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.

8.2.1 Notion de directions conjuguées

Pour une matrice A symétrique définie positive donnée, la notion de conjugaison équivaut à la notion d'orthogonalité pour le produit scalaire associé à A. De façon plus précise, on a les définitions suivantes.

On dit que u et $v \in \mathbb{R}^n$ sont *conjuguées* si $u^\mathsf{T} A v = 0$. Plus généralement, on dit que $u_1, \dots u_p \in \mathbb{R}^n$ sont conjuguées si $u_i^\mathsf{T} A u_j = 0$, $\forall i \neq j$. Enfin, on dit que u est conjuguée par rapport à un sous-espace vectoriel $E \subseteq \mathbb{R}^n$ si $u^\mathsf{T} A v = 0$, $\forall v \in E$.

Si $v_1, \dots, v_p$ sont des vecteurs linéairement indépendants dans $\mathbb{R}^n$, on pourra leur associer p directions conjuguées $u_1, \dots, u_p$ en utilisant le procédé du Gram-Schmidt (section B.1.1) avec le produit scalaire $\langle u, v \rangle = u^\mathsf{T} A v$, dans lequel on se passe des étapes de normalisation superflues (étapes 1 et 2.3). Il suffit de prendre, pour $k = 1, \dots, p$,

$$u_k = v_k - \sum_{i=1}^{k-1} \beta_{ki} u_i \tag{8.5}$$

avec

$$\beta_{ki} = \frac{v_k^\mathsf{T} A u_i}{u_i^\mathsf{T} A u_i}. \tag{8.6}$$

8.2.2 Algorithme des directions conjuguées

Une méthode de directions conjuguées pour la minimisation d'une fonction quadratique f est une méthode qui à chaque itération prend $x_k \in x_1 + E_k$ de façon à minimiser f sur le sous-espace affine $x_1 + E_k$. Le lien avec la conjugaison des directions est donné par la proposition suivante.

On note

$$g_k = \nabla f(x_k) = A x_k - b$$

le gradient de f en x_k pour le produit scalaire euclidien et

$$y_k = g_{k+1} - g_k.$$

Proposition 8.5 *Si x_k réalise le minimum de f sur $x_1 + E_{k-1}$ ($k \geqslant 2$), $d_k \neq 0$ et $x_{k+1} = x_k + \alpha_k d_k$, alors x_{k+1} réalise le minimum de f sur $x_1 + E_k$ si, et seulement si, l'une des conditions suivantes est réalisée*
(i) $g_k^\mathsf{T} d_k = 0$ et $\alpha_k = 0$,
(ii) $g_k^\mathsf{T} d_k \neq 0$, d_k est conjuguée par rapport à $d_1, \dots, d_{k-1}$ et

$$\alpha_k = -\frac{g_k^\mathsf{T} d_k}{d_k^\mathsf{T} A d_k}. \tag{8.7}$$

Démonstration. Si x_k réalise le minimum de f sur $x_1 + E_{k-1}$, g_k est orthogonal à E_{k-1}, c'est-à-dire :

$$g_k^\mathsf{T} d_i = 0, \quad \text{pour } i = 1, \dots, k-1. \tag{8.8}$$

D'autre part, f étant quadratique, on a

$$g_{k+1} = g_k + A(x_{k+1} - x_k) = g_k + \alpha_k A d_k. \tag{8.9}$$

Alors, comme en (8.8), x_{k+1} réalise le minimum de f sur $x_1 + E_k$ si, et seulement si,

$$g_{k+1}^{\mathsf{T}} d_i = 0, \quad \text{pour } i = 1, \dots, k.$$

D'après (8.9), ceci est équivalent à

$$g_k^{\mathsf{T}} d_i + \alpha_k d_k^{\mathsf{T}} A d_i = 0, \quad \text{pour } i = 1, \dots, k.$$

En prenant $i = k$, on trouve soit que $g_k^{\mathsf{T}} d_k = 0$ et alors $\alpha_k = 0$ ($d_k \neq 0$), soit que $g_k^{\mathsf{T}} d_k \neq 0$ et α_k est donné par (8.7). En prenant $1 \leqslant i \leqslant k-1$ et en tenant compte de (8.8), on trouve dans le cas (ii) que d_k est conjuguée par rapport à $d_1, \dots, d_{k-1}$. Inversement, on voit que les conditions (i) ou (ii) sont suffisantes. □

Le pas α_k donné en (8.7) est optimal dans le sens où il réalise le minimum de l'application

$$\alpha \to f(x_k + \alpha d_k).$$

D'après la proposition 8.5, si l'on veut minimiser f sur les sous-espaces affines successifs $x_1 + E_1$, $x_1 + E_2$..., il suffit de se donner n directions conjuguées $d_1, \dots, d_n$ et de minimiser f le long des droites $\alpha \to x_k + \alpha d_k$. Ces directions conjuguées peuvent s'obtenir à partir de la donnée de n directions linéairement indépendantes $\tilde{d}_1, \dots, \tilde{d}_n$ que l'on utilisera pour construire des directions conjuguées au moyen du procédé de Gram-Schmidt (8.5)–(8.6), où v_i devient $\tilde{d}_i$ et u_i devient d_i. On obtient alors l'algorithme suivant :

Algorithme 8.6 (des directions conjuguées) Une itération passe de l'itéré courant $x_k \in \mathbb{R}^n$ à l'itéré suivant $x_{k+1} \in \mathbb{R}^n$ par les étapes suivantes :

1. *Calcul de la direction* : on choisit une direction arbitraire $\tilde{d}_k \notin \text{vect}\{\tilde{d}_1, \dots, \tilde{d}_{k-1}\}$ et on calcule la direction d_k par la formule

$$d_k = \begin{cases} \tilde{d}_1 & \text{si} \quad k = 1, \\ \tilde{d}_k - \displaystyle\sum_{i=1}^{k-1} \frac{\tilde{d}_k^{\mathsf{T}} A d_i}{d_i^{\mathsf{T}} A d_i} d_i & \text{si} \quad k \geqslant 2. \end{cases} \tag{8.10}$$

2. *Calcul du pas optimal* :

$$\alpha_k = -\frac{g_k^{\mathsf{T}} d_k}{d_k^{\mathsf{T}} A d_k}.$$

3. *Nouveau point* : $x_{k+1} = x_k + \alpha_k d_k$.

Cet algorithme ne demande pas la connaissance explicite de la matrice A. Il suffit de pouvoir calculer le produit de cette matrice par un vecteur arbitraire. Ce produit matrice-vecteur intervient à deux reprises dans l'algorithme ci-dessus : dans le calcul du gradient et dans la détermination du pas optimal. Si cette opération est coûteuse en temps de calcul, on peut n'en faire qu'une à chaque itération, pour calculer Ad_k, et en mettant à jour le gradient par la formule $g_{k+1} = g_k + \alpha_k A d_k$.

La méthode des directions conjuguées peut être améliorée en ce qui concerne l'encombrement mémoire, mais nous allons voir qu'un choix judicieux des directions $\tilde{d}_1$, $\ldots$, $\tilde{d}_n$ conduit à une amélioration beaucoup plus importante encore.

8.2.3 Algorithme du gradient conjugué

La méthode du gradient conjugué est le cas particulier de la méthode précédente ou l'on prend

$$\tilde{d}_k = -g_k. \tag{8.11}$$

Comme on va le voir, ce choix conduit à une simplification considérable de l'algorithme des directions conjuguées.

Proposition 8.7 *Dans la méthode du gradient conjugué, le sous-espace vectoriel E_k engendré par les directions d_1, $\ldots$, d_k est aussi le sous-espace vectoriel engendré par les gradients successifs :*

$$E_k = \mathrm{vect}\{g_1, \ldots, g_k\} \tag{8.12}$$

et ceux-ci sont orthogonaux entre eux :

$$g_k^{\mathsf{T}} g_i = 0, \quad 1 \leqslant i \leqslant k-1. \tag{8.13}$$

Démonstration. En effet, grâce au choix (8.11), les directions d_1, $\ldots$, d_{k-1} sont à présent obtenues en orthogonalisant $-g_1$, $\ldots$, $-g_{k-1}$ (pour le produit scalaire associé à A). La relation (8.12) est alors une conséquence du procédé de Gram-Schmidt.

