Algorithmes du premier ordre

Il n'y pas d'algorithme meilleur que tout autre quel que soit le critère de performance que l'on adopte, même dans le champ restreint de l'optimisation sans contrainte. Les critères d'appréciation sont en effet multiples : vitesse de convergence des suites générées ou de la valeur optimale, coût interne (nombre d'opérations) ou externe (nombre d'évaluations de fonctions) de l'itération, complexité itérative de l'algorithme (nombre total d'itérations pour atteindre un seuil donné), espace mémoire requis, *etc.* On trouve plutôt tout un éventail de méthodes plus ou moins bien adaptées à des problèmes particuliers. Il faut donc bien connaître les caractéristiques de ces algorithmes pour qu'en présence d'un problème donné, l'on puisse faire un choix à bon escient. Ce chapitre présente un premier ensemble de techniques qui sont toutes utiles à des degrés divers et selon le contexte ; cela sera précisé aux cours des sections qui leur sont consacrées. Elles ont la caractéristique d'être du *premier ordre*, c'est-à-dire de n'utiliser de la fonction à minimiser, que sa valeur et sa dérivée première.

L'algorithme du gradient est une bien mauvaise méthode (elle converge trop lentement), mais son analyse sert de référence à l'étude d'autres algorithmes plus complexes; nous l'aborderons à la section 7.1. Le chapitre étudie l'algorithme proximal pour la minimisation de fonctions convexes (section 7.2). Celui-ci, bien que non implémentable en pratique, sera utilisé pour interpréter certains algorithmes de dualité au chapitre 14.

Connaissances supposées. La section 7.2 sur l'algorithme proximal suppose connues les notions de sous-différentiabilité de fonctions convexes, de point proximal et de régularisée de Moreau-Yosida (sections 3.6 et 3.7).

7.1 Algorithme du gradient

On ne le répétera sans doute jamais assez : *l'algorithme du gradient est une très mauvaise méthode d'optimisation* ! La preuve en est qu'elle requiert génériquement un nombre infini d'itérations pour minimiser une fonction quadratique strictement convexe de deux variables. Sachant que ce problème est équivalent à la résolution d'un système linéaire de deux équations à deux inconnues, dont la solution peut s'obtenir manuellement par élimination d'une des deux variables, on comprend l'ampleur de l'inefficacité. Pire, si le rapport des valeurs propres du hessien de la fonction est grand (mauvais conditionnement du problème), une précision raisonnable sur la solution ne peut jamais être atteinte en un temps de calcul supportable. L'opinion la plus couramment rencontrée est qu'il est toujours préférable d'utiliser l'algorithme de BFGS à mémoire limitée (section 11.2.5), qui a un champ d'application à peu près identique

– oracle (ou simulateur) et espace mémoire semblables – et n'est guère plus difficile à implémenter. Cette opinion reste vraie selon nous, mais ne condamne pas pour autant l'algorithme du gradient. D'abord, on ne peut se passer de son étude, car elle sert de base à celle d'algorithmes plus complexes. Ensuite, il permet d'interpréter d'autres algorithmes qui ne se présentent pas dans leur conception comme un algorithme de gradient, mais qui y sont apparentés (nous pensons à la relaxation lagrangienne et à la méthode des multiplicateurs). Enfin, l'utilisation de l'optimisation dans de nouvelles applications (traitement du signal, analyse des données massives, apprentissage, *etc* [470, 568]) a remis cet algorithme et ses variantes au goût du jour.

Dans cette section, $\mathbb{E}$ est un espace euclidien, dont le produit scalaire est noté $\langle \cdot, \cdot \rangle$ et la norme associée est notée $\| \cdot \|$. L'espace vectoriel $\mathbb{E}$ étant supposé de dimension finie, le nombre de variables des problèmes d'optimisation posés sur $\mathbb{E}$ en est la dimension

$$n := \dim \mathbb{E}.$$

7.1.1 Définition

On s'intéresse dans cette section au problème de minimisation sans contrainte

$$f_* := \inf_{x \in \mathbb{E}} f(x),$$

où $f : \Omega \to \mathbb{R}$ est une fonction définie et différentiable sur un ouvert $\Omega \subseteq \mathbb{E}$.

L'*algorithme du gradient* (ou *de la plus profonde descente*) a déjà été introduit à la section 6.2.1. Il génère une suite $\{x_k\} \subseteq \mathbb{E}$ par la récurrence

$$\boxed{x_{k+1} = x_k - \alpha_k g_k,} \tag{7.1}$$

où $g_k = \nabla f(x_k)$ est le gradient de f pour le produit scalaire de $\mathbb{E}$ et $\alpha_k > 0$ est un pas déterminé par recherche linéaire (section 6.3).

Il s'agit donc d'un algorithme à directions de descente (chapitre 6), dont les directions sont les *antigradients* $-g_k$, des directions opposées au gradient. Ces directions ont la particularité d'avoir un *angle de descente* θ_k défini en (6.2) qui est nul, si bien que son cosinus vaut un :

$$\forall\, k \geqslant 1 : \qquad \cos \theta_k = 1. \tag{7.2}$$

Cette particularité implique des résultats de convergence et de complexité itérative aisé à déterminer.

Le lemme suivant est utilisé pour étudier la vitesse de convergence de la méthode du gradient.

Lemme 7.1 (inégalité de Kantorovitch [345]) *Soient M une matrice d'ordre n symétrique définie positive et $v \in \mathbb{R}^n$ de norme 1. Alors*

$$(v^\mathsf{T} M v)(v^\mathsf{T} M^{-1} v) \leqslant \frac{(\lambda_{\max} + \lambda_{\min})^2}{4 \lambda_{\max} \lambda_{\min}},$$

où $\lambda_{\max}$ et $\lambda_{\min}$ sont les valeurs propres maximale et minimale de M.

DÉMONSTRATION. □

Proposition 7.2 *Soit f une fonction quadratique strictement convexe de hessien H (défini positif). Utilisée pour minimiser cette fonction, la méthode du gradient à pas optimal génère une suite $\{x_k\}$ convergeant q-linéairement vers l'unique minimum x_* de f. Plus précisément,*

$$\frac{\|x_{k+1} - x_*\|_H}{\|x_k - x_*\|_H} \leqslant \frac{\kappa - 1}{\kappa + 1}, \quad \text{pour } k \geqslant 1,$$

où $\|\cdot\|_H = (\cdot^\mathsf{T} H \cdot)^{1/2}$ est la norme associée à H et κ est le conditionnement de H.

DÉMONSTRATION. □

7.2 Algorithme proximal

7.2.1 Définition

Soit $\mathbb{E}$ un espace euclidien (produit scalaire $\langle\cdot,\cdot\rangle$ et norme associée $\|\cdot\|$). À un opérateur auto-adjoint défini positif M, on associe le produit scalaire

$$\langle\cdot,\cdot\rangle_M : (u,v) \in \mathbb{E}\times\mathbb{E} \mapsto \langle u,v\rangle_M := \langle Mu, v\rangle$$

et la norme

$$\|\cdot\|_M : u \in \mathbb{E} \mapsto \|u\|_M := \langle u,u\rangle_M^{1/2}.$$

On s'intéresse dans cette section à la minimisation d'une fonction $f \in \overline{\mathrm{Conv}}(\mathbb{E})$ (c.-à-d., propre, convexe et fermée), qui n'est pas nécessairement différentiable. Soit $\{M_k\}$ une suite d'opérateurs auto-adjoints définis positifs, qui seront utilisés comme *préconditionneurs* de l'algorithme proximal présenté ci-après, et peuvent être générés au fur et à mesure que l'algorithme progresse. Rappelons que le point proximal d'un point $x \in \mathbb{E}$, associé à la fonction f et à l'opérateur M_k, est le point noté et défini par

$$P_k(x) := \operatorname*{arg\,min}_{y\in\mathbb{E}} \left(f(y) + \frac{1}{2}\|y - x\|^2_{R_k^{-1}} \right). \tag{7.3}$$

Cette notion a été introduite à la section 3.7.1. Nous ferons évidemment souvent référence aux résultats de cette section. Par la condition nécessaire et suffisante d'optimalité du problème dans (7.3), on a

$$x_p = P_k(x) \qquad \Longleftrightarrow \qquad \exists\, g_p \in \partial f(x_p) : \quad x_p = x - R_k g_p.$$

où le sous-différentiel $\partial f(x)$ est calculé pour le produit scalaire originel $\langle\cdot,\cdot\rangle$.

L'*algorithme proximal* génère une suite $\{x_k\}$ par la formule

$$\boxed{x_{k+1} := P_k(x_k) = x_k - R_k g_{k+1},} \qquad \text{où} \qquad g_{k+1} \in \partial f(x_{k+1}). \tag{7.4}$$

On peut interpréter cet algorithme de diverses manières.

