quad q_{i-1} = V_{k-\bar{m}+i} q_i \quad (\text{pour } i = \bar{m}, \dots, 1).$$

Alors, grâce à (11.10), on a par récurrence

$$\begin{aligned} \varphi_{\bar{m}}(v) &= \varphi_{\bar{m}}(q_{\bar{m}}) \\ &= \frac{1}{2} q_{\bar{m}}^\top V_k^\top W_k^{\bar{m}-1} V_k q_{\bar{m}} + \frac{\rho_k}{2} (s_k^\top q_{\bar{m}})^2 \\ &= \varphi_{\bar{m}-1}(q_{\bar{m}-1}) + \frac{\rho_k}{2} (s_k^\top q_{\bar{m}})^2 \\ &= \varphi_0(q_0) + \sum_{i=1}^{\bar{m}} \frac{\rho_{k-\bar{m}+i}}{2} (s_{k-\bar{m}+i}^\top q_i)^2. \end{aligned}$$

Cette valeur $\varphi_{\bar{m}}(v)$ se calcule donc par l'algorithme

$$\begin{aligned} & \varphi := 0; \\ & q := v; \\ & \text{for } (i := \bar{m}; i \geq 1; i := i - 1) \{ \\ & \quad \alpha_i := s_{k-\bar{m}+i}^\top q; \\ & \quad \varphi := \varphi + \frac{\rho_{k-\bar{m}+i}}{2} \alpha_i^2; \\ & \quad q := q - \rho_{k-\bar{m}+i} \alpha_i y_{k-\bar{m}+i}; \\ & \quad \} \\ & \varphi := \varphi + \frac{1}{2} q^\top W_k^0 q; \end{aligned} \tag{11.23}$$

Comme $W_{k+1}v$ est le gradient en v de $v \mapsto \varphi_{\bar{m}}(v)$, on peut calculer ce vecteur en exécutant le code adjoint de (11.23) (voir la section 5.5.3 et plus spécialement le code (5.32) et l'exercice 5.9). On note $\bar{\varphi}$, \bar{q} et $\bar{\alpha} := (\bar{\alpha}_1, \dots, \bar{\alpha}_{\bar{m}})$ les variables adjointes de φ , q et $\alpha := (\alpha_1, \dots, \alpha_{\bar{m}})$. Ce code adjoint s'écrit alors

$$\begin{aligned} & \bar{\varphi} := 1; \\ & \bar{q} := 0; \\ & \bar{\alpha} := 0; \\ & \bar{q} := W_k^0 q; \\ & \text{for } (i := 1; i \leq \bar{m}; i := i + 1) \{ \\ & \quad \bar{\alpha}_i := \bar{\alpha}_i - \rho_{k-\bar{m}+i} y_{k-\bar{m}+i}^\top \bar{q}; \\ & \quad \bar{\alpha}_i := \bar{\alpha}_i + \rho_{k-\bar{m}+i} \alpha_i \bar{\varphi}; \\ & \quad \bar{q} := \bar{q} + \bar{\alpha}_i s_{k-\bar{m}+i}; \\ & \quad \bar{\alpha}_i := 0; \\ & \quad \} \\ & \bar{v} := \bar{q}; \\ & \bar{q} := 0; \\ & \bar{\varphi} := 0; \end{aligned} \tag{11.24}$$

La valeur de $W_{k+1}v$ est obtenu à la sortie de l'algorithme (11.24) dans le vecteur \bar{v} . En combinant (11.23) et (11.24), en ne conservant que les instructions qui sont utiles au calcul de $W_{k+1}v$ et en plaçant \bar{q} dans q et tous les $\bar{\alpha}_i$ dans β , on obtient l'algorithme suivant :

$$\begin{aligned}
 & q := v; \\
 & \text{for } (i := \bar{m}; i \geq 1; i := i - 1) \{ \\
 & \quad \alpha_i := s_{k-\bar{m}+i}^\top q; \\
 & \quad q := q - \rho_{k-\bar{m}+i} \alpha_i y_{k-\bar{m}+i}; \\
 & \} \\
 & q := W_k^0 q; \\
 & \text{for } (i := 1; i \leq \bar{m}; i := i + 1) \{ \\
 & \quad \beta := \rho_{k-\bar{m}+i} (\alpha_i - y_{k-\bar{m}+i}^\top q); \\
 & \quad q := q + \beta s_{k-\bar{m}+i}; \\
 & \}
 \end{aligned} \tag{11.25}$$

La valeur de $W_{k+1}v$ est obtenu à la sortie de l'algorithme (11.25) dans le vecteur q . L'approche que nous avons utilisée pour obtenir l'algorithme de calcul (11.25) montre pourquoi celui est efficace : il est fondé sur l'algorithme compact (11.23) calculant rapidement $\frac{1}{2}v^\top W_{k+1}v$ et le mode inverse de différentiation automatique, que l'on sait ne pas dégrader l'efficacité de (11.23).

Le nombre d'opérations pour le calcul de la direction de ℓ -BFGS par (11.25) est approximativement $4mn$ additions et $4mn$ multiplications.