Ensuite, comme x_k réalise un minimum de f sur $x_1 + E_{k-1}$, g_k est orthogonal à E_{k-1}. Par (8.12), on en déduit (8.13). □

D'après la relation d'orthogonalité des gradients (8.13), on a

$$g_k^{\mathsf{T}} y_i = g_k^{\mathsf{T}} (g_{i+1} - g_i) = 0, \quad \text{pour } 1 \leqslant i \leqslant k-2.$$

Alors, en utilisant (8.9), l'expression du coefficient de d_i dans (8.10) devient

$$\frac{\tilde{d}_k^{\mathsf{T}} y_i}{d_i^{\mathsf{T}} y_i}.$$

Compte tenu de la relation d'orthogonalité des gradients, on voit que seul le dernier terme de la somme dans (8.10) est non nul et que l'opposé de son coefficient s'écrit

$$\beta_k = \frac{\|g_k\|^2}{\|g_{k-1}\|^2}. \tag{8.14}$$

On a utilisé le fait que, d'après (8.13), $-g_k^\mathsf{T} y_{k-1} = \|g_k\|^2$ et $d_{k-1}^\mathsf{T} y_{k-1} = -d_{k-1}^\mathsf{T} g_{k-1} = \|g_{k-1}\|^2$. Ceci conduit à l'algorithme du gradient conjugué pour les fonctions quadratiques.

Algorithme 8.8 (du gradient conjugué) Une itération met à jour l'itéré courant $x_k \in \mathbb{R}^n$, le gradient courant g_k et le carré de la norme de ce dernier $\gamma_k := \|g_k\|_2^2$, par les étapes suivantes.

1. *Test d'arrêt* : si $\gamma_k \simeq 0$, on s'arrête.
2. *Paramètre de conjugaison* : si $k \geqslant 2$, $\beta_k := \gamma_k/\gamma_{k-1}$.
3. *Déplacement en* x :

$$d_k = \begin{cases} -g_1 & \text{si } k = 1 \\ -g_k + \beta_k d_{k-1} & \text{si } k \geqslant 2. \end{cases}$$

4. *Déplacement en* g : $p_k = Ad_k$.
5. *Calcul du pas* : $\alpha_k = \gamma_k/(d_k^\mathsf{T} p_k)$.
6. *Nouveau point* : $x_{k+1} = x_k + \alpha_k d_k$.
7. *Nouveau gradient* : $g_{k+1} = g_k + \alpha_k p_k$ et $\gamma_{k+1} = \|g_{k+1}\|_2^2$.

8.2.4 Propriétés de l'algorithme du gradient conjugué

Terminaison finie

Notons que tant que $g_k \neq 0$, les directions $g_1, \ldots, g_k$ sont linéairement indépendantes. Cela résulte de la condition d'orthogonalité (8.13). Les directions $d_1, \ldots, d_k$ seront donc conjuguées, et le sous-espace vectoriel E_k sera de dimension k. Par conséquent, *l'algorithme du gradient conjugué trouve la solution en au plus n itérations.* On peut être plus précis.

On peut montrer que le sous-espace vectoriel E_k n'est autre que le sous-espace de Krylov

$$K_k(A, g_1) \equiv \text{vect}\{g_1, Ag_1, \ldots, A^{k-1} g_1\},$$

associé à A et g_1 (exercice 8.1). Ces sous-espaces saturent en $k = s$ lorsque

$$A^{-1} g_1 \in K_s.$$

Comme $A^{-1} g_1 = x_1 - x_*$, la saturation se produit lorsque $x_* \in x_1 + K_s$. Comme la méthode du gradient conjugué minimise f sur $x_1 + E_k = x_1 + K_k$, elle s'arrêtera exactement en $k = s(A, g_1)$. Une matrice symétrique définie positive étant non défective, on a, grâce au corollaire 8.3, $s(A, g_1) \leqslant t$, où t est le nombre de valeurs propres distincts de A. Nous avons donc montré la proposition suivante.

Proposition 8.9 *La méthode du gradient conjugué trouve le minimum d'une fonction quadratique strictement convexe en au plus t itérations, où t est le nombre de valeurs propres distinctes de la hessienne de la fonction.*

Vitesse de convergence

Si n est grand (par exemple $n \geqslant 10^3$), il est trop coûteux d'effectuer n itérations de l'algorithme du gradient conjugué pour obtenir la solution. La qualité de l'approximation x_k obtenue peut être estimée par la vitesse de convergence de l'algorithme. Le résultat suivant [Luenberger (1973), pg. 187] donne une estimation de la décroissance de la norme de l'erreur $x_k - x_*$, en termes du conditionnement ℓ_2 de la matrice A :

$$\kappa_2(A) = \frac{\lambda_{\max}(A)}{\lambda_{\min}(A)}.$$

La norme utilisée est la suivante :

$$\|x\|_A = (x^\mathsf{T} A x)^{1/2}.$$

Proposition 8.10 *Soit κ le conditionnement ℓ_2 de A. La suite $\{x_k\}$ générée par l'algorithme du gradient conjugué vérifie l'estimation*

$$\|x_{k+1} - x_*\|_A \leqslant 2 \left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1} \right)^k \|x_1 - x_*\|_A.$$

Il s'agit d'une vitesse de convergence r-linéaire, dont le taux est à comparer avec celui de la méthode du gradient à pas optimal (proposition 7.2), qui forme une suite $\{x_k\}$ vérifiant :

$$\|x_{k+1} - x_*\|_A \leqslant \left(\frac{\kappa - 1}{\kappa + 1} \right) \|x_k - x_*\|_A \leqslant \left(\frac{\kappa - 1}{\kappa + 1} \right)^k \|x_1 - x_*\|_A.$$

La convergence est donc plus rapide avec la méthode du gradient conjugué. La figure 8.1 permet de comparer les taux de convergence des méthodes du Gradient et du gradient conjugué en fonction du conditionnement κ.

Si la matrice symétrique A n'est pas définie positive

Si A a des valeurs propres strictement positives et négatives, le dénominateur apparaissant dans le calcul du pas α_k (étape 3.5) n'est plus nécessairement strictement positif ; il peut être arbitrairement proche de zéro ou même nul, ce qui entraîne l'instabilité ou l'échec de l'algorithme. Ce mauvais comportement de l'algorithme

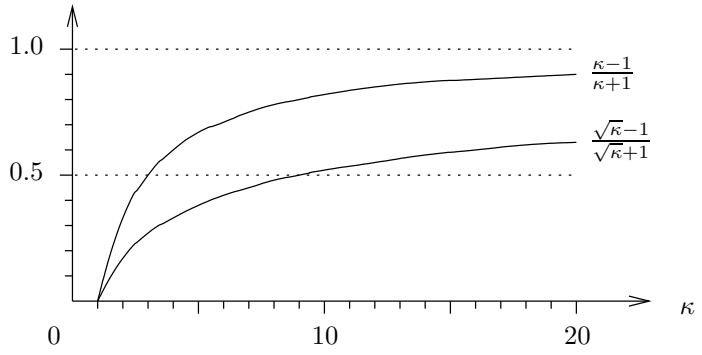

Fig. 8.1. Comparaison des taux de convergence du G et du GC

s'observe en général dès les premières itérations, si bien que l'on ne peut espérer trouver ainsi une solution approchée du système linéaire. Évidemment, si toutes les valeurs propres de A sont strictement négatives, on pourra résoudre par gradient conjugué le système linéaire $(-A)x = -b$, équivalent à (8.1), puisque sa matrice $-A$ est définie positive. En conclusion, *l'algorithme du gradient conjugué ne convient pas pour résoudre des systèmes linéaires dont la matrice est indéfinie, même de manière approchée.*

Le comportement de l'algorithme du GC lorsque $A \succcurlyeq 0$ est plus satisfaisant. Il est décrit dans la proposition suivante. L'algorithme est dit bien défini à l'itération k si $d_k^{\mathsf{T}} A d_k > 0$, si bien que le pas α_k est bien défini à l'étape 3.5 et que l'on peut passer de x_k à x_{k+1}. Enfin, lorsque $b \in \mathcal{R}(A)$, la *solution de norme minimale* de $Ax = b$ est l'unique solution du problème

$$\begin{cases} \min \ \frac{1}{2}\|x\|_2^2 \\ Ax = b, \end{cases}$$

qui est donc la projection de 0 sur le sous-espace affine $\{x \in \mathbb{R}^n : Ax = b\}$. Comme A est symétrique, les conditions d'optimalité de ce problème, qui sont ici nécessaires et suffisantes car le problème est convexe, montrent que c'est l'unique point x vérifiant $Ax = b$ et $x \in \mathcal{R}(A)$.

