MDI 210 - Optimisation

1 Problèmes de l'analyse numérique

On considère un système linéaire écrite sous la forme matricielle $Ax = b$.
Deux types d'erreurs peuvent être commises :

- Les erreurs d'arrondi, dues aux limites du codage employé.
- Les erreurs de troncature, due à la limite choisie en nombre d'iterations.

Conditionnement d'un système linéaire

Def. On appelle **conditionnement** de A (relativement à la norme $\|\cdot\|$, la quantité $\|A\| \cdot \|A^{-1}\|$, que l'on note $\mathrm{cond}_{\|\cdot\|}(A)$ ou $\mathrm{cond}(A)$.

Dans le cas $A(x + \delta x) = b + \delta b$ on a alors $\frac{\|\delta x\|}{\|x\|} \leqslant \mathrm{cond}(A)\frac{\|\delta b\|}{\|b\|}$. Dans le cas $(A + \delta A)(x + \delta x) = b$ alors $\frac{\|\delta x\|}{\|x+\delta x\|} \leqslant \mathrm{cond}(A)\frac{\|\delta A\|}{\|A\|}$.

Une matrice est d'autant mieux conditionnée que son conditionnement est proche de 1.

Th. *Soit A une matrice inversible. On a alors :*

- $\mathrm{cond}(A) \geqslant 1$
- $\mathrm{cond}(A) = \mathrm{cond}(A^{-1})$
- $\forall \alpha \neq 0, \mathrm{cond}(\alpha A) = \mathrm{cond}(A)$
- *En notant cond_2 le conditonnement associé à $\|\cdot\|_2$ et en notant respectivement $\mu_1(A)$ et $\mu_n(A)$ la plus petite et la plus grande des valeurs singulières de A,* $\mathrm{cond}_2(A) = \frac{\mu_n(A)}{\mu_1(A)}$.
- *Si A est normale (i.e. $AA^* = A^*A$),* $\mathrm{cond}_2(A) = \frac{\max_i|\lambda_i(A)|}{\min_i|\lambda_i(A)|}$ *où les $\lambda_i(A)$ représentent les valeurs propres de A.*
- *Si A est unitaire ou orthogonale,* $\mathrm{cond}_2(A) = 1$.
- $\mathrm{cond}_2(A)$ *est invariant par transformation unitaire ou orthogonale : si $UU^* = I$ (resp. $UU^\mathsf{T} = I$) alors*

$$\mathrm{cond}_2(A) = \mathrm{cond}_2(AU) = \mathrm{cond}_2(UA) = \mathrm{cond}_2(U^*AU) \qquad \textit{(resp. } \mathrm{cond}_2(U^\mathsf{T}AU)\textit{).}$$

Def. Le problème de **l'équilibrage d'une matrice** consiste à chercher à chercher deux matrices D_1 et D_2 diagonales inversibles telle que $\mathrm{cond}(D_1AD_2) = \inf_{\Delta_1,\Delta_2 \text{ diagonales inversibles}} \mathrm{cond}(\Delta_1A\Delta_2)$.

On résout alors $Ax = b$ en deux étapes : résolution de $D_1AD_2y = D_1b$ puis de $x = D_2y$.

En pratique on essaie plus simplement de minimiser le rapport entre le plus grand et le plus petit élément non nul de $A' = \Delta_1A\Delta_2$. Posons $E = \{(i, j) \in [\![1\,;n]\!]^2 \mid a'_{i,j} \neq 0\}$. On minimise $\frac{\max_{(i,j)\in E}|a'_{i,j}|}{\min_{(i,j)\in E}|a'_{i,j}|}$.

Conditionnement d'un problème de recherche de valeurs propres

Th. *Soit A diagonalisable et P une matrice telle que $P^{-1}AP = \mathrm{diag}(\lambda_i)$. Soit $\|\cdot\|$ une norme matricielle telle que, pour toute matrice diagonale $\mathrm{diag}(\alpha_i)$: $\|\mathrm{diag}(\alpha_i)\| = \max_i|\alpha_i|$. Alors, pour toute matrice δA,* $\mathrm{Sp}(A + \delta A) \subset \bigcup_{i=1}^n D_i$ *avec* $D_i = \{z \in \mathbf{C} \mid |z - \lambda_i| \leqslant \mathrm{cond}_{\|\cdot\|}(P) \cdot \|\delta A\|\}$.

Def. Le conditionnement $\Gamma(A)$ relatif à la recherche des valeurs propres est le minimum de $\mathrm{cond}_{\|\cdot\|}(P)$ pour P tel que $P^{-1}AP$ soit diagonale.

On a alors $\mathrm{Sp}(A + \delta A) \subset \bigcup_{i=1}^n B(\lambda_i, \Gamma(A) \cdot \|\delta A\|)$.

Th. *Soit A symétrique et $B = A + \delta A$ où la perturbation δA est également symétrique. Soit $\alpha_1 \leqslant \ldots \leqslant \alpha_n$ les valeurs propres de A et $\beta_1 \leqslant \ldots \leqslant \beta_n$ celles de B. Alors $\forall i \in [\![1\,;n]\!], |\alpha_i - \beta_i| \leqslant \|\delta A\|_2$.*

2 Résolution de systèmes linéaires

Problème : résoudre $Ax = b$ sachant A inversible.

Méthode de Gauss (pour une matrice quelconque)

Par combinaisons linéaires successives entre lignes de A on se ramène à $(MA)x = Mb$ où MA est triangulaire supérieure. On résout ensuite cette équation par une méthode de remontée.