1) *C'est une méthode de sous-gradient implicite pour minimiser f.*
Le sous-gradient g_{k+1} est en effet évalué en x_{k+1} (qui est inconnu), plutôt qu'en x_k (dans l'algorithme du gradient de la section 7.1). Donc l'équation (7.4) est non linéaire en x_{k+1}, si bien que le nouvel itéré x_{k+1} s'obtiendra en général par un procédé itératif, par exemple celui qui consiste à résoudre itérativement le problème d'optimisation dans (7.3).

2) *C'est un algorithme à directions de descente sur f avec pas unité.*
La direction $-R_k g_{k+1}$ est en effet une direction de descente de f en x_k parce que $R_k g_{k+1}$ est une direction de montée en x_{k+1} et que f est convexe. On a en effet

$$f'(x_{k+1}; R_k g_{k+1}) \geqslant \langle g_{k+1}, R_k g_{k+1}\rangle \geqslant 0, \tag{7.5}$$

où la première inégalité vient du fait que $g_{k+1} \in \partial f(x_{k+1})$ (point (i) de la proposition 3.55). Puis, par le point (iv) de la proposition 3.19 et (7.4) :

$$f'(x_k; x_{k+1}-x_k) \leqslant -f'(x_{k+1}; x_k - x_{k+1}) = -f'(x_{k+1}; R_k g_{k+1}) \leqslant 0.$$

On observe aussi que l'algorithme s'impose systématiquement un pas unité le long de la direction $-R_k g_{k+1}$. On constate aussi que ce pas fait décroître f à chaque itération, puisqu'en prenant $y = x_k$ dans (7.3), on obtient

$$f(x_{k+1}) \leqslant f(x_{k+1}) + \frac{1}{2}\|x_{k+1} - x_k\|^2_{R_k^{-1}} \leqslant f(x_k). \tag{7.6}$$

3) *C'est l'algorithme du gradient, pour le produit scalaire $\langle\cdot,\cdot\rangle_{R_k^{-1}}$, avec pas unité, pour minimiser la régularisée de Moreau-Yosida* de f, à savoir la fonction

$$\tilde f : x \in \mathbb{E} \mapsto \tilde f(x) := \inf_{y\in\mathbb{E}} \left(f(y) + \frac{1}{2}\|y - x\|^2_{R_k^{-1}} \right).$$

En effet, d'après la proposition 3.80, le gradient de $\tilde f$ pour le produit scalaire $\langle\cdot,\cdot\rangle$ s'écrit

$$\nabla \tilde f(x_k) = R_k^{-1}(x_k - x_{k+1}) = g_{k+1}.$$

Dès lors $R_k g_{k+1}$ est le gradient de $\tilde f$ en x_{k+1} pour le produit scalaire $\langle\cdot,\cdot\rangle_{R_k^{-1}}$.
Notons que les itérés générés font aussi décroître $\tilde f$ de façon monotone. En effet, (7.6) se récrit $f(x_{k+1}) \leqslant \tilde f(x_k) \leqslant f(x_k)$, si bien que l'on a

$$\tilde f(x_{k+1}) \leqslant f(x_{k+1}) \leqslant \tilde f(x_k).$$

Voilà un algorithme bien étrange, puisque pour minimiser f, il faut qu'à chaque itération, l'algorithme proximal minimise la fonction

$$f^k : y \in \mathbb{E} \mapsto f^k(y) := f(y) + \frac{1}{2}\|y - x_k\|^2_{R_k^{-1}},$$

qui semble bien être aussi compliquée que f. Ce point de vue doit être relativisé au vu des remarques suivantes.

1) La fonction f^k à minimiser (à chaque itération, ne l'oublions pas) peut être plus attrayante que f, du fait de sa *forte convexité*. Pour certains algorithmes, cette propriété est une aubaine, permettant d'accélérer leur convergence et de mieux la contrôler. Cette fonction a aussi un minimum unique, ce qui n'est pas nécessairement le cas de f.

2) Comme nous le verrons au chapitre 14, certains algorithmes de dualité peuvent être *interprétés* comme des algorithmes proximaux. Ces algorithmes de dualité s'appliquent en fait à des fonctions f dont l'évaluation résulte de la minimisation d'une fonction (le lagrangien) et il n'est pas plus coûteux d'évaluer f que de minimiser f^k (pourvu que $R_k \succ 0$), qui revient à minimiser une autre fonction (le lagrangien augmenté). Ces algorithmes de dualité ont donc tout leur sens. Leur interprétation en termes d'algorithme proximal permet alors d'en obtenir des propriétés difficiles à mettre en évidence autrement.

3) Observons que si f est *séparable*, c'est-à-dire si elle s'écrit comme suit

$$f(x) = \sum_{j=1}^{N} f_j(x_{D_j}),$$

où les D_j sont de *petites* parties disjointes de l'ensemble des indices $[1:n]$, il en est de même de f^k (pourvu que R_k soit diagonale par blocs). C'est une propriété intéressante lorsqu'on cherche à résoudre de grands problèmes par des techniques de décomposition.

4) L'algorithme proximal a un *effet stabilisant*. Lorsque f a plusieurs minimiseurs, l'algorithme génère une suite convergeant vers l'un d'entre eux. L'algorithme proximal est parfois utilisé pour stabiliser des algorithmes qui ne convergeraient pas sans modification lorsque le problème considéré devient singulier.

7.2.2 Convergence

Les propositions 7.3 et 7.4 ci-desous explorent quelques propriétés de convergence de l'algorithme proximal, dans des situations de plus en plus restrictives.

Un des rôles de l'opérateur R_k dans l'itération de l'algorithme proximal peut se comprendre en examinant l'influence de son ordre de grandeur dans le problème (7.3) dont x_{k+1} est la solution. Si R_k est *très petit*, la pénalité introduite par le terme quadratique est *très grande*, si bien que le déplacement $x_{k+1} - x_k$ peut devenir *très petit* au point d'empêcher le progrès vers une solution (de ce point de vue, l'opérateur $R_k = +\infty I$ conviendrait parfaitement, puisqu'il permettrait de résoudre le problème de minimisation en une seule itération). Comme nous le verrons dans la démonstration de la proposition 7.3 ci-dessous, la condition suivante

$$\sum_{k\geqslant 0} \lambda_{\min}(R_k) = +\infty \tag{7.7}$$

est suffisante.

L'algorithme proximal à métrique variable R_k est difficile à analyser du fait de la modification de R_k à chaque itération, qui empêche d'utiliser les relations de monotonie issues de la formule (3.67), permettant de comparer deux itérations successives ou les itérés de l'itération courante à une solution du problème (un argument classique du cas où $R_k = r_k I$). Le résultat de convergence de la proposition 7.3 repose alors sur la monotonie suivante, observée au point 2 de la page 328 : l'algorithme proximal fait décroître la fonction f à chaque itération.

On note $\tilde{f}(x_k)$ la valeur optimale dans (7.3), qui est la valeur de la régularisée de Moreau-Yosida en x_k. Comme x_k minimise f si, et seulement si, $\tilde{f}(x_k) = f(x_k)$, il est naturel de s'intéresser à l'écart entre ces deux valeurs, que l'on note

$$\begin{aligned}
\delta_k &:= f(x_k) - \tilde{f}(x_k) \\
&= f(x_k) - f(x_{k+1}) - \frac{1}{2}\|x_{k+1} - x_k\|^2_{R_k^{-1}} \qquad \text{[définition de } \tilde{f}(x_k)\text{]} \\
&= f(x_k) - f(x_{k+1}) - \frac{1}{2}\langle R_k g_{k+1}, g_{k+1}\rangle \qquad [(7.4)]. \\
&\geqslant 0 \qquad [(7.6)].
\end{aligned} \tag{7.8}$$

On peut obtenir mieux que cette inégalité en utilisant l'inégalité de convexité $f(x_{k+1}) + f'(x_{k+1}; x_k - x_{k+1}) \leqslant f(x_k)$ et (7.5), à savoir

$$f(x_{k+1}) + \langle R_k g_{k+1}, g_{k+1}\rangle \leqslant f(x_k). \tag{7.9}$$

Proposition 7.3 (algorithme proximal, R_k général) *Soient $\{x_k\}$ la suite générée par l'algorithme proximal et $\{g_k\}$ la suite des sous-gradients qui interviennent dans (7.4).*

1) *La suite $\{f(x_k)\}$ décroît vers une valeur $f_* \in \mathbb{R} \cup \{-\infty\}$.*
2) *Si $f_* > -\infty$, alors $\sum_k \delta_k < +\infty$.*

On suppose désormais que (7.7) a lieu.