Proposition 8.11 (GC lorsque $A \succcurlyeq 0$) *On considère l'algorithme du gradient conjugué pour résoudre le système linéaire (8.1) ou minimiser la fonction quadratique f définie en (8.2), avec A symétrique semi-définie positive de rang r.*

1) *Si $b \notin \mathcal{R}(A)$, l'algorithme est bien défini pendant au plus r itérations, disons jusqu'en x_l, où la direction d_l générée est non nulle, dans le noyau de A et telle que $f(x_l + \alpha d_l) = f(x_l) - \alpha\, b^{\mathsf{T}} d_l \to -\infty$ lorsque $\alpha \to +\infty$.*
2) *Si $b \in \mathcal{R}(A)$, l'algorithme est bien défini et converge en au plus r itérations ; de plus, si l'itéré initial est pris dans $\mathcal{R}(A)$ (par exemple $x_1 = 0$), les itérés convergent vers la solution de norme minimale de (8.1).*

DÉMONSTRATION. 1) Si A est semi-définie positive et $b \notin \mathcal{R}(A)$, le système linéaire (8.1) n'a pas de solution. L'algorithme du GC peut itérer tant que le dénominateur $d_k^\mathsf{T} A d_k$ de la fraction donnant le pas α_k à l'étape 3.5 reste strictement positif (avec une instabilité possible si ce dénominateur s'approche de zéro). Alors l'itéré suivant x_{k+1} minimise toujours f sur l'espace de Krylov $K_k(A, g_1)$. Observons que K_k est de dimension k et que $K_k \cap \mathcal{N}(A) = \{0\}$, si bien que cette situation ne peut se poursuivre au-delà de $\dim K_k \leqslant n - \dim \mathcal{N}(A) = \dim \mathcal{R}(A) = r$ itérations : l'algorithme finit par trouver une direction $d_l \in \mathcal{N}(A)$ et doit s'arrêter. Comme $g_l \perp K_{l-1} \ni d_{l-1}$, $g_l^\mathsf{T} d_l = -\|g_l\|_2^2 < 0$ ($g_l \neq 0$ car $b \notin \mathcal{R}(A)$), d_l est une direction de descente (non nulle) de f en x_l. On en déduit que $f(x_l + \alpha d_l) = f(x_l) - \alpha\, b^\mathsf{T} d_l \downarrow -\infty$ lorsque $\alpha \uparrow +\infty$ ($b^\mathsf{T} d_l > 0$ car $0 > g_l^\mathsf{T} d_l = (Ax_l - b)^\mathsf{T} d_l = -b^\mathsf{T} d_l$).

2) Montrons que $x_k \in x_1 + \mathcal{R}(A)$, pour tout $k \geqslant 1$. On obtient ce résultat par récurrence en montrant que toutes les directions de recherche du GC vérifient $d_k \in \mathcal{R}(A)$. C'est clairement vrai pour la première direction qui s'écrit $d_1 = -g_1 = -Ax_1 + b \in \mathcal{R}(A)$, puisque $b \in \mathcal{R}(A)$. Ensuite, pour $k \geqslant 2$, $d_k = -g_k + \beta_k d_{k-1}$ est dans $\mathcal{R}(A)$ par récurrence.

Pour vérifier que le GC est bien défini, il suffit de vérifier que $d_k^\mathsf{T} A d_k$ est strictement positif tant que x_k n'est pas solution de (8.1). Cette propriété résulte du fait que $d_k \in \mathcal{R}(A)$ et que A est définie positive sur $\mathcal{R}(A)$ ($d^\mathsf{T} A d = 0$ et $d \in \mathcal{R}(A)$ impliquent que $d \in \mathcal{N}(A) \cap \mathcal{R}(A) = \{0\}$).

Enfin, comme $\mathcal{R}(A)$ est de dimension r, qu'avant convergence les directions d_k sont linéairement indépendantes et engendrent K_k et que $K_k \subseteq \mathcal{R}(A)$, l'algorithme converge en au plus r itérations.

D'autre part, si x_* est la solution trouvée par l'algorithme, on a de ce qui précède $x_* - x_1 \in \mathcal{R}(A)$. Si $x_1 \in \mathcal{R}(A)$ on en déduit que $x_* \in \mathcal{R}(A)$ et $Ax_* = b$. Dès lors x_* est la solution de norme minimale de (8.1). □

8.2.5 Mise en œuvre de la méthode du gradient conjugué

Encombrement mémoire et comptage des opérations

L'algorithme 8.8 du gradient conjugué est particulièrement économe en place mémoire. C'est l'un de ses principaux attraits. Il ne requiert que le stockage de 4 vecteurs, ceux mémorisant x_k, g_k, d_k et p_k (les trois premiers sont mis à jour à chaque itération en remplaçant le vecteur de l'itération précédente).

Le nombre d'opérations par itération est en $O(10n)$, plus un unique produit matrice-vecteur ($O(2n^2)$ opérations si la matrice est pleine). En arithmétique exacte, nous avons vu que l'algorithme requiert au plus n itérations pour trouver la solution. Il faut donc $O(2n^3)$ opérations pour trouver la solution d'un système linéaire symétrique défini positif plein par l'algorithme du gradient conjugué. On retrouve un nombre d'opérations du même ordre que la factorisation gaussienne.

Le gradient conjugué préconditionné

Dès que le nombre n de variables dépasse quelques dizaines ou centaines (cela dépend du conditionnement de A), l'algorithme du GC peut avoir des difficultés à minimiser la fonction quadratique et, du fait des erreurs d'arrondi, il peut demander

beaucoup plus de n itérations pour avoir une solution acceptable. Pour remédier à cette situation, il faut essayer d'améliorer le conditionnement du problème, c'est-à-dire *préconditionner* le problème ou l'algorithme. Nous présentons ci-après l'approche qui utilise un changement de variables.

Si l'on fait le changement de variables

$$\tilde{x} = Lx, \tag{8.15}$$

au moyen d'une matrice L d'ordre n inversible, la fonction $\tilde{f} : \mathbb{R}^n \to \mathbb{R}$ définie par

$$\tilde{f} = f \circ L^{-1}$$

est telle que $\tilde{f}(\tilde{x}) = f(x)$, si $\tilde{x}$ et x se correspondent par (8.15). Les fonctions f et $\tilde{f}$ atteignent donc la même valeur minimale et ce aux points x_* et $\tilde{x}_* = Lx_*$, respectivement. L'algorithme du GC préconditionné s'obtient en appliquant l'algorithme du GC standard dans l'espace des $\tilde{x}$ et en traduisant l'algorithme résultant dans l'espace des x. L'algorithme est ainsi modifié, amélioré si le changement de variables est bien choisi ; on dit aussi qu'il est sensible à un changement de variables.

Le gradient et la hessienne de $\tilde{f}$ en $\tilde{x} = Lx$ s'écrivent

$$\tilde{g} \equiv \nabla \tilde{f}(\tilde{x}) = L^{-\mathsf{T}} \nabla f(x) \qquad \text{et} \qquad \tilde{A} \equiv \nabla^2 \tilde{f} = L^{-\mathsf{T}} A L^{-1}.$$

Soient $\{\tilde{x}_k\}$ la suite générée par le GC dans l'espace des $\tilde{x}$ et $x_k := L^{-1}\tilde{x}_k$. On note $\tilde{g}_k := \nabla \tilde{f}(\tilde{x}_k)$ et $g_k = \nabla f(x_k)$, si bien que $\tilde{g}_k = L^{-\mathsf{T}} g_k$. La direction de la méthode du gradient conjugué dans l'espace des $\tilde{x}$ s'écrit

$$\tilde{d}_k = -\tilde{g}_k + \tilde{\beta}_k \tilde{d}_{k-1} = -L^{-\mathsf{T}} g_k + \frac{\|L^{-\mathsf{T}} g_k\|^2}{\|L^{-\mathsf{T}} g_{k-1}\|^2} \tilde{d}_{k-1}.$$

Ramenée à l'espace des x, cela donne pour $d_k = L^{-1}\tilde{d}_k$ la formule

$$d_k = \begin{cases} -P g_1 & \text{si} \quad k = 1, \\ -P g_k + \dfrac{g_k^{\mathsf{T}} P g_k}{g_{k-1}^{\mathsf{T}} P g_{k-1}} d_{k-1} & \text{si} \quad k \geqslant 2, \end{cases} \tag{8.16}$$

où

$$P := L^{-1} L^{-\mathsf{T}}.$$

On montre que g_k est encore orthogonal à d_{k-1} (voir l'exercice 8.3), si bien que le pas optimal le long de d_k est donné par la formule

$$\alpha_k = \frac{g_k^{\mathsf{T}} P g_k}{d_k^{\mathsf{T}} A d_k}. \tag{8.17}$$

Si l'on remplace la direction d_k et le pas α_k de la méthode du gradient conjugué par les valeurs données par les formules (8.16) et (8.17), on obtient la *méthode du gradient conjugué préconditionné.*

Algorithme 8.12 (du gradient conjugué préconditionné) Étant donnée une matrice de préconditionnement P, une itération met à jour l'itéré courant $x_k \in \mathbb{R}^n$, le gradient courant g_k et la norme préconditionnée au carré de ce dernier $\gamma_k := g_k^\mathsf{T} P g_k$, par les étapes suivantes.