Pour conclure, résumons les instructions de l'algorithme ℓ -BFGS.

Algorithme 11.9 (ℓ -BFGS)

0. On se donne deux constantes ω_1 et ω_2 pour la recherche linéaire de Wolfe :
 $0 < \omega_1 < \frac{1}{2}$ et $\omega_1 < \omega_2 < 1$.
 Choix d'un itéré initial $x_1 \in \mathbb{R}^n$ et d'un nombre de mises à jour m .
 Initialisation : $k := 1$, $\bar{m} = 0$.
 1. *Test d'arrêt* : si $\nabla f(x_k) = 0$, arrêt de l'algorithme.
 2. *Calcul de la direction de descente* d_k : on détermine une matrice initiale W_k^0 et on calcule d_k par l'algorithme (11.25) (sauf si $\bar{m} = 0$, auquel cas on prend $d_k = -W_k^0 g_k$).
 3. *Recherche linéaire de Wolfe* : trouver un pas $\alpha_k > 0$ tel que l'on ait (6.11a) et (6.11b).
 4. $x_{k+1} := x_k + \alpha_k d_k$.
 5. Si $k > m$, effacer (y_{k-m}, s_{k-m}) de la mémoire. Stocker (y_k, s_k) .
 6. Accroître k de 1 et aller en 1.
-

On résout aujourd'hui couramment des problèmes avec $n \simeq 10^6 \dots 10^8$ par l'algorithme ℓ -BFGS, en particulier en météorologie [145, 146] ou océanographie.

Notes ▲

L'algorithme de la sécante pour trouver un zéro d'une fonction non linéaire remonte au moins à Newton, probablement vers 1665 [617 ; I, p. 489-491]. [639]

Le théorème 11.6 et son corollaire 11.7 sont dus à Powell [490; 1976]. Ce résultat ne peut pas s'étendre aux fonctions non convexes : on a trouvé des contre-exemples lorsque l'algorithme est globalisé par une recherche linéaire exacte [492; 1984], par celle de Wolfe [151; 2002] ou par celle prenant le pas α_k donnant le minimum exact le long de la direction de recherche et assurant la condition d'Armijo (6.9) [411; 2004]. Ce point noir de l'algorithme n'empêche pas son utilisation avec succès. Certains auteurs ont proposé des modifications de l'algorithme de manière à pouvoir en montrer la convergence sur des fonctions non convexes, mais cette manière de procéder n'a pas l'approbation de tous. Ainsi Li et Fukushima [393; 2001] ont obtenu un résultat de convergence globale et superlinéaire avec une modification du vecteur y_k en $y_k + r_k s_k$, où le scalaire $r_k \sim \|g_k\|$. Comme lot de consolation, mentionnons que l'algorithme de DFP, bien moins efficace que l'algorithme de BFGS, converge pour les problèmes avec 2 variables et une recherche linéaire prenant le premier minimum local [496; 2000].

Prise en compte de la parcimonie : technique originale due à Toint [592; 1977]; difficultés de non-existence et de non-définie-positivité relevées par Sorensen [564; 1982]; approche de Fletcher [213; 1995] utilisant la fonction ψ de Byrd et Nocedal.

L'algorithme ℓ -BFGS de la section 11.2.5 a été proposé par Nocedal [458]. La dérivation de l'algorithme (11.25) présentée, utilisant la différentiation automatique, est reprise de [246]. Dans les problèmes avec contraintes, c'est l'approximation de hessiennes directes dont on a besoin ; on peut en construire des approximations quasi-newtoniennes à mémoire limitée [103, 102]. Le choix du bon nombre m de mises à jour reste aujourd'hui un sujet intrigant. De manière surprenante, il n'est pas vrai que prendre m le plus grand possible améliore les performances de l'algorithme. Le nombre d'itérations pour atteindre un seuil d'optimalité donné peut facilement varier avec un facteur 3 lorsque m s'échelonne, disons de 5 à 50, avec une efficacité optimale pour des valeurs erratiques de m . Boggs et Byrd [68] ont proposé des techniques efficaces pour déterminer m de manière adaptative au cours des itérations. Lin, Harchaoui et Mairal [396; 2017] proposent un algorithme fondé sur la technique ℓ -BFGS, destiné à minimiser $f = f_0 + \psi$, où f_0 est convexe $\mathcal{C}^{1,1}$ et ψ est convexe non lisse (par exemple $\psi = \|\cdot\|_1$ ou $\psi = \mathcal{I}_C$), qui peut se voir comme une méthode ℓ -BFGS inexacte sur la régularisée de Moreau-Yosida de f (section 3.7.2).

Des algorithmes de BFGS et de ℓ -BFGS pour la minimisation d'une fonction réelle de variables complexes sont proposés dans [562].

Divers résultats ont aussi été obtenus pour minimiser une fonction non lisse avec l'algorithme de BFGS [392, 637, 296].