3) *Si $f_* > -\infty$, alors zéro est point d'adhérence de la suite $\{g_k\}$.*
4) *Si $\{x_k\}$ est bornée, alors les points d'adhérence de $\{x_k\}$ sont des minima de f (en particulier, f a un minimum).*

Démonstration. 1) Le fait que $\{f(x_k)\}$ décroisse a été observé en (7.6) ou (7.9). Comme les $f(x_k) \in \mathbb{R}$, $f(x_k)$ décroît vers une limite f_* dans $\mathbb{R} \cup \{-\infty\}$.

2) En sommant les égalités (7.8), on obtient

$$\sum_{k\geqslant 0} \delta_k + \frac{1}{2}\sum_{k\geqslant 0} \langle R_k g_{k+1}, g_{k+1}\rangle = f(x_0) - f_* < \infty.$$

Comme la seconde série est positive, on en déduit le résultat.

3) Grâce à l'identité précédente, on a aussi

$$\frac{1}{2}\sum_{k\geqslant 0}\lambda_{\min}(R_k)\|g_{k+1}\|^2 < +\infty.$$

On en déduit qu'il existe une sous-suite de $\{g_k\}$ qui tend vers zéro, sinon il existerait une constante $\gamma > 0$ tel que $\|g_k\| \geqslant \gamma$ et l'inégalité ci-dessus contredirait (7.7).

4) Si $\{x_k\}$ est bornée, alors $f_* > -\infty$, car $f(x_k) \to f_*$ et f a une minorante affine (proposition 3.7). Il suffit de montrer que f_* est la valeur minimale de f, puisque $f(x_k) \to f_*$ (point 1) et qu'alors un point d'adhérence x_* de $\{x_k\}$ sera tel que $f(x_*) = f_*$ (f est fermée), ce qui implique que x_* minimise f. Par le point 3, il existe une sous-suite d'indices $\mathcal{K}$ telle que $\{g_k\}_{k\in\mathcal{K}} \to 0$. Comme $\{x_k\}$ est bornée, il existe une sous-suite d'indices $\mathcal{K}' \subseteq \mathcal{K}$ telle que $\{x_k\}_{k\in\mathcal{K}'} \to x_*$. On peut alors passer à la limite dans $g_k \in \partial f(x_k)$ (résultat de (7.4)) pour obtenir $0 \in \partial f(x_*)$ (point (iii) de la proposition 3.67), c'est-à-dire que x_* minimise f et donc que $f_* = \inf f$. □

L'algorithme proximal a davantage de proprétés lorsque $R_k = r_k R$, où r_k est un scalaire strictement positif et R est un opérateur auto-adjoint défini positif fixé. Le cas où $R_k = r_k I$ est souvent rencontré. Dans ce cas, la condition de convergence (7.7) devient la condition sur les r_k suivante :

$$\sum_{k\geqslant 0} r_k = +\infty. \tag{7.10}$$

Comme exemple de propriété supplémentaire : la suite $\{x_k\}$ est nécessairement minimisante, même si elle n'est pas bornée (voir le point 1 ci-dessous).

Proposition 7.4 (algorithme proximal, $R_k = r_k R$) *Soient $\{x_k\}$ la suite générée par l'algorithme proximal avec $R_k = r_k R$, où r_k est un scalaire strictement positif et R est un opérateur auto-adjoint défini positif, et $\{g_k\}$ la suite des sous-gradients qui interviennent dans (7.4). On suppose que (7.10) a lieu.*

1) *La suite $\{f(x_k)\}$ décroît vers la valeur minimale de f (qui peut être $-\infty$).*
2) *Si, de plus, $\{r_k\}$ est bornée et f a un minimiseur, alors $\{x_k\}$ converge vers un minimiseur de f.*

Démonstration. 1) L'argument consiste à examiner comment $\{x_k\}$ dévie d'une suite de Fejér, la déviation étant mesurée au moyen de la fonction f. On regarde donc comment évolue l'écart $x_k - x$ en norme R^{-1}, pour un $x \in \mathbb{E}$ pour l'instant arbitraire. La récurrence $x_{k+1} = x_k - r_r R g_{k+1}$ conduit à $x_{k+1} - x = x_k - x - r_r R g_{k+1}$, puis à l'identité suivante en prenant le carré des normes R^{-1} :

$$\|x_{k+1} - x\|^2_{R^{-1}} = \|x_{k+1} - x_k\|^2_{R^{-1}} + 2\langle x_{k+1} - x_k, x_k - x\rangle_{R^{-1}} + \|x_k - x\|^2_{R^{-1}}. \tag{7.11}$$

L'idée-clé est d'estimer les deux premiers termes du membre de droite (qui font éventuellement dévier $\{x_k\}$ d'une suite de Fejér) par une variation de f. On a

$$\begin{aligned} f(x) &\geqslant f(x_{k+1}) + \langle g_{k+1}, x - x_{k+1}\rangle \qquad \text{[inégalité de convexité]} \\ &= f(x_{k+1}) + \langle R_k^{-1}(x_k - x_{k+1}), x - x_{k+1}\rangle \qquad \text{[(7.4)]} \\ &= f(x_{k+1}) + \langle R_k^{-1}(x_k - x_{k+1}), x - x_k\rangle + \langle R_k^{-1}(x_k - x_{k+1}), x_k - x_{k+1}\rangle. \end{aligned}$$

On fait ensuite apparaître δ_k donné par (7.8) :

$$f(x) \geqslant f(x_k) + \frac{1}{r_k}\,\langle x_k - x_{k+1}, x - x_k\rangle_{R^{-1}} + \frac{1}{2r_k}\,\|x_k - x_{k+1}\|_{R^{-1}}^2 - \delta_k.$$

En multipliant par $2r_k$, on obtient une majoration des deux premiers termes du membre de droite de (7.11), qui devient

$$\|x_{k+1} - x\|_{R^{-1}}^2 \leqslant \|x_k - x\|_{R^{-1}}^2 + 2r_k\Big[f(x) - f(x_k) + \delta_k\Big]. \tag{7.12}$$

Choisissons maintenant $x \in \mathbb{E}$. On sait d'après le point 1 de la propoisiton 7.3, que $f(x_k)$ décroît vers une limite f_*. Si $f_* = -\infty$, le résultat est démontré. Supposons désormais que $f_* > -\infty$. Si $f_* \neq \inf f$, on peut trouver un $x \in \mathbb{E}$ et un $\eta > 0$ tel que $f(x) + \eta \leqslant f(x_k)$ pour tout indice $k \geqslant 0$. Alors l'inégalité précédente devient

$$\forall\, k \geqslant 0 : \quad \|x_{k+1} - x\|_{R^{-1}}^2 \leqslant \|x_k - x\|_{R^{-1}}^2 + 2r_k(\delta_k - \eta).$$

Mais $\delta_k \to 0$ par le point 2 de la proposition 7.3, si bien que pour k_1 assez grand :

$$\forall\, k \geqslant k_1 : \quad \|x_{k+1} - x\|_{R^{-1}}^2 \leqslant \|x_k - x\|_{R^{-1}}^2 - \eta r_k.$$

En sommant ces inégalités de k_1 à $k_2 \geqslant k_1$, on obtient

$$0 \leqslant \|x_{k_2+1} - x\|_{R^{-1}}^2 \leqslant \|x_{k_1} - x\|_{R^{-1}}^2 - \eta \sum_{k=k_1}^{k_2} r_k,$$

si bien que la série $\sum_k r_k$ serait convergente, en contradiction avec l'hypothèse (7.10).

2) Soit $\bar{x}$ un minimiseur de f. En prenant $x = \bar{x}$ dans (7.12), on obtient

$$\forall\, k \geqslant 0 : \quad \|x_{k+1} - \bar{x}\|_{R^{-1}}^2 \leqslant \|x_k - \bar{x}\|_{R^{-1}}^2 + 2r_k\delta_k. \tag{7.13}$$

Si $\{r_k\}$ est bornée, la série $\sum_k r_k\delta_k$ converge et, en sommant les inégalités précédentes, on voit que $\{x_k\}$ est bornée.