1. *Test d'arrêt* : si $\gamma_k \simeq 0$, on s'arrête.
2. *Paramètre de conjugaison* : si $k \geqslant 2$, $\beta_k := \gamma_k / \gamma_{k-1}$.
3. *Déplacement en x* :

$$d_k = \begin{cases} -P g_1 & \text{si } k = 1 \\ -P g_k + \beta_k d_{k-1} & \text{si } k \geqslant 2. \end{cases}$$

4. *Déplacement en g* : $p_k = A d_k$.
5. *Calcul du pas* : $\alpha_k = \gamma_k / (d_k^\mathsf{T} p_k)$.
6. *Nouveau point* : $x_{k+1} = x_k + \alpha_k d_k$.
7. *Nouveau gradient* : $g_{k+1} = g_k + \alpha_k p_k$ et $\gamma_{k+1} = g_{k+1}^\mathsf{T} P g_{k+1}$.

Le choix de L, donc de P, est gouverné par le souhait d'avoir $\tilde{A}$ proche de la matrice identité, ce qui a pour effet d'accélérer la convergence de l'algorithme dans l'espace des $\tilde{x}$ (proposition 8.10), donc aussi dans l'espace des x. On voit qu'il est souhaitable de prendre L proche de $A^{1/2}$, ou encore

$$P \simeq A^{-1}.$$

Préconditionneur diagonal

Le *préconditionneur diagonal*, qui consiste à prendre pour P une matrice diagonale, est sans doute le plus simple. On peut par exemple prendre $L = \mathrm{Diag}(A_{ii}^{1/2})$ en (8.15) (on se rappelle que les $A_{ii} > 0$) et donc

$$P = \mathrm{Diag}(A_{11}, \dots, A_{nn})^{-1}. \tag{8.18}$$

Ce dernier préconditionneur diagonal a une propriété de minimalité, dans le sens où il minimise, à un facteur n près, le conditionnement de DAD parmi toutes les matrices diagonales D. En effet, selon van der Sluis [599 ; théorème 4.1], on a

$$\kappa_2(P^{1/2} A P^{1/2}) \leqslant n \left(\min_{\substack{D \text{ diagonale} \\ D \succ 0}} \kappa_2(DAD) \right).$$

S'il a le mérite de la simplicité, sauf cas exceptionnels (celui d'une matrice A diagonale est un cas particulier évident), ce préconditionneur élémentaire n'apporte pas toujours une amélioration notable de la vitesse de convergence de l'algorithme du gradient conjugué. Il ne doit toutefois pas être négligé, si aucun autre préconditionneur ne s'impose.

Si les éléments diagonaux A_{ii}, pour $i \in I$, sont nuls, la formule (8.18) n'est plus bien définie. Cependant, la semi-définie positivité de A implique alors que les colonnes (et les lignes) de A avec indice dans I sont nulles, si bien que les vecteurs de base $\{e^i\}_{i\in I}$ sont dans le noyau de A. Si $b_I = 0$, les itérés de l'algorithme du gradient conjugué sont générés dans $x_1 + \mathrm{vect}\{e^i : i \in I\}^\perp$ et on peut prendre le préconditionneur diagonal P avec $P_{ii} = 0$ si $i \in I$ et $P_{ii} = A_{ii}^{-1}$ sinon. S'il existe un indice $i \in I$ tel que $b_i \neq 0$, la direction $d = \mathrm{sgn}(b_i)e^i$ est une *direction de non bornitude* de la forme quadratique associée $x \mapsto f(x) = \frac{1}{2}x^\mathsf{T}Ax - b^\mathsf{T}x$, dans le sens où, quel que soit $x \in \mathbb{R}^n$, $f(x + td) \to -\infty$ lorsque $t \to \infty$; l'équation $Ax = b$ n'a pas de solution.

Préconditionneur projecteur

Supposons à présent que le préconditionneur symétrique $P \succcurlyeq 0$ soit un projecteur ($P^2 = P$) non trivial ($P \neq I$, donc avec des valeurs propres nulles). L'algorithme du GC préconditionné par P, utilisant les directions (8.16) et les pas (8.17), est encore bien défini et génère une suite dans $x_1 + \mathcal{R}(P)$ (voir l'exercice 8.3). Il porte le nom d'*algorithme du gradient conjugué projeté* pour minimiser le critère quadratique sur le sous-espace affine $x_1 + \mathcal{R}(P)$.

Redémarrage du gradient conjugué

La méthode du gradient conjugué est une méthode dans laquelle les erreurs d'arrondi sont amplifiées au cours des itérations. Peu à peu les relations de conjugaison de d_k avec les premières directions sont perdues parce qu'en présence d'erreurs d'arrondi les relations (8.13) d'orthogonalité des gradients ne sont pas vérifiées exactement.

Notons que les erreurs d'arrondi sont mieux contrôlées si l'on remplace la formule (8.14) de β_k, dite de *Fletcher-Reeves* (1964), par la formule équivalente, dite de *Polak-Ribière* (1969) :

$$\beta_k = \frac{g_k^\mathsf{T} y_{k-1}}{\|g_{k-1}\|^2}.$$

On peut détecter la présence d'erreurs d'arrondi en regardant si

$$|g_k^\mathsf{T} g_{k-1}| \geqslant \nu \|g_k\|^2,$$

où $\nu \simeq 0.2$ (normalement le membre de gauche doit être nul). Ce critère est connu sous le nom de *critère de redémarrage de Powell* (1977). S'il est vérifié pour $k = r$, certains numériciens préconisent de redémarrer l'algorithme du gradient conjugué en x_r dans la direction $-g_r$. Cette pratique n'est plus guère utilisée.

8.2.6 Méthode du gradient conjugué non linéaire

On s'intéresse ici à la minimisation d'une fonction $f : \mathbb{R}^n \to \mathbb{R}$, non nécessairement quadratique :

$$\min_{x\in\mathbb{R}^n} f(x),$$

et on cherche à étendre la méthode du gradient conjugué à ce problème. Il y a plusieurs manières de le faire et peu de critères permettant de dire laquelle est la meilleure. Une

extension possible consiste simplement à reprendre les formules utilisées dans le cas quadratique. On se propose donc d'étudier les méthodes où la direction d_k est définie par la formule de récurrence suivante ($\beta_k \in \mathbb{R}$)

$$d_k = \begin{cases} -g_1 & \text{si} \quad k = 1, \\ -g_k + \beta_k d_{k-1} & \text{si} \quad k \geqslant 2, \end{cases} \tag{8.19}$$

et où $\{x_k\}$ est générée par la formule

$$x_{k+1} = x_k + \alpha_k d_k, \tag{8.20}$$

le pas $\alpha_k \in \mathbb{R}$ étant déterminé par une recherche linéaire.

Ces méthodes sont des extensions de la méthode du gradient conjugué si β_k prend l'une des valeurs

$$\begin{aligned} \beta_k^{\mathrm{FR}} &= \frac{\|g_k\|^2}{\|g_{k-1}\|^2}, \\ \beta_k^{\mathrm{PR}} &= \frac{g_k^{\mathsf{T}} y_{k-1}}{\|g_{k-1}\|^2}, \end{aligned}$$

où $y_{k-1} = g_k - g_{k-1}$. Dans le cas quadratique avec recherche linéaire exacte, on a vu que $\beta_k^{\mathrm{FR}} = \beta_k^{\mathrm{PR}}$. Si f est quelconque, il n'en est plus de même et on parle respectivement de *méthode de Fletcher-Reeves* (1964) ou de *méthode de Polak-Ribière* (1969) selon que l'on utilise β_k^{FR} ou β_k^{PR} à la place de β_k dans (8.19).