Exercices

11.1. *Formule de BFGS par blocs.* Soient $Y, S \in \mathbb{R}^{n \times p}$ avec S injective et $M \in \mathcal{S}_{++}^n$. On considère le problème

$$\begin{cases} \min_{\bar{M}} \psi(M^{-1/2} \bar{M} M^{-1/2}) \\ Y = \bar{M} S \\ \bar{M} \in \mathcal{S}_{++}^n, \end{cases}$$

où $\psi : \mathcal{S}_{++}^n \rightarrow \mathbb{R}$ est la fonction définie par (11.6). Montrez que ce problème a une solution si, et seulement si, $Y^\top S \in \mathcal{S}_{++}^n$. Montrez que, sous cette condition, la solution \bar{M} du problème est unique et est donnée par l'une des formules suivantes :

$$\begin{aligned}\bar{M} &= M - MS(S^TMS)^{-1}S^TM + Y(Y^TS)^{-1}Y^T \\ \bar{W} &= \left(I - S(Y^TS)^{-1}Y^T\right)W\left(I - Y(Y^TS)^{-1}S^T\right) + S(Y^TS)^{-1}S^T \\ &= W + (S - WY)(Y^TS)^{-1}S^T + S(Y^TS)^{-1}(S - WY)^T \\ &\quad - S(Y^TS)^{-1}Y^T(S - WY)(Y^TS)^{-1}S^T.\end{aligned}$$

où on a noté $W := M^{-1}$ et $\bar{W} := \bar{M}^{-1}$.

- 11.2.** *Formule de mise à jour PSB.* Soient y_k et s_k deux vecteurs de \mathbb{R}^n et M_k une matrice d'ordre n symétrique. On considère le problème en $M \in \mathbb{R}^{n \times n}$ suivant

$$\begin{cases} \min \|M - M_k\|_F \\ y_k = Ms_k \\ M = M^T, \end{cases} \quad (11.26)$$

où $\|\cdot\|_F$ est la **norme de Frobenius** sur l'ensemble des matrices. Montrez que, si $s_k \neq 0$, le problème (11.26) a une solution unique M_{k+1} et que celle-ci s'écrit

$$M_{k+1} = M_k + \frac{(y_k - M_k s_k)s_k^T + s_k(y_k - M_k s_k)^T}{\|s_k\|_2^2} - \frac{(y_k - M_k s_k)^T s_k}{\|s_k\|_2^4} s_k s_k^T. \quad (11.27)$$

Si $s_k = 0$ et $y_k \neq 0$, le problème (11.26) n'a pas de solution. Si $s_k = y_k = 0$, le problème (11.26) a comme unique solution $M_{k+1} = M_k$.

Remarque. La formule (11.27) porte le nom de *formule PSB* (Powell-Symétrique-Broyden). En pratique, elle donne généralement de moins bons résultats que la formule de BFGS. On observera que le cas $s_k = 0$ et $y_k \neq 0$ ne peut pas se présenter en optimisation ; quant au cas $s_k = y_k = 0$, il est normalement signe que l'itéré courant est stationnaire.

- 11.3.** *Initialisation de l'algorithme de BFGS.* Montrez que σ_1 et $\bar{\sigma}_1$ donnés dans (11.20) sont respectivement solutions des problèmes en $(\sigma, M) \in \mathbb{R} \times \mathbb{R}^{n \times n}$ et $(\bar{\sigma}, W) \in \mathbb{R} \times \mathbb{R}^{n \times n}$ suivants

$$\begin{cases} \min_{\sigma, M} \|\sigma I - M\|_F \\ y_1 = Ms_1 \\ M = M^T \end{cases} \quad \text{et} \quad \begin{cases} \min_{\bar{\sigma}, W} \|\bar{\sigma} I - W\|_F \\ W y_1 = s_1 \\ W = W^T. \end{cases}$$

- 11.4.** *Propriétés de l'algorithme de BFGS.* On considère l'algorithme de BFGS pour minimiser une fonction non linéaire f . On note x_k le k -ième itéré, g_k le gradient de f en x_k , $\mathcal{G}_k := \text{vect}\{g_1, \dots, g_k\}$, $W_k = M_k^{-1}$ la matrice générée et \perp_Q l'orthogonalité par rapport au produit scalaire associé à une matrice symétrique définie positive Q . Montrez que pour tout $k \geq 1$, on a

- (i) $W_k v \in W_1(\mathcal{G}_k)$, pour tout $v \in \mathcal{G}_k$; en particulier, $W_k g_k \in W_1(\mathcal{G}_k)$;
- (ii) $W_k v = W_1 v$, pour tout $v \perp_{W_1} \mathcal{G}_k$.

De plus, si $W_1 = \sigma I$, avec $\sigma > 0$, on a

- (iii) $M_k v \in \mathcal{G}_k$, pour tout $v \in \mathcal{G}_k$;
- (iv) $M_k v = \sigma v$, pour tout $v \perp_I \mathcal{G}_k$.

Remarque. Ce résultat montre que la direction de recherche est dans $W_1(\mathcal{G}_k)$.