Soit $\bar{x}$ un point d'adhérence de $\{x_k\}$, qui est nécessairement un minimiseur de f (car $f(x_k) \to \inf f$, par le point 1, et f est s.c.i.). Montrons que toute la suite $\{x_k\}$ convergence vers $\bar{x}$. Soit $\varepsilon > 0$. Il suffit de montrer que $\|x_k - \bar{x}\|_{R^{-1}}^2 \leqslant \varepsilon$ pour k assez grand. Par définition de $\bar{x}$, on peut trouver un indice k_1 tel que

$$\|x_{k_1} - \bar{x}\|_{R^{-1}}^2 \leqslant \frac{\varepsilon}{2} \qquad \text{et} \qquad 2\sum_{k \geqslant k_1} r_k\delta_k \leqslant \frac{\varepsilon}{2}.$$

En sommant les inégalités (7.13), on trouve que $\|x_k - \bar{x}\|_{R^{-1}}^2 \leqslant \varepsilon$ pour tout $k \geqslant k_1$. □

7.3 Méthode de Gauss-Seidel

La méthode de Gauss-Seidel est initialement une méthode itérative de résolution d'un système d'équations linéaires (de dimension finie) de la forme $Ax = b$ (section 7.3.1), ce qui signifie qu'elle génère une suite qui converge vers une solution de cette équation, lorsque celle-ci en a une et lorsque des conditions de convergence sont satisfaites (par exemple lorsque A est symétrique définie positive). L'algorithme suppose que la diagonale de A est formée d'éléments non nuls. Elle se décline aussi en une version « par blocs ».

Le principe de la méthode peut s'étendre à la résolution de systèmes d'équations non linéaires (section 7.3.2) et à l'optimisation (section 7.3.3), mais avec des conditions d'efficacité moins claires. En optimisation, l'utilité de cette approche dépendra beaucoup de la structure du problème. Le principe gauss-seidelien permet aussi d'interpréter d'autres algorithmes.

7.3.1 En algèbre linéaire

Version élément par élément

Rappelons d'abord ce qu'est la méthode de Gauss-Seidel pour résoudre en $x \in \mathbb{R}^n$ le système linéaire

$$Ax = b,$$

dans lequel $A \in \mathbb{R}^{n\times n}$ et $b \in \mathbb{R}^n$. Il s'agit d'un algorithme itératif, générant donc une suite $\{x_k\} \subseteq \mathbb{R}^n$. On interrompt le calcul de la suite lorsque l'itéré courant, disons x_k, est jugé suffisamment proche d'une solution, par exemple parce que la norme du *résidu* $\|Ax_k - b\|$ est petite.

Soit $x_k = ((x_k)_1, \ldots, (x_k)_n) \in \mathbb{R}^n$ l'itéré courant. L'itéré suivant $x_{k+1} = ((x_{k+1})_1, \ldots, (x_{k+1})_n) \in \mathbb{R}^n$ se calcule en n étapes, comme suit.

- *Étape* 1. Si l'on suppose que $a_{11} \neq 0$ et connaissant $((x_k)_2, \ldots, (x_k)_n)$, on peut calculer $(x_{k+1})_1$ au moyen de la première équation du système linéaire $Ax = b$. De manière plus précise, $(x_{k+1})_1 \in \mathbb{R}$ est pris comme l'unique solution de

$$a_{11}\boxed{(x_{k+1})_1} + a_{12}(x_k)_2 + \cdots + a_{1n}(x_k)_n = b_1.$$

- *Étape* 2. Si l'on suppose que $a_{22} \neq 0$ et connaissant $((x_{k+1})_1, (x_k)_3, \ldots, (x_k)_n)$, on peut calculer $(x_{k+1})_2$ au moyen de la deuxième équation du système linéaire $Ax = b$. De manière plus précise, $(x_{k+1})_2 \in \mathbb{R}$ est pris comme l'unique solution de

$$a_{21}(x_{k+1})_1 + a_{22}\boxed{(x_{k+1})_2} + a_{23}(x_k)_3 + \cdots + a_{2n}(x_k)_n = b_2.$$

- *Étape* $i \in [1\,{:}\,n]$ (cas général). Si l'on suppose que $a_{ii} \neq 0$ et connaissant $((x_{k+1})_1, \ldots, (x_{k+1})_{i-1}, (x_k)_{i+1}, \ldots, (x_k)_n)$, on peut calculer $(x_{k+1})_i$ au moyen de la i-ième équation du système linéaire $Ax = b$. De manière plus précise, $(x_{k+1})_i \in \mathbb{R}$ est pris comme l'unique solution de

$$a_{i1}(x_{k+1})_1 + \cdots + a_{i,i-1}(x_{k+1})_{i-1} + a_{ii}\boxed{(x_{k+1})_i} + a_{i,i+1}(x_k)_{i+1} + \cdots + a_{in}(x_k)_n = b_i.$$

En résumé, on calcule les composantes $(x_{k+1})_i$ de x_{k+1} de manière séquentielle pour $i = 1, \ldots, n$ par

$$\boxed{(x_{k+1})_i = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij}(x_{k+1})_j - \sum_{j=i+1}^{n} a_{ij}(x_k)_j \right).}$$

La formule fait intervenir les éléments $(x_{k+1})_j$ $(j = 1, \ldots, i-1)$ calculés dans les étapes précédentes.

L'expression matricielle de l'algorithme suppose que la matrice A se décompose comme suit

$$A = L + D + U,$$

où D est la partie diagonale de A, L sa partie triangulaire inférieure stricte et U sa partie triangulaire supérieure stricte. Une itération de la méthode de Gauss-Seidel, celle passant de x_k à x_{k+1}, consiste alors à résoudre le système triangulaire inférieur

$$\boxed{(L + D)x_{k+1} = b - Ux_k,}$$

de « haut en bas », c'est-à-dire en déterminant successivement $(x_{k+1})_1$, $(x_{k+1})_2$, …, $(x_{k+1})_n$.

Version par blocs

La méthode de Gauss-Seidel peut se décliner en une « *version par blocs* ». Celle-ci procède de manière similaire à la méthode élément par élément décrite ci-dessus, mais en remplaçant l'utilisation des éléments de A par des sous-matrices de A, appelées ici des *blocs*. On suppose que l'ensemble des indices $[1:n]$ est partitionné en p sous-intervalles (non vides et deux-à-deux disjoints) :

$$[1:n] = I_1 \cup I_2 \cup \cdots \cup I_p.$$

La matrice A et le vecteur b sont alors décomposés comme suit

$$A = \begin{pmatrix} A_{I_1 I_1} & A_{I_1 I_2} & \cdots & A_{I_1 I_p} \\ A_{I_2 I_1} & A_{I_2 I_2} & \cdots & A_{I_2 I_p} \\ \vdots & \vdots & \ddots & \vdots \\ A_{I_p I_1} & A_{I_p I_2} & \cdots & A_{I_p I_p} \end{pmatrix} \qquad \text{et} \qquad b = \begin{pmatrix} b_{I_1} \\ b_{I_2} \\ \vdots \\ b_{I_p} \end{pmatrix},$$

où A_{IJ} est la sous-matrice de A obtenue en sélectionnant les éléments avec indices de ligne dans I et indices de colonnes dans J, tandis que b_I est le sous-vecteur de b obtenu en sélectionnant les éléments avec indices dans I.

La méthode de Gauss-Seidel par blocs suppose que les sous-matrices principales $A_{I_i I_i}$, avec $i \in [1:p]$, sont inversibles.

Une itération de la méthode de Gauss-Seidel par blocs, celle passant de x_k à x_{k+1}, s'écrit de la même manière que la méthode élément par élément, à savoir

$$\boxed{(L + D)x_{k+1} = b - Ux_k,}$$

mais avec des définitions différentes de L, D et U :

$$L = \begin{pmatrix} 0 & \cdots & \cdots & 0 \\ A_{I_2 I_1} & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ A_{I_p I_1} & \cdots & A_{I_p I_{p-1}} & 0 \end{pmatrix}, \qquad D = \begin{pmatrix} A_{I_1 I_1} & 0 & \cdots & 0 \\ 0 & A_{I_2 I_2} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & A_{I_p I_p} \end{pmatrix}$$

et $U = A - L - D$. La résolution du système triangulaire par blocs ci-dessus, se fait également de « haut en bas », c'est-à-dire en déterminant successivement $(x_{k+1})_{I_1}$, $(x_{k+1})_{I_2}$, …, $(x_{k+1})_{I_p}$.

7.3.2 Pour les systèmes non linéaires

Le principe de la méthode de Gauss-Seidel peut également s'appliquer à la résolution d'un système d'équations non linéaires $F(x) = 0$, où $F : \mathbb{R}^n \to \mathbb{R}^n$. Ce système s'écrit donc sous la forme de n équations non linéaires à n inconnues :

$$\begin{cases} F_1(x_1, x_2, \ldots, x_n) = 0 \\ F_2(x_1, x_2, \ldots, x_n) = 0 \\ \cdots \\ F_n(x_1, x_2, \ldots, x_n) = 0. \end{cases}$$

La *méthode de Gauss-Seidel* résout ce système de manière itérative, en générant donc une suite $\{x_k\} \subseteq \mathbb{R}^n$. On interrompt le calcul de la suite lorsque l'itéré courant, disons x_k, est jugé suffisamment proche d'une solution, par exemple parce que la norme du *résidu* $\|F(x_k)\|$ est petite.