Pour que les méthodes ainsi définies soient utilisables, il faut répondre aux deux questions suivantes. Les directions d_k définies par (8.19) sont-elles des directions de descente de f ? Les méthodes ainsi définies sont-elles convergentes ?

En ce qui concerne la première question remarquons que, quel que soit $\beta_k \in \mathbb{R}$, d_k est une direction de descente si l'on fait de la recherche linéaire exacte, c'est-à-dire si le pas α_{k-1} est un point stationnaire de $\alpha \to f(x_{k-1} + \alpha d_{k-1})$. En effet, dans ce cas $g_k^{\mathsf{T}} d_{k-1} = 0$ et on trouve lorsque $g_k \neq 0$:

$$g_k^{\mathsf{T}} d_k = -\|g_k\|^2 < 0.$$

Cependant, il est fortement déconseillé de faire de la recherche linéaire exacte lorsque f n'est pas quadratique : le coût de détermination de α_k est excessif.

Le résultat suivant, que l'on peut trouver dans [245 ; 1992], généralise un résultat de Al-Baali (1985). Il montre que si β_k n'est pas trop grand et si l'on utilise la règle de *recherche linéaire de Wolfe forte* :

$$f(x_{k+1}) \leqslant f(x_k) + \omega_1 \alpha_k g_k^{\mathsf{T}} d_k, \tag{8.21}$$

$$|g_{k+1}^{\mathsf{T}} d_k| \leqslant \omega_2 |g_k^{\mathsf{T}} d_k|, \tag{8.22}$$

avec des constantes positives ω_1 et ω_2 bien choisies, alors d_k est une direction de descente et la méthode converge.

Proposition 8.13 *Supposons que l'ensemble* $\mathcal{L} := \{x \in \mathbb{R}^n : f(x) \leqslant f(x_1)\}$ *soit borné et que* f *soit continûment différentiable dans un voisinage de* $\mathcal{L}$ *avec un gradient lipschitzien. Toute méthode du type* (8.19)–(8.20) *dans laquelle* β_k *vérifie*

$$|\beta_k| \leqslant \beta_k^{FR}, \quad \forall k \geqslant 1,$$

et le pas α_k *est déterminé par la règle de Wolfe forte* (8.21)–(8.22) *avec* $0 < \omega_1 < \omega_2 < 1/2$ *est une méthode de descente* ($g_k^{\mathsf{T}} d_k < 0$, $\forall k \geqslant 1$) *convergente, dans le sens où*

$$\liminf_{k\to\infty} \|g_k\| = 0.$$

D'un point de vue théorique, ce résultat est satisfaisant et suggère que la méthode de Fletcher-Reeves est une bonne méthode. Cependant, en pratique, il est préférable d'utiliser la méthode de Polak-Ribière dont les performances moyennes dépassent de loin celles de la méthode de Fletcher-Reeves. Le résultat suivant est dû à Polak et Ribière (1964).

Proposition 8.14 *Si* f *est fortement convexe, de classe* C^1 *avec un gradient lipschitzien, alors la méthode de Polak-Ribière avec recherche linéaire exacte génère une suite* $\{x_k\}$ *convergeant vers l'unique point* x_* *réalisant le minimum de* f.

Démonstration. Montrons dans un premier temps que

$$\cos\theta_k = \frac{-g_k^{\mathsf{T}} d_k}{\|g_k\|\,\|d_k\|}$$

est uniformément positif. Grâce à la recherche linéaire exacte, on a

$$y_{k-1}^{\mathsf{T}} d_{k-1} = -g_{k-1}^{\mathsf{T}} d_{k-1} = \|g_{k-1}\|^2.$$

La forte convexité de f implique que

$$y_{k-1}^{\mathsf{T}} d_{k-1} = \frac{1}{\alpha_{k-1}} y_{k-1}^{\mathsf{T}}(x_k - x_{k-1}) \geqslant \frac{\gamma}{\alpha_{k-1}} \|x_k - x_{k-1}\|^2,$$

où $\gamma > 0$ est le module de forte convexité de f. On en déduit, en utilisant la constante de lipschitz L de f' :

$$|\beta_k^{\mathrm{PR}}| = \frac{|g_k^{\mathsf{T}} y_{k-1}|}{\|g_{k-1}\|^2} = \frac{|g_k^{\mathsf{T}} y_{k-1}|}{y_{k-1}^{\mathsf{T}} d_{k-1}} \leqslant \frac{\alpha_{k-1} L}{\gamma} \frac{\|g_k\|\|x_k - x_{k-1}\|}{\|x_k - x_{k-1}\|^2} = \frac{L}{\gamma} \frac{\|g_k\|}{\|d_{k-1}\|}.$$

On peut alors borner $\|d_k\|$ par

$$\|d_k\| \leqslant \|g_k\| + |\beta_k^{\mathrm{PR}}|\,\|d_{k-1}\| \leqslant \left(1 + \frac{L}{\gamma}\right) \|g_k\|.$$

Ensuite

$$g_k^{\mathsf{T}} d_k = -\|g_k\|^2 \leqslant -\left(1 + \frac{L}{\gamma}\right)^{-1} \|g_k\|\|d_k\|,$$

ou encore $\cos\theta_k \geqslant (1 + L/\gamma)^{-1}$.

D'après la proposition 6.10 et la recherche linéaire exacte, la condition de Zoutendijk (6.19) est vérifiée. Mais f et donc $\{f(x_k)\}$ est bornée inférieurement (car f est fortement convexe). On en déduit que $g_k \to 0$ (proposition 6.8). D'autre part, $\{x_k\}$ est bornée (f est fortement convexe) et possède donc des sous-suites convergentes. La limite de celles-ci ne peut être que l'unique minimum x_* de f (car $g_k \to 0$). Donc *toute* la suite $\{x_k\}$ converge vers x_*. □

Si f n'est pas convexe, la méthode de Polak-Ribière peut ne pas converger. Powell (1984) a donné un exemple de fonction pour laquelle l'algorithme génère une suite $\{x_k\}$ dont aucun des points d'adhérence n'est stationnaire. On peut trouver des remèdes simples à ce comportement inattendu [245 ; 1992]. Le plus simple et apparemment le plus efficace est de prendre

$$\beta_k = (\beta_k^{\mathrm{PR}})^+ \equiv \max(\beta_k^{\mathrm{PR}}, 0),$$

ce qui revient à redémarrer l'algorithme chaque fois que $\beta_k^{\mathrm{PR}} < 0$. On peut montrer la convergence globale pour cette valeur de β_k et une recherche linéaire adaptée. Les performances numériques de cette méthode sont très semblables à celles de la méthode de Polak-Ribière.

Comme pour la minimisation de fonctions quadratiques convexes, la méthode du gradient conjugué non linéaire est économe en place mémoire et en temps de calcul propre. Elle s'utilise encore parfois dans les deux situations suivantes : lorsque n est très grand relativement à la place mémoire disponible ou lorsque le coût de calcul de f et de son gradient est très faible (ce qui est plutôt rare dans les problèmes réels) et que l'on ne veut pas perdre de temps avec une méthode plus sophistiquée mais plus gourmande en temps de calcul. Dès que l'espace mémoire disponible est supérieur à, disons $10n$, il est souvent préférable d'utiliser une méthode de quasi-Newton à mémoire limitée, par exemple la méthode l-BFGS (voir la section 11.2.5).

8.3 Algorithme du résidu minimal généralisé (GMRES)

Revenons à la résolution numérique du système linéaire

$$Ax = b, \tag{8.23}$$

lorsque A est inversible mais n'est plus nécessairement symétrique. On note toujours

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

la solution unique de (8.23). On appelle *résidu* du système linéaire (8.23) en $x \in \mathbb{R}^n$, le vecteur $r \in \mathbb{R}^n$ défini par

$$r = b - Ax.$$

On note $r_k = b - Ax_k$ le résidu en x_k. Lorsque A est symétrique, le résidu est l'opposé du gradient de $f(x) = \frac{1}{2}x^\mathsf{T}Ax - b^\mathsf{T}x$.