Soit $x_k = ((x_k)_1, \ldots, (x_k)_n) \in \mathbb{R}^n$ l'itéré courant. L'itéré suivant $x_{k+1} = ((x_{k+1})_1, \ldots, (x_{k+1})_n) \in \mathbb{R}^n$ se calcule en n étapes, comme suit.

- *Étape* 1. Connaissant $((x_k)_2, \ldots, (x_k)_n)$, on calcule $(x_{k+1})_1$ comme solution de l'équation non linéaire (cette solution est supposée exister) :

$$F_1(\boxed{(x_{k+1})_1}, (x_k)_2, \ldots, (x_k)_n) = 0.$$

- *Étape* 2. Connaissant $((x_{k+1})_1, (x_k)_3, \ldots, (x_k)_n)$, on calcule $(x_{k+1})_2$ comme solution de l'équation non linéaire (cette solution est supposée exister) :

$$F_1((x_{k+1})_1, \boxed{(x_{k+1})_2}, (x_k)_2, \ldots, (x_k)_n) = 0.$$

- *Étape* $i \in [1:n]$ (cas général). Connaissant $((x_{k+1})_1, \ldots, (x_{k+1})_{i-1}, (x_k)_{i+1}, \ldots, (x_k)_n)$, on calcule $(x_{k+1})_i$ comme solution de l'équation non linéaire (cette solution est supposée exister) :

$$F_i((x_{k+1})_1, \ldots, (x_{k+1})_{i-1}, \boxed{(x_{k+1})_i}, (x_k)_{i+1}, \ldots, (x_k)_n) = 0.$$

La version « par blocs » se définit facilement en considérant des groupes d'équations et d'inconnues, au lieu de considérer, comme ci-dessus, équation et inconnue une par une.

7.3.3 En optimisation

Le principe de la méthode de Gauss-Seidel décrit dans la section précédente s'applique naturellement au problème d'optimisation non linéaire

$$\inf_{x\in X} f(x), \tag{7.14}$$

dans lequel on minimise une fonction $f : \mathbb{R}^n \to \mathbb{R}$ sur un sous-ensemble X de $\mathbb{R}^n$. Nous présentons directement ci-dessous la version « par blocs », qui est la plus utile lorsque le nombre p de blocs est faible (souvent $p = 2$). La méthode de Gauss-Seidel perd en effet de sa pertinence lorsque p est grand, par manque d'efficacité dans ce cas. la version « élément par élément » peut être vue comme un cas particulier de la version par blocs, obtenue en prenant n blocs de cardinal 1.

On suppose donc que l'ensemble des indices $[1:n]$ est *partitionné* en p blocs,

$$[1:n] = I_1 \cup I_2 \cup \cdots \cup I_p, \tag{7.15}$$

et que l'ensemble admissible est un produit cartésien de p ensembles,

$$X = X_1 \times X_2 \times \cdots \times X_p, \tag{7.16}$$

où chaque X_i est un convexe de $\mathbb{R}^{|I_i|}$. La variable $x \in \mathbb{R}^n$ se décomposera comme suit

$$x = (x_{I_1}, x_{I_2}, \ldots, x_{I_p}).$$

Lorsque f est différentiable et que $X = \mathbb{R}^n$, on pourrait obtenir une méthode de Gauss-Seidel en appliquant la méthode de la section 7.3.2 à la condition d'optimalité du premier ordre de ce problème d'optimisation sans contrainte, à savoir

$$\nabla f(x) = 0,$$

qui est un système de n équations non linéaires à n inconnues $x = (x_1, \ldots, x_n)$. Mais on peut préférer, comme ci-dessous, rester dans le domaine de l'optimisation en minimisant f séquentiellement, bloc par bloc. Cette option a l'avantage de pouvoir prendre en compte des contraintes, c'est-à-dire de restreindre les variables à l'ensemble admissible X.

La *méthode de Gauss-Seidel*[1] résout le problème d'optimisation (7.14) de manière itérative, en générant donc une suite $\{x_k\} \subseteq \mathbb{R}^n$. L'algorithme passe d'un itéré au suivant en minimisant f un bloc de variables à la fois, en séquence. On interrompt le calcul de la suite lorsque l'itéré courant, disons x_k, est jugé suffisamment proche d'une solution, par exemple parce que la norme du *gradient projeté* $\|g^{\mathrm{P}}(x_k)\|$ est jugée suffisamment petite (on rappelle que le gradient projeté $g^{\mathrm{P}}(x_k)$ est la projection orthogonale de $\nabla f(x_k)$ sur l'opposé du cône tangent $\mathrm{T}_{x_k} X$ et que celui-ci est nul en une solution de (7.14)).

[1] Cette méthode est appelée *méthode de relaxation* par Glowinski, Lions, Trémolières [252 ; 1976, page 60-68], mais cette appellation est utilisée pour beaucoup trop d'algorithmes pour qu'elle soit suffisamment discriminante.

Algorithme 7.5 (de Gauss-Seidel par bloc en optimisation) Une itération passe de l'itéré courant $x_k \in X$ à l'itéré suivant $x_{k+1} \in X$ en p étapes successives, indicées par $i = 1, \dots, p$:

$$(x_{k+1})_{I_i} \in \operatorname*{arg\,min}_{x_{I_i} \in X_i} f((x_{k+1})_{I_1}, \dots, (x_{k+1})_{I_{i-1}}, x_{I_i}, (x_k)_{I_{i+1}}, \dots, (x_k)_{I_p}). \tag{7.17}$$

La version « élément par élément » se définit facilement en considérant des blocs I_i de cardinal 1 et en minimisant f composante par composante. Ce dernier algorithme porte aussi le nom de *méthode de descente par coordonnée*.

Le résultat suivant montre la convergence de la méthode de Gauss-Seidel lorsque f est de classe C^1, coercive et strictement convexe. Il repose sur la caractérisation de la stricte convexité, donnée à la proposition 3.23.

Proposition 7.6 (convergence de Gauss-Seidel en optimisation) *Si, pour chaque $i \in [1:p]$, X_i est un convexe fermé non vide de $\mathbb{R}^{|I_i|}$ et si f est coercive sur X, strictement convexe sur X et de classe C^1 dans un voisinage de X, alors*
1) *le problème (7.14)-(7.16) a une unique solution $\bar{x}$,*
2) *l'algorithme 13.21 est bien défini et, quel que soit l'itéré initial $x_1 \in X$, il génère une suite $\{x_k\} \subseteq X$ qui converge vers $\bar{x}$.*

Démonstration. 1) L'existence de solution se déduit de la continuité, de la coercivité de f et du fait que l'ensemble admissible est non vide et fermé (proposition 1.4). L'unicité se déduit de la stricte convexité de f et de la convexité de X (proposition 3.5).

2) *Préliminaires.* Pour les mêmes raisons qu'au point 1, tous les *problèmes intermédiaires* (7.17) ont une solution unique, si bien que l'algorithme 13.21 est bien défini. Ces solutions uniques ou *itérés intermédiaires* sont notés comme suit : $x_{k+1,0} := x_k$ et

$$x_{k+1,i} := ((x_{k+1})_{I_1}, \dots, (x_{k+1})_{I_i}, (x_k)_{I_{i+1}}, \dots, (x_k)_{I_p}), \quad \text{pour } i \in [1:p].$$

Donc $x_{k+1,i}$ est la solution du problème intermédiaire (7.17) et $x_{k+1} = x_{k+1,p}$. Par construction

$$f(x_{k+1}) \leqslant f(x_{k+1,p-1}) \leqslant \cdots \leqslant f(x_{k+1,1}) \leqslant f(x_k).$$

Comme f est bornée inférieurement par $f(\bar{x})$, ces inégalités impliquent que

$$\{f(x_k)\} \text{ et les } \{f(x_{k,i})\} \text{ convergent vers la même valeur.} \tag{7.18}$$

Par ailleurs, l'optimalité du problème intermédiaire (7.17) implique que

$$\forall\, x_{I_i} \in X_i : \quad \nabla_{x_i} f(x_{k+1,i})^{\mathsf{T}} (x_{I_i} - (x_{k+1})_{I_i}) \geqslant 0. \tag{7.19}$$