On aimerait, pour les systèmes linéaires non symétriques, disposer d'un algorithme itératif ayant les propriétés attrayantes du gradient conjugué que sont le peu d'encombrement-mémoire (due à une formule de récurrence courte) et une propriété de minimalité sur l'espace de Krylov (qui lui confère de la stabilité). Ceci n'est malheureusement pas possible [611, 196]. Dans l'algorithme GMRES, décrit dans cette section, on ne garde que la propriété de minimalité ; quant à l'encombrement-mémoire, il va croître linéairement avec le nombre d'itérations, si bien que la mise en œuvre de l'algorithme se fait avec redémarrages (section 8.3.6).

L'algorithme GMRES est aujourd'hui l'algorithme itératif standard de résolution des systèmes non symétriques. Il a été proposé par Saad et Schultz [532 ; 1986].

8.3.1 Principe général

Partant d'un point $x_1 \in \mathbb{R}^n$, la *méthode du résidu minimal généralisée* ou *méthode GMRES* (Generalized Minimal RESidual) consiste à générer une suite de points $x_k \in \mathbb{R}^n$ par la formule

$$x_k = x_1 + z_k, \quad k \geqslant 1,$$

où z_k dans le sous-espace de Krylov associé au résidu initial :

$$K_k \equiv K_k(A, r_1) = \text{vect}\{r_1, Ar_1, \ldots, A^{k-1}r_1\}.$$

Cette façon de procéder est motivée par les deux remarques suivantes.

- Si x_1 est une bonne approximation de la solution il y a un sens à approcher x_* par des itérés x_k tels que $x_k - x_1$ appartienne pour k petit à un sous-espace vectoriel de faible dimension (ici un sous-espace de Krylov).
- Si l'on prend z_k dans un sous-espace de Krylov $K_k(A, r)$, pour un certain $r \in \mathbb{R}^n$, et si l'on veut obtenir la solution par cette méthode, il faut que

$$x_* \in x_1 + K_s(A, r), \tag{8.24}$$

où s est l'indice de saturation de ces sous-espaces (voir la section 8.1). Cet indice est caractérisé par le fait que

$$A^{-1}r \in K_s(A, r).$$

On voit qu'en prenant $r = r_1$, cette dernière condition est équivalente à (8.24).

La méthode détermine ensuite x_k dans $x_1 + K_k$, de manière à minimiser le résidu sur $x_1 + K_k$. Il s'agit donc de résoudre à chaque itération

$$\min_{x \in x_1 + K_k} \|b - Ax\|_2, \tag{8.25}$$

où $\|\cdot\|_2$ est la norme ℓ_2 de $\mathbb{R}^n$. Ce problème a une solution unique qui, pour $k = s(A, r_1)$, ne peut être que $x = x_*$ (qui annule le résidu sur $\mathbb{R}^n$). En écrivant $x = x_1 + z$, $z \in K_k$, on a $b - Ax = r_1 - Az$ et le problème (8.25) devient

$$\min_{z \in K_k} \|r_1 - Az\|_2. \tag{8.26}$$

Ceci montre que Az_k est la projection orthogonale de r_1 sur AK_k et donc

$$r_k = r_1 - Az_k \perp Az_k.$$

Le résidu est rendu orthogonal à AK_k.

Remarque 8.15 On peut aussi déterminer x_k dans $x_1 + K_k$ en rendant le résidu r_k orthogonal à K_k. On parle alors de *méthode du résidu orthogonal* ou de *méthode d'Arnoldi*. La méthode du gradient conjugué est de ce type (exercice 8.1). Numériquement les deux méthodes se valent [94]. □

Proposition 8.16 *La méthode GMRES converge en au plus β itérations, où β est le degré du polynôme minimal annihilant A (donc $\beta \leqslant n$). Si A est non défective, β est le nombre de valeurs propres distinctes de A.*

Démonstration. Il suffit de constater que la détermination de x_k par (8.25) donne $x_k = x_*$ dès que $x_* \in x_1 + K_{k-1}$. Ceci aura lieu exactement à la saturation du sous-espace de Krylov K_k, c'est-à-dire pour $k = s(A, r_1)$. Le résultat est alors une conséquence de la proposition 8.2 et de son corollaire. □

Venons-en maintenant au calcul effectif de x_k.

8.3.2 Construction d'une base orthonormale de K_{k+1}

Le problème (8.25) — ou (8.26) — est un problème avec contraintes que l'on peut transformer en problème sans contrainte en se donnant une base de K_k. Tant que $k < s(A, r_1)$, les vecteurs $r_1, Ar_1, \ldots, A^k r_1$ forment une base de K_{k+1}. Cependant, l'utilisation d'une base orthonormale s'avérera utile par la suite.

L'algorithme de Gram-Schmidt permet d'obtenir une base orthonormale de K_{k+1} en « orthonormalisant » les vecteurs $r_1, Ar_1, \ldots, A^k r_1$. Si l'on a déjà une base orthonormale de K_k formée des vecteurs $v_1, \ldots, v_k$, on voit que pour obtenir v_{k+1} il suffit d'orthonormaliser $A^k r_1$ par rapport à $v_1, \ldots, v_k$. On peut aussi orthonormaliser Av_k par rapport à $v_1, \ldots, v_k$ car on verra que, compte tenu de la manière dont les v_i sont calculés, on a

$$A^k r_1 \notin K_k \implies Av_k \in K_{k+1} \setminus K_k, \tag{8.27}$$

Ceci permet d'éviter le stockage de $A^k r_1$ dans un vecteur auxiliaire. La relation (8.27) se montre par récurrence. Mais auparavant, spécifions le calcul des vecteurs de base v_i en adaptant à notre cadre l'algorithme d'orthonormalisation de Gram-Schmidt (section B.1.1). On utilise le produit scalaire euclidien. Si $r_1 = b - Ax_1 = 0$, le point initial x_1 est solution et on s'arrête. Sinon, on prend

$$v_1 = \frac{r_1}{\|r_1\|_2}.$$

Ensuite, pour $k \geqslant 1$, v_{k+1} est calculé comme suit

$$\begin{array}{ll}(i) & h_{i,k} = v_i^{\mathsf{T}} A v_k, \quad \text{pour} \quad i = 1, \ldots, k\,; \\ (ii) & \tilde{v}_{k+1} = A v_k - \sum_{i=1}^{k} h_{i,k} v_i\,; \\ (iii) & h_{k+1,k} = \|\tilde{v}_{k+1}\|_2\,; \\ (iv) & v_{k+1} = \tilde{v}_{k+1}/h_{k+1,k}. \end{array}$$

DÉMONSTRATION. (de (8.27)). L'implication est vraie pour $k = 1$, puisque $Av_1 \parallel Ar_1 \in K_2 \setminus K_1$. Ensuite, si $Av_{k-1} \in K_k \setminus K_{k-1}$ (hypothèse de récurrence), on a

$$Av_{k-1} = \alpha A^{k-1} r_1 + w,$$

avec $\alpha \neq 0$ et $w \in K_{k-1}$. L'étape (ii) de l'algorithme précédent donne

$$\tilde{v}_k = \alpha A^{k-1} r_1 + \tilde{w},$$

avec $\tilde{w} \in K_{k-1}$. Mais $\tilde{v}_k \neq 0$, car $A^{k-1} r_1 \notin K_{k-1}$. D'après l'étape (iv) on trouve alors

$$v_k = \alpha' A^{k-1} r_1 + w',$$

où $\alpha' \neq 0$ et $w' \in K_{k-1}$. Donc $Av_k = \alpha' A^k r_1 + Aw' \in K_{k+1} \setminus K_k$ car $A^k r_1 \in K_{k+1} \setminus K_k$ et $Aw' \in K_k$. □

Remarquons que l'on a

$$Av_k = \sum_{i=1}^{k+1} h_{i,k} v_i.$$

Si l'on note $\bar{H}_k$ la matrice $(k+1) \times k$ dont les éléments h_{ij} sont calculés par l'algorithme précédent et $V_k = (v_1 \ \ldots \ v_k)$ la matrice $n \times k$ dont les colonnes sont formées des vecteurs de la base orthonormale de K_k, on a la relation

$$AV_k = V_{k+1} \bar{H}_k. \tag{8.28}$$

Comme

$$(\bar{H}_k)_{ij} = 0, \quad \text{pour} \quad i \geqslant j + 2,$$

les k premières lignes de $\bar{H}_k$ forment une matrice H_k de Hessenberg supérieure :

$$\bar{H}_k = \begin{pmatrix} H_k \\ 0 \cdots 0 \ h_{k+1,k} \end{pmatrix}.$$

8.3.3 Calcul de l'itéré x_k

Il s'agit de résoudre le problème (8.26). Comme $z \in K_k$ peut s'exprimer dans la base $v_1, \ldots, v_k$ par $z = V_k y$, $y \in \mathbb{R}^k$, (8.26) peut se récrire comme un problème de minimisation de

$$J(y) = \|r_1 - AV_k y\|_2,$$

pour $y \in \mathbb{R}^k$.

En utilisant (8.28) et le fait que $r_1 = \beta V_{k+1} e_1$, $e_1 = (1, 0, \ldots, 0)^{\mathsf{T}} \in \mathbb{R}^{k+1}$ et $\beta = \|r_1\|$, on a

$$J(y) = \|V_{k+1}(\beta e_1 - \bar{H}_k y)\|_2.$$

Compte tenu de l'orthonormalité des colonnes de V_{k+1} on obtient finalement

$$J(y) = \|\beta e_1 - \bar{H}_k y\|_2, \tag{8.29}$$

fonction qu'il s'agit de minimiser. On verra à la section 8.3.5 une méthode de résolution de ce problème de moindres-carrés, basée sur la factorisation QR de $\bar{H}_k$.

8.3.4 L'algorithme GMRES

Après la phase d'initialisation (étapes 1 à 3), l'algorithme GMRES se déroule en deux temps. Dans un premier temps (étape 4), on calcule une base de $K_s(A, r_1)$, l'indice s se détermine en détectant la saturation des espaces de Krylov. Dans un second temps (étape 5), on calcule la solution x_* du système linéaire en résolvant le problème (8.29).

Algorithme 8.17 (GMRES)

1. Initialisation : choix de $x_1 \in \mathbb{R}^n$ et calcul du résidu $r_1 = b - Ax_1$;
2. Si $r_1 = 0$ on s'arrête ;
3. $v_1 = r_1/\|r_1\|$;
4. Pour $k = 1, 2, \ldots$ faire :
   4.1. $h_{i,k} = v_i^{\mathsf{T}} A v_k$, pour $i = 1, \ldots, k$;
   4.2. $\tilde{v}_{k+1} = Av_k - \sum_{i=1}^{k} h_{i,k} v_i$;
   4.3. $h_{k+1,k} = \|\tilde{v}_{k+1}\|_2$;
   4.4. Si $h_{k+1,k} = 0$ aller en 5 ;
   4.5. $v_{k+1} = \tilde{v}_{k+1}/h_{k+1,k}$;
5. Calcul de la solution :
   5.1. $y_k = \arg\min_{y \in \mathbb{R}^k} J(y)$, où $J(y)$ donné par (8.29) ;
   5.2. $x_k = x_1 + V_k y_k$.

À l'étape 4.4, lorsque $h_{k+1,k} = 0$, on a $Av_k \in K_k$. D'après l'implication (8.27), cela entraîne que $K_{k+1} = K_k$: l'espace de Krylov K_k est saturé. Par conséquent, x_k calculé à l'étape 5 sera la solution x_*.

8.3.5 L'algorithme GMRES/QR

L'étape 5 de l'algorithme GMRES ne spécifie pas comment on résout le problème de minimisation de la fonction J donnée en (8.29). Dans cette section, nous montrons comment procéder en utilisant une factorisation QR de $\bar{H}_k$:

$$\bar{H}_k = Q_k \bar{R}_k, \quad \bar{R}_k = \begin{pmatrix} R_k \\ 0 \end{pmatrix},$$

où Q_k est une matrice orthogonale d'ordre $k+1$ et $\bar{R}_k$ est une matrice $(k+1) \times k$ dont les k premières lignes forment une matrice R_k triangulaire supérieure et la $(k+1)$-ième ligne est nulle.

Avec cette factorisation, J s'écrit

$$J(y) = \|Q_k^{\mathsf{T}}(\beta e_1 - \bar{H}_k y)\|_2 = \|\bar{q}_k - \bar{R}_k y\|_2,$$

où

$$\bar{q}_k = \beta Q_k^{\mathsf{T}} e_1. \tag{8.30}$$

On voit que le vecteur $y_k \in \mathbb{R}^k$ minimisant $J(y)$ sur $\mathbb{R}^k$ est solution du système triangulaire supérieur :

$$R_k y = q_k,$$

où $q_k \in \mathbb{R}^k$ est formé des k premiers éléments de $\bar{q}_k$. Remarquons aussi que la norme du résidu en x_k s'obtient comme la composante $k+1$ de $\bar{q}_k$:

$$\|r_k\| = J(y_k) = (\bar{q}_k)_{k+1}. \tag{8.31}$$

On a

$$\bar{q}_k = \begin{pmatrix} q_k \\ \|r_k\| \end{pmatrix}.$$

Voyons à présent comment calculer $\bar{R}_{k+1}$ et $\bar{q}_{k+1}$ à partir de $\bar{R}_k$ et $\bar{q}_k$ en n'utilisant qu'une rotation de Givens. Supposons que l'on dispose de la factorisation QR de $\bar{H}_k$:

$$Q_k^\mathsf{T} \bar{H}_k = \bar{R}_k = \begin{pmatrix} R_k \\ 0 \end{pmatrix},$$

et que la matrice orthogonale Q_k^T d'ordre $k+1$ soit le produit de p rotations de Givens d'ordre $k+1$ (en fait, on aura $p = k$)

$$Q_k^\mathsf{T} = G_p \cdots G_1.$$

La matrice $\bar{H}_{k+1}$ s'obtient à partir de $\bar{H}_k$ en lui adjoignant une ligne et une colonne supplémentaire

$$\bar{H}_{k+1} = \begin{pmatrix} & & & \times \\ & \bar{H}_k & & \vdots \\ & & & \times \\ 0 & \cdots & 0 & h_{k+2,k+1} \end{pmatrix},$$

où $\times$ désigne des éléments éventuellement non nuls. Les matrices d'ordre $k+2$

$$\tilde{G}_i = \begin{pmatrix} & & & 0 \\ & G_i & & \vdots \\ & & & 0 \\ 0 & \cdots & 0 & 1 \end{pmatrix}, \quad 1 \leqslant i \leqslant p$$

sont encore des rotations de Givens. L'application de $\tilde{Q}_{k+1}^\mathsf{T} = \tilde{G}_p \cdots \tilde{G}_1$ à $\bar{H}_{k+1}$ laissera sa dernière ligne inchangée et l'on aura

$$\tilde{Q}_{k+1}^\mathsf{T} \bar{H}_{k+1} = \begin{pmatrix} & & & \times \\ & \bar{R}_k & & \vdots \\ & & & \times \\ 0 & \cdots & 0 & h_{k+2,k+1} \end{pmatrix}.$$

On voit que pour annuler la dernière ligne de la matrice dans le membre de droite, il suffira d'utiliser une rotation de Givens supplémentaire $\tilde{G}_{p+1}$.

Quant au vecteur $\bar{q}_{k+1}$ défini en (8.30), il s'obtient à partir de $\bar{q}_k$, comme suit ($e_1 \in \mathbb{R}^{k+1}$)

$$\bar{q}_{k+1} = \tilde{G}_{p+1} \tilde{G}_p \cdots \tilde{G}_1 \begin{pmatrix} \beta e_1 \\ 0 \end{pmatrix} = \tilde{G}_{p+1} \begin{pmatrix} \bar{q}_k \\ 0 \end{pmatrix}.$$

En pratique, on modifie l'algorithme GMRES pour contrôler son arrêt au moyen d'un seuil de tolérance $\varepsilon > 0$: on s'arrête si $\|r_k\| \leqslant \varepsilon$. Ce contrôle peut facilement être pris en compte dans le test d'arrêt de l'étape 2. Quant à l'arrêt au cours de l'étape 4.4, il peut se faire sans le calcul explicite de x_k, en contrôlant le résidu par $(\bar{q}_k)_{k+1}$; voir (8.31).

8.3.6 Algorithme GMRES avec redémarrage

Pour la résolution de grands systèmes linéaires, l'algorithme GMRES a l'inconvénient de devoir mémoriser un nombre de vecteurs croissant avec le nombre d'itérations. Il faut en effet mémoriser V_k et les k rotations de Givens donnant Q_k^{T} et R_k. Pour remédier à cet inconvénient, on peut redémarrer la méthode toutes les m itérations.