Montrons que $x_{k+1} - x_k \to 0$. La coercivité de f implique la bornitude de ces ensembles de sous-niveau (point (ii) de l'exercice 1.3). Il existe donc un réel $\beta > 0$ tel que

$$\{x : \mathbb{R}^n : f(x) \leqslant f(x_1)\} \subseteq \beta \bar{B},$$

où $\bar{B}$ est la boule unité fermée de $\mathbb{R}^n$, si bien que tous les itérés (intermédiaires ou pas) sont dans $\beta\bar{B}$. Alors, la stricte convexité donne (point (ii) de la proposition 3.23) :

$$\begin{aligned} & f(x_{k+1,i-1}) - f(x_{k+1,i}) \\ & \quad \geqslant \quad f'(x_{k+1,i}) \cdot (x_{k+1,i-1} - x_{k+1,i}) + \tfrac{1}{2} g_\beta(\tfrac{1}{2}\|x_{k+1,i-1} - x_{k+1,i}\|) \\ & \quad = \quad \underbrace{\nabla_{x_{I_i}} f(x_{k+1,i})^{\mathsf{T}}((x_k)_{I_i} - (x_{k+1})_{I_i})}_{\geqslant 0 \text{ par } (7.19)} + \tfrac{1}{2} g_\beta(\tfrac{1}{2}\|x_{k+1,i-1} - x_{k+1,i}\|) \\ & \quad \geqslant \quad \tfrac{1}{2} g_\beta(\tfrac{1}{2}\|x_{k+1,i-1} - x_{k+1,i}\|), \end{aligned}$$

où $\|\cdot\|$ désigne la norme euclidienne. En sommant :

$$f(x_k) - f(x_{k+1}) \geqslant \textstyle\sum_{i=1}^{p} \tfrac{1}{2} g_\beta(\tfrac{1}{2}\|x_{k+1,i-1} - x_{k+1,i}\|).$$

Alors (7.18) et la positivité de g_β impliquent que

$$\forall\, i \in [1:p] : \qquad g_\beta(\tfrac{1}{2}\|x_{k+1,i-1} - x_{k+1,i}\|) \to 0.$$

On utilise alors le fait que $g_\beta(0) = 0$, que $g_\beta(t) > 0$ pour $t > 0$ et la continuité de g_β pour en déduire que $\frac{1}{2}\|x_{k+1,i-1} - x_{k+1,i}\| \to 0$ lorsque $k \to \infty$, pour tout $i \in [1:p]$. Comme p est fini :

$$x_{k+1} - x_k \to 0. \tag{7.20}$$

Montrons que $x_k \to \bar{x}$, ce qui conclura la démonstration. On utilise cette fois le point (iii) de la proposition 3.23 entre x_{k+1} et la solution $\bar{x}$:

$$(f'(x_{k+1}) - f'(\bar{x})) \cdot (x_{k+1} - \bar{x}) \geqslant g_\beta(\|x_{k+1} - \bar{x}\|).$$

Mais $f'(\bar{x}) \cdot (x_{k+1} - \bar{x}) \geqslant 0$ par optimalité de $\bar{x}$ et $x_{k+1} \in X$, si bien qu'en inversant l'inégalité, on a

$$\begin{aligned} g_\beta(\|x_{k+1} - \bar{x}\|) \quad & \leqslant \quad f'(x_{k+1}) \cdot (x_{k+1} - \bar{x}) \\ & = \quad \sum_{i=1}^{p} \nabla_{x_{I_i}} f(x_{k+1})^{\mathsf{T}} (x_{k+1} - \bar{x})_{I_i} \\ & \leqslant \quad \sum_{i=1}^{p} \left(\nabla_{x_{I_i}} f(x_{k+1}) - \nabla_{x_{I_i}} f(x_{k+1,i})\right)^{\mathsf{T}} (x_{k+1} - \bar{x})_{I_i} \\ & \qquad\qquad [\text{par } (7.19) \text{ avec } x_{I_i} = \bar{x}_{I_i}] \\ & \leqslant \quad \sum_{i=1}^{p} \|\nabla_{x_{I_i}} f(x_{k+1}) - \nabla_{x_{I_i}} f(x_{k+1,i})\| \, \|x_{k+1} - \bar{x}\|, \end{aligned}$$

par l'inégalité de Cauchy-Schwarz et $\|(x_{k+1} - \bar{x})_{I_i}\| \leqslant \|x_{k+1} - \bar{x}\|$. Cela permet de conclure. En effet, $\|x_{k+1} - \bar{x}\|$ est borné par 2β. Par ailleurs, $\|x_{k+1} - x_{k+1,i}\| \leqslant \|x_{k+1} - x_k\| \to 0$ par (7.20). Alors la continuité de f', donc de $\nabla_{x_{I_i}} f$, sur le compact $\beta\bar{B}$ se traduit en une continuité uniforme, si bien que, pour tout $i \in [1:p]$, $\nabla_{x_{I_i}} f(x_{k+1}) - \nabla_{x_{I_i}} f(x_{k+1,i}) \to 0$. On en déduit que $g_\beta(\|x_{k+1} - \bar{x}\|) \to 0$ et donc que $x_k \to \bar{x}$ par le même raisonnement que pour obtenir (7.20). □

Remarques 7.7 1) Si l'on applique la proposition 7.6 au cas où $X = \mathbb{R}^n$ et f est la fonction quadratique $x \mapsto \frac{1}{2}x^\mathsf{T}Ax - b^\mathsf{T}x$, on retrouve le résultat affirmant que la méthode de Gauss-Seidel par blocs pour résoudre le système linéaire $Ax = b$ convergence, quels que soient le vecteur b et le point initial, pourvu que A soit définie positive.

2) La méthode de Gauss-Seidel est un algorithme lent (il requiert beaucoup d'itérations), dont la mise en œuvre est coûteuse (chaque itération peut demander beaucoup de temps de calcul, selon les cas). Tel qu'il est présenté, il requiert en effet la minimisation *exacte* de f dans chaque problème intermédiaire et ces p minimisations doivent être réalisées à chaque itération. Son application est donc restreinte au cas où le nombre de blocs est petit.

3) L'algoithme 13.21 ne s'étend pas aisément à des ensembles admissibles plus complexes qu'un produit cartésien d'ensembles convexes. Par exemple si l'on cherche à minimiser composante par composante la fonction linéaire $f : \mathbb{R}^2 \to \mathbb{R} : (x_1, x_2) \mapsto x_1 + x_2$ sur l'ensemble $X := \{x \in \mathbb{R}^2_+ : x_1 x_2 \geqslant 1\}$, qui n'est pas le produit cartésien de deux intervalles, tout point de la frontière de X est bloquant (c'est-à-dire que l'algorithme ne peut y progresser), alors que seul le point $\bar{x} = (1, 1)$ est solution [252] ; voir la figure 7.1.

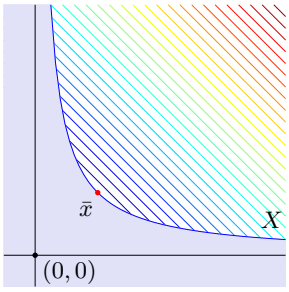

Fig. 7.1. Exemple de problème pour lequel l'algoithme 13.21, de Gauss-Seidel, reste bloqué en tout point de la frontière du domaine admissible (les droites sont les « courbes » de niveau de la fonction objectif dans l'ensemble admissible X).

4) En l'absence de convexité, la méthode de Gauss-Seidel ne converge pas nécessairement, même pour des fonctions de classe C^∞. Powell [488 ; 1973] a en effet construit plusieurs fonctions conduisant à la non-convergence de la méthode de Gauss-Seidel composante par composante, notamment une fonction C^∞ de trois variables pour laquelle les itérés générés ont un cycle limite formé de 6 points en lesquels le gradient n'est pas nul. On peut toutefois avoir convergence si les coordonnées ne sont pas choisies cycliquement [481 ; 1971].

5) D'autres résultats de convergence sont donnés par Luo et Tseng [402 ; 1992]. □

Notes

La convergence de l'algorithme de Gauss-Seidel pour les systèmes linéaires (section 7.3.1) est présentée, par exemple, par Varga [605] et pour les systèmes non linéaires (section 7.3.2) par Ortega et Rockoff [464].

Le résultat de convergence globale de la méthode de Gauss-Seidel en optimisation, la proposition 7.6, est adaptée de [252 ; théorème 1.2, page 66]. Pour d'autres résultats, voir [293] et ses références.

Exercices

7.1. *Terminaison finie de l'algorithme proximal.* Soient $f \in \mathrm{C}\overline{\mathrm{onv}}(\mathbb{E})$ et $\tilde{f}$ sa régularisée de Moreau-Yosida (section 3.7.2). On suppose que $\bar{x} \in (\operatorname{dom} f)^{\circ}$ minimise f et que $0 \in \operatorname{int} \partial f(\bar{x})$ (de manière plus précise : il existe $\varepsilon > 0$ tel que $\bar{B}(0,\varepsilon) \subseteq \partial f(\bar{x})$). On note x_p le point proximal de $x \in \mathbb{E}$. Montrez que

(*i*) $g \in \partial f(x)$ et $\|g\| < \varepsilon \implies x = \bar{x}$,
(*ii*) $\|x - \bar{x}\| \leqslant \varepsilon \implies x_p = \bar{x}$,
(*iii*) $\tilde{f}$ est quadratique sur $\bar{B}(\bar{x},\varepsilon)$.

8 Optimisation quadratique

On s'intéresse dans ce chapitre à la résolution numérique des systèmes d'équations linéaires par des méthodes *itératives finies* ; nous verrons plus loin ce que l'on entend par ces deux qualificatifs. Pour certaines méthodes, on considérera leur extension à la résolution de systèmes d'équations non linéaires. Pour d'autres, cette extension se fera dans d'autres chapitres.

On cherche donc un point $x_* \in \mathbb{R}^n$ solution du système linéaire en x suivant

$$Ax = b, \tag{8.1}$$

où A est une matrice d'ordre n donnée et b est un vecteur donné dans $\mathbb{R}^n$. Nous supposerons toujours que A est inversible. Dans ce cas, la solution unique de (8.1) s'écrit

$$x_* = A^{-1}b,$$

où A^{-1} est la matrice inverse de A. Pour obtenir x_*, on ne calcule jamais A^{-1}. Cela n'est ni nécessaire, ni opportun dès que n dépasse quelques unités (le temps de calcul requis est alors trop important), ni stable numériquement [313 ; 2002, section 14.2]. Rappelons que la résolution d'un système linéaire d'ordre n non structuré par factorisation de la matrice A demande de l'ordre de $O(n^3)$ opérations. L'algorithme le plus simple et le plus utilisé pour ce faire est l'*élimination gaussienne*, équivalente à la *factorisation gaussienne* de la matrice A, avec pivotage partiel. Cet algorithme requiert $2n^3/3 + O(n^2)$ opérations flottantes. Les algorithmes présentés dans cette section ont aussi une complexité opérationnelle en $O(n^3)$. Nous distinguerons deux cas : celui où A est symétrique définie positive et celui où A est non symétrique.

Si la matrice A est symétrique, la solution du système (8.1) est aussi le point stationnaire de la fonction quadratique

$$f(x) = \frac{1}{2}x^\mathsf{T}Ax - b^\mathsf{T}x. \tag{8.2}$$

Si, de plus, A est définie positive, x_* réalise le minimum de f. On peut dans ce cas obtenir des algorithmes de résolution du système (8.1) à partir d'algorithmes de minimisation de f. L'*algorithme du gradient conjugué* est de ce type (section 8.2).

Si A n'est pas symétrique, on peut remplacer la résolution du système (8.1) par la résolution de l'*équation normale* équivalente

$$A^\mathsf{T}Ax = A^\mathsf{T}b. \tag{8.3}$$

Comme la matrice de ce système est symétrique définie positive, on peut songer à utiliser les techniques mentionnées ci-dessus. Cependant, le système (8.3) peut être

beaucoup moins bien conditionné que le système original (8.1), si bien qu'il est souvent préférable d'utiliser des méthodes s'attaquant directement au système (8.1), sans faire appel à des algorithmes de minimisation de fonction. L'idée est alors de générer des approximations de x_* qui sont, dans un sens à préciser, les meilleures approximations sur des sous-espaces affines de dimension croissante. On peut en effet espérer que si l'on a une « bonne » approximation de x_* sur le sous-espace affine x_1+K_p, K_p étant un sous-espace vectoriel de dimension p, il ne sera pas trop difficile d'obtenir une « bonne » approximation sur un sous-espace affine $x_1 + K_{p+1}$, de dimension $p + 1$, contenant $x_1 + K_p$. Dans l'*algorithme du résidu minimal* (section 8.3), les sous-espaces K_p sont obtenus par un procédé particulier : ce sont des *sous-espaces de Krylov* (section 8.1). D'ailleurs, nous verrons que l'algorithme du gradient conjugué génère des itérés qui sont aussi dans un tel sous-espace de Krylov, si bien que ces deux algorithmes font partie de la même famille, celle dite des *méthodes de Krylov*.

Les méthodes de Krylov ont la propriété de trouver la solution du système (8.1) en un nombre fini d'étapes (en artihmétique exacte). Elles s'apparentent sur ce point aux *méthodes directes* de résolution, fondées sur la factorisation de la matrice A : factorisation gaussienne, factorisation QR... En pratique cependant, la présence d'erreurs d'arrondi dans les calculs empêche les méthodes de Krylov de trouver la solution en un nombre fini d'étapes, dès que la dimension du système linéaire dépasse quelques dizaines, si bien qu'on les range souvent dans la famille des méthodes itératives. Ces dernières sont intéressantes pour plusieurs raisons. D'abord, comme elles procèdent par améliorations successives, on peut s'arrêter lorsque la précision est jugée satisfaisante. Ceci peut se produire bien avant que la solution ne soit trouvée. Une solution « acceptable » peut donc être obtenue à un faible coût. Ensuite, elles permettent de tirer parti d'une bonne approximation initiale, ce qui est souvent le cas lorsqu'on doit résoudre une succession d'équations linéaires provenant de la linéarisation d'une même équation non linéaire. Elles ont aussi des inconvénients, par exemple, de ne pouvoir exploiter la « creusité » (caractère creux) éventuelle de A que par des produits matrice-vecteur plus rapides et d'avoir, en pratique, besoin d'un bon préconditionneur pour converger en un nombre raisonnable d'itérations.

Notons pour terminer qu'il existe aussi des méthodes itératives ne trouvant pas la solution en un nombre fini d'itérations : méthodes de Jacobi, de Gauss-Seidel, de relaxation... Les premières ont été introduites au XIX^e siècle pour calculer plus rapidement (à la main, bien sûr) des solutions approchées, alors que la résolution par élimination directe était trop fastidieuse (voir la lettre de Gauss en épigraphe de ce chapitre). Elles sont cependant généralement considérées comme moins efficaces que les méthodes de Krylov [601]. Pour une introduction à ces méthodes, on pourra consulter les livres de Varga [605 ; 1962], de Ciarlet [126 ; 1982] et de Hackbusch [298 ; 1994].

Connaissances supposées. L'introduction de l'algorithme du gradient conjugué se fait en utilisant le procédé d'orthogonalisation de Gram-Schmidt (section B.1.1).

8.1 Sous-espaces de Krylov

On appelle *sous-espace de Krylov* d'ordre p, associé à une matrice A d'ordre n et à un vecteur $r \in \mathbb{R}^n$, le sous-espace vectoriel de $\mathbb{R}^n$ engnedré par les vecteurs r, Ar,

A^2r, …, $A^{p-1}r$. On le note

$$K_p \equiv K_p(A,r) \equiv \text{vect}\{r, Ar, \ldots, A^{p-1}r\}.$$

La proposition suivante montre que la dimension de K_p vaut p, tant que p reste inférieur ou égal à un certain indice s, à partir duquel sa dimension ne bouge plus (et vaut s). On peut avoir $s < n$. On dit que le sous-espace de Krylov K_s est *saturé* et que $s = s(A,r)$ est l'*indice de saturation* des sous-espaces de Krylov associés à A et à r. La proposition montre aussi que l'indice de saturation est atteint dès que $A^{-1}r$ est « capturé » par les sous-espaces de Krylov.

Proposition 8.1 *Si A est inversible, alors il existe un indice $s \geqslant 1$ tel que*

$$K_1 \subsetneqq \cdots \subsetneqq K_s = K_p, \quad \forall p \geqslant s.$$

L'indice s est caractérisé par le fait que

$$A^{-1}r \in K_s \setminus K_{s-1}.$$

Démonstration. Il suffit de montrer que

$$A^{-1}r \in K_p \iff K_p = K_{p+1}.$$

Le cas où $r = 0$ est trivial. On peut donc supposer que $r \neq 0$.

Si $A^{-1}r \in K_p$, alors il existe des réels α_i, non tous nuls (car $r \neq 0$) tels que

$$A^{-1}r = \sum_{i=0}^{j-1} \alpha_i A^i r, \qquad 1 \leqslant j \leqslant p, \qquad \alpha_{j-1} \neq 0.$$

En appliquant A^{p-j+1}, on en déduit

$$\begin{aligned} A^{p-j}r &= \sum_{i=0}^{j-1} \alpha_i A^{p-j+i+1} r, \\ A^p r &= \frac{1}{\alpha_{j-1}} \left(A^{p-j}r - \sum_{i=0}^{j-2} \alpha_i A^{p-j+i+1} r \right). \end{aligned}$$

Ceci montre que $K_{p+1} = K_p$.