Cependant, une certaine prudence s'impose dans l'utilisation d'un tel algorithme, car il n'est pas nécessairement convergent. On peut facilement construire un exemple dans le cas où $m = 1$. Il suffit en effet que sur $x_1 + K_1$, le résidu soit minimal en x_1. Alors $x_2 = x_1$ et on ne quitte pas ce point si l'on redémarre la méthode à chaque itération. C'est le cas de l'exemple suivant

$$A = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \quad b = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \quad x_1 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

On trouve

$$r_1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

Alors $x_2 = x_1 + \bar{\alpha} r_1$, où $\bar{\alpha} = 0$ est la solution de

$$\min_{\alpha} \|b - \alpha A r_1\|_2 = \min_{\alpha} \left\| \begin{pmatrix} 1 - \alpha \\ 1 + \alpha \end{pmatrix} \right\|_2.$$

Notes

Rappelons qu'il existe des algorithmes de résolution de systèmes linéaires par factorisation de la matrice, demandant moins de $O(n^3)$ opérations flottantes. Un des premiers à descendre sous cette complexité est dû à Strassen [573], avec une borne en $O(n^{2.807})$, mais il est numériquement instable [313 ; 2002, section 26.3.2]. Le schéma de Coppersmith et Winograd [139] est en $O(n^{2.495})$.

La méthode du GC a été proposée par Hestenes et Stiefel [312 ; 1952] pour la résolution de systèmes linéaires symétriques et reprise par Fletcher et Reeves [216 ; 1964] pour l'optimisation de fonctions quadratiques strictement convexes. L'effet des erreurs d'arrondi sur la vitesse de convergence est analysé par Greenbaum [279 ; 1989] et Notay [462 ; 1993] et sur la précision de la solution par Greenbaum et Strakoš [283 ; 1992], Greenbaum [280 ; 1994] et Greenbaum [282 ; 1997]. Van der Vorst [600 ; 1990] analyse l'effet des erreurs d'arrondi sur le gradient conjugué préconditionné par factorisation de Cholesky incomplète. La dérivée des itérés par rapport au second membre b est exhibée dans [276].

Andrei [14 ; 2020] propose une synthèse sur les algorithmes de gradient conjugué pour la minimisation sans contrainte de fonctions non linéaires.

La méthode GMRES a été principalement développée par Saad et Schultz [532 ; 1986]. La stabilité numérique de l'algorithme est analysée dans [180 ; 1995].

Pour en savoir plus sur les méthodes itératives de résolution de systèmes linéaires, on pourra consulter les livres ou synthèses de Greenbaum [281 ; 1997], Gutknecht [297 ; 1997], van der Vorst [602 ; 2003] et Saad [531 ; 2003]. Saad et van der Vorst [533, 601 ; 2000] donnent un apreçu historique intéressant du développement des méthodes itératives de résolution de systèmes linéaires au cours du XX-ième siècle.

Exercices

8.1. *Le GC comme méthode de Krylov.* Soient A une matrice d'ordre n, symétrique, définie positive et $b \in \mathbb{R}^n$. Soient d_1, d_2, ..., les directions générées par la méthode du GC pour minimiser $f(x) = \frac{1}{2}x^\mathsf{T}Ax - b^\mathsf{T}x$ et $E_p := \text{vect}\{d_1, d_2, \ldots, d_p\}$. On note $K_p := \text{vect}\{g_1, Ag_1, \ldots, A^{p-1}g_1\}$ le sous-espace de Krylov d'ordre p, associé à A et au gradient initial $g_1 = Ax_1 - b$. Montrez que $E_p = K_p$.

8.2. *Propriétés de monotonie dans l'algorithme du gradient conjugué.* Soient $\{x_k\}_{k=1}^p$ les itérés générés par le GC (donc $x_p = x_*$). On suppose que $p \geqslant 2$.

(*i*) Montrez que $\{\|x_k - x_1\|_2\}_{k=1}^p$ est strictement croissante et que $\{\|x_k - x_*\|_2\}_{k=1}^p$ est strictement décroissante.

(*ii*) Montrez que l'angle entre $x_2 - x_1$ et $x_k - x_1$ est strictement croissant.

8.3. *Le GC préconditionné.* On considère l'algorithme du gradient conjugué préconditionné de la section 8.2.5. Soit P la matrice symétrique définie positive utilisée pour préconditionner l'algorithme. Montrez que les propriétés suivantes ont lieu.

(*i*) C'est un algorithme à directions conjuguées dans lequel $\tilde{d}_k = -Pg_k$.

(*ii*) On a $\text{vect}\{d_1, \ldots, d_k\} = P\,\text{vect}\{g_1, \ldots, g_k\} = K_k(PA, Pg_1)$ (l'espace de Krylov associé à la matrice PA et au vecteur Pg_1).

(*iii*) On a $g_k^\mathsf{T}Pg_i = 0$ et $g_k^\mathsf{T}d_i = 0$, pour $1 \leqslant i \leqslant k-1$.

(*iv*) Montrez que si $P^{1/2}AP^{1/2} = \alpha I + E$, où $\alpha > 0$ et E est une matrice de rang m, alors le GC préconditionné converge en au plus m itérations.

8.4. *Algorithmes du GC réduit et du GC projeté.* On cherche à minimiser par GC une fonction quadratique strictement convexe $x \in \mathbb{R}^n \mapsto f(x) := \frac{1}{2}x^\mathsf{T}Ax - b^\mathsf{T}x$ sur un sous-espace affine $\mathcal{A}$ de dimension p de $\mathbb{R}^n$. Dans ce but, on se donne $x_1 \in \mathcal{A}$ et une matrice Z de type $n \times p$ orthogonale (c.-à-d., $Z^\mathsf{T}Z = I$) telle que tout point de $x \in \mathcal{A}$ puisse s'écrire sous la forme $x = x_1 + Zu$, avec $u \in \mathbb{R}^p$. On note $P = ZZ^\mathsf{T}$ le projecteur orthogonal sur $\mathcal{A}$. On considère deux formes de l'algorithme du GC pour résoudre ce problème.

(A1) L'*algorithme du GC réduit* consiste à minimiser par GC la *fonction réduite*

$$u \in \mathbb{R}^p \mapsto f^r(u) := \frac{1}{2}u^\mathsf{T}Z^\mathsf{T}AZu + (Ax_1 - b)^\mathsf{T}Zu,$$

obtenue en remplaçant x par $x_1 + Zu$ dans $f(x)$.

(A2) L'*algorithme du GC projeté* consiste à minimiser f dans $\mathbb{R}^n$ par l'algorithme du gradient conjugué préconditionné de la section 8.2.5 qui utilise le préconditionneur singulier P.

Montrez que les deux algorithmes sont identiques (dans un sens à préciser) si le premier est démarré en $u_1 = 0$ et le second en x_1.

8.5. *Algorithme du GC projeté préconditionné.* On se place dans le cadre défini à l'exercice 8.4, dans lequel on cherche à minimiser une fonction quadratique strictement convexe $x \in \mathbb{R}^n \mapsto f(x) := \frac{1}{2}x^\mathsf{T}Ax - b^\mathsf{T}x$ sur un sous-espace affine $\mathcal{A}$ de dimension p de $\mathbb{R}^n$. On note Z une matrice de type $n \times p$ orthogonale (c.-à-d., $Z^\mathsf{T}Z = I$) telle que tout point de $x \in \mathcal{A}$ puisse s'écrire sous la forme $x = x_1 + Zu$, avec $u \in \mathbb{R}^p$. Soit L une approximation de $A^{1/2}$ que l'on cherche à utiliser pour préconditionner le GC projeté. Cet algorithme s'obtient en faisant un changement de variable $\tilde{x} = Lx$, en appliquant le GC projeté dans l'espace des $\tilde{x}$, puis en revenant dans l'espace des x. Montrez que l'algorithme résultant est l'algorithme du gradient conjugué préconditionné de la section 8.2.5 qui utilise le préconditionneur singulier $P = Z(Z^\mathsf{T}L^\mathsf{T}LZ)^{-1}Z^\mathsf{T}$.

8.6. *Application de GMRES et comparaison avec le GC.* Utilisez l'algorithme GMRES pour trouver la solution x_* de $Ax = b$, en partant de $x_1 = 0$, lorsque

$$A = \begin{pmatrix} -1 & 2 & 0 \\ -2 & 3 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad \text{et} \quad b = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}.$$

Comparez le nombre d'itérations de cette méthode avec le nombre d'étapes que peut demander la méthode du GC pour calculer x_* comme solution de l'équation normale $A^\mathsf{T}Ax = A^\mathsf{T}b$.