Inversement, si $K_p = K_{p+1}$, on a $A^p r \in K_p$. Il existe donc des réels β_i, non tous nuls, tels que

$$A^p r = \sum_{i=k}^{p-1} \beta_i A^i r, \qquad 0 \leqslant k \leqslant p-1, \qquad \beta_k \neq 0.$$

En appliquant A^{-k-1}, on en déduit

$$A^{-1}r = \frac{1}{\beta_k} \left(- \sum_{i=k+1}^{p-1} \beta_i A^{i-k-1} r + A^{p-k-1} r \right).$$

Ceci montre que $A^{-1}r \in K_{p-k} \subseteq K_p$. □

L'indice s de saturation des sous-espaces de Krylov $K_p(A, r)$ dépend du vecteur r. Par exemple, une conséquence immédiate de la proposition 8.1 est que l'on a $s = 1$ si, et seulement si, r est un vecteur propre de A. La proposition suivante donne une borne supérieure pour $s(A, r)$ ne dépendant que de la matrice A. Nous aurons besoin de la notion suivante (voir [414 ; 1973] par exemple).

On dit que le polynôme

$$p(\xi) = a_0 + a_1\xi + \cdots + a_d\xi^d,$$

de degré d en ξ et à coefficients dans $\mathbb{R}$ *annihile* la matrice A si

$$p(A) \equiv a_0 + a_1A + \cdots + a_dA^d$$

est la matrice nulle. On dit que p est un *polynôme minimal annihilant* A si $p \neq 0$ et s'il n'y a pas de polynôme non nul annihilant A de degré strictement inférieur à celui de p. On dit enfin qu'un polynôme est *unitaire* si le coefficient du terme de degré le plus élevé vaut 1.

On sait qu'il existe un unique polynôme minimal unitaire annihilant A. Il s'écrit

$$\check{p}(\xi) = \prod_{i=1}^{t}(\xi - \lambda_i)^{\beta_i},$$

où $\lambda_1, \ldots, \lambda_t$ sont toutes les valeurs propres distinctes de A et $1 \leqslant \beta_i \leqslant \alpha_i$, α_i étant la multiplicité algébrique de λ_i (sa multiplicité comme racine du polynôme caractéristique de A). La valeur des β_i peut être obtenue en calculant la forme normale de Jordan de A (voir plus loin).

Proposition 8.2 *Si A est inversible, alors pour tout $r \in \mathbb{R}^n$*

$$s(A, r) \leqslant \beta,$$

où β est de degré du polynôme minimal unitaire annihilant A. Cette majoration est optimale : on peut trouver un vecteur r pour lequel on a $s(A, r) = \beta$.

Démonstration. Le polynôme minimal unitaire annihilant A s'écrit

$$\check{p}(\xi) = \prod_{i=1}^{t}(\xi - \lambda_i)^{\beta_i} = a_0 + a_1\xi + \cdots + a_\beta\xi^\beta,$$

où $\beta = \sum_{i=1}^{t} \beta_i$ et $a_\beta = 1$. Le coefficient $a_0 = \prod_{i=1}^{t}(-\lambda_i)^{\beta_i}$ est non nul car, étant inversible, A n'a pas de valeur propre nulle. Comme $\check{p}$ annihile A, on a

$$a_0 + a_1A + \cdots + a_\beta A^\beta = 0.$$

En appliquant A^{-1}, on a quel que soit $r \in \mathbb{R}^n$:

$$A^{-1}r \in \text{vect}\{r, Ar, \ldots, A^{\beta-1}r\} = K_\beta(A, r).$$

D'après la proposition 8.1, cela implique que $s(A, r) \leqslant \beta$.

Enfin, tout polynôme $\tilde{p}$ de degré $\beta - 1$ dont le terme de degré zéro est non nul n'annihile pas A. Il existe donc un vecteur $r \in \mathbb{R}^n$ tel que $\tilde{p}(A)r \neq 0$. Ceci implique que $A^{-1}r \notin K_{\beta-1}$ et par conséquent l'indice de saturation $s(A, r) = \beta$. □

Comme le polynôme caractéristique de A annihile A (théorème de Cayley-Hamilton) et est de degré n, on a nécessairement

$$s \leqslant n,$$

ce que l'on savait déjà. Dans le cas général, le degré

$$\beta = \sum_{i=1}^{t} \beta_i \tag{8.4}$$

du polynôme minimal annihilant A peut être calculé à partir de la forme normale de Jordan de A : il existe une matrice inversible V telle que

$$V^{-1}AV = \text{diag}(J_1^1, \ldots, J_1^{\gamma_1}, J_2^1, \ldots, J_2^{\gamma_2}, \ldots, J_t^1, \ldots, J_t^{\gamma_t}),$$

où chaque J_i^j est un bloc de Jordan de valeur propre λ_i, c'est-à-dire ayant la forme suivante (les éléments en dehors des deux diagonales indiquées sont nuls) :

$$J_i^j = \begin{pmatrix} \lambda_i & 1 & & \\ & \lambda_i & \ddots & \\ & & \ddots & 1 \\ & & & \lambda_i \end{pmatrix}.$$

La forme normale de Jordan est unique à une permutation des blocs près. Alors, β_i est l'ordre le plus élevé des blocs de Jordan de valeur propre λ_i :

$$\beta_i = \max_{i \leqslant j \leqslant \gamma_i} \text{ordre}(J_i^j).$$

On en déduit le corollaire suivant :

Corollaire 8.3 *Si A est inversible et non défective, alors pour tout $r \in \mathbb{R}^n$, $s(A, r) \leqslant t$, où t est le nombre de valeurs propres distinctes de A.*

Démonstration. En effet, A est non défective si (par définition) elle est diagonalisable par similitude : il existe une matrice V d'ordre n, inversible telle que

$$V^{-1}AV = \Lambda,$$

où Λ est diagonale. Dans ce cas, les blocs de Jordan sont des matrices d'ordre 1 et $\beta_i = 1$, $\forall i$. On en déduit le résultat en utilisant (8.4) et le théorème. □

Exemples 8.4 La matrice

$$A_1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 2 \end{pmatrix}$$

est diagonale et a deux valeurs propres distinctes. Donc $s(A_1, r) \leqslant 2$, $\forall r \in \mathbb{R}^3$.

La matrice

$$A_2 = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 2 \end{pmatrix}$$

est défective et sous forme normale de Jordan. A la valeur propre 1 correspond un bloc de Jordan d'ordre 2. Donc $s(A_2, r) \leqslant 3$, $\forall r \in \mathbb{R}^3$. En prenant le vecteur

$$r_2 = \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}$$

on peut voir que $s(A_2, r_2) = 3$. □

Les deux algorithmes itératifs de résolution du système linéaire (8.1) que nous décrivons dans ce chapitre, l'algorithme du gradient conjugué (section 8.2) et l'algorithme GMRES (section 8.3), sont des méthodes de Krylov, ce qui veut dire que l'itété x_k appartient à un certain sous-espace de Krylov $K_k(A, r)$. Dans les deux cas, le vecteur r est le résidu $r_1 = b - Ax_1$ évalué en l'itéré initial x_1. La raison pour laquelle ce choix de r convient est expliqué au début de la section 8.3.1. L'algorithme du gradient conjugué est adapté aux systèmes linéaires dans lesquels la matrice A est symétrique définie positive (ou semi-définie positive), alors que l'algorithme GMRES est plus général (et plus difficile à mettre en œuvre) puisqu'il s'affranchit de toute hypothèse sur A.

8.2 Algorithme du gradient conjugué

Dans cette section, on considère le cas où A est symétrique définie positive. On rappelle que résoudre le système linéaire (8.1) revient alors à minimiser la fonction

$$f(x) = \frac{1}{2} x^\mathsf{T} A x - b^\mathsf{T} x$$

sur $\mathbb{R}^n$.

L'*algorithme du gradient conjugué* est une méthode à direction de descente sur f. Les itérés sont donc générés par

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

où d_k est une direction de descente de f (elle vérifie $\nabla f(x_k)^\mathsf{T} d_k < 0$) et α_k est un pas positif. On notera

$$E_k = \text{vect}\{d_1, d_2, \ldots, d_k\}$$

($k \geqslant 1$) le sous-espace vectoriel engendré par les k premières directions de descente.