Algorithme 1 : Étape d'élimination

Entrées : Matrice A.
$A' \leftarrow A$;
tant que $\dim(A') > 1$ **faire**
  `pivot` $\leftarrow a'_{i,1} \neq 0$ (qui existe car A' est inversible) ;
  si $i \neq 1$ **alors**
    $L_i \leftrightarrow L_1$;
  pour tous les $i > 1$ *tel que* $a_{i,1} \neq 0$ **faire**
    $L_i \leftarrow L_i - \frac{a_{i,1}}{a_{1,1}} L_1$;
    `// on annule tous les termes de la colonne du pivot situés sous la diagonale`
  `// notre matrice` A' `n'a alors que des 0 sous le premier terme, qui est non nul`
  `// on met alors de côté la première ligne et la première colonne de` A'
  $A' \leftarrow (a'_{i,j})_{2 \leqslant i,j}$
On replace les lignes et colonnes supprimées au fur et à mesure pour obtenir une matrice triangulaire (MA).

Un pivot trop petit en valeur absolue peut causer des erreurs d'arrondi du fait de la division par le pivot. D'où deux stratégies :

- *pivot partiel* : on prend le terme de plus grande valeur absolue de la colonne courante, sur ou en dessous de la diagonale,
- *pivot total* : on choisit le terme de plus grande valeur absolue de la matrice résiduelle, i.e. si on est à l'étape $n-k+1$, la matrice constituée des k dernières lignes et colonnes (plus couteux).

La complexité est en $O\left(\frac{2n^3}{3}\right)$ sans choix du pivot.

Méthode de Gauss-Jordan (variante de la précédente)

Dans la phase d'éliminations on élimine également les terme au-dessus de la diagonale. On obtient ainsi une matrice diagonale, ce qui permet de calculer efficacement l'inverse.

Factorisation LU

Th. *Soit* $A = (a_{i,j}) \in \mathcal{GL}_n(\mathbf{K})$ *telle que* $\forall k \in [\![1\,;n]\!], \begin{pmatrix} a_{11} & \dots & a_{1k} \\ \vdots & & \vdots \\ a_{k1} & \dots & a_{kk} \end{pmatrix} \in \mathcal{GL}_k(\mathbf{K})$. *Alors* A *admet une factorisation sous la forme* $A = LU$ *avec* L *triangulaire inférieure et* U *triangulaire supérieure. De plus on peut choisir* $\forall i \in [\![1\,;n]\!], (L)_{ii} = 1$ *et la décomposition est alors unique.*

Cela signifie que, dans l'algorithme de Gauss, les pivots successifs peuvent toujours être pris sur la diagonale. Si la factorisation échoue ou peut toujours permuter des lignes de A pour arriver à une matrice A' qui admet une factorisation LU.

Cette factorisation est utile lorsque plusieurs systèmes linéaires sont à résoudre : on résout un système sur L puis un système sur U.

Méthode de Cholesky (matrices symétriques définies positives)

Th. *Soit* $A \in \mathcal{S}_n^{++}(\mathbf{K})$. *Il existe une matrice triangulaire* B *vérifiant* $A = BB^\mathsf{T}$. *De plus on peut imposer que tous les éléments diagonaux de* B *soient tous strictement positifs et la factorisation* $A = BB^\mathsf{T}$ *est alors unique.*

En pratique, on calcule B colonne par colonne à partir de $\forall 1 \leqslant i \leqslant j \leqslant n, a_{ij} = \sum_{k=1}^{i} b_{ik} b_{jk} = a_{ji}$.

Rem. Le déterminant de la matrice peut alors se calculer facilement : $\det(A) = (b_{11} b_{22} \cdots b_{nn})^2$.

Un système $Ax = b$ devient alors $BB^\mathsf{T} x = b$. Pour résoudre le système on résout $By = b$ puis $B^\mathsf{T} x = y$. La complexité de la factorisation suivie de la résolution est en $O\left(\frac{n^3}{3}\right)$.

3 Valeurs propres et vecteurs propres : méthode de Jacobi

Def. Soit le polynôme $P(\lambda) = \lambda^n + a_1\lambda^{n-1} + \ldots + a_{n-1}\lambda + a_n$. Est dite « compagne du polynôme P » la matrice suivante :

$$C(P) = \begin{pmatrix} -a_1 & -a_2 & -a_3 & \ldots & \ldots & -a_{n-1} & -a_n \\ 1 & 0 & & & & & \\ 0 & 1 & 0 & & & & \\ & 0 & 1 & \ddots & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & 0 & 1 & 0 & \\ & & & & 0 & 1 & 0 \end{pmatrix}$$

Prop. *Le polynôme caractéristique de $C(P)$ vaut $(-1)^n P(\lambda)$. La matrice a donc pour valeurs propres les racines de P.*

Ce lien prouve, par le théorème d'Abel, que la recherche des valeurs propres d'une matrice ne peut se faire en un nombre fini d'opérations au-delà de la dimension 5.

Méthode de Jacobi Soit $A \in \mathcal{S}_n^+(\mathbf{R})$ non diagonale. On dispose de p et q, $p < q$ tels que $a_{pq} \neq 0$. On définit la matrice suivante :

$$\Omega = I_n + \sin(\theta) \cdot (E_{pq} - E_{qp}) + (1 - \cos(\theta)) \cdot (E_{pp} + E_{qq}) \,.$$

C'est la matrice de rotation d'angle $-\theta$ dans le plan défini par les p^{e} et q^{e} vecteurs de la base (donc orthogonale). On pose $B = \Omega^{\mathsf{T}} A \Omega \in \mathcal{S}_n(\mathbf{R})$, $c = \cos(\theta)$, $s = \sin(\theta)$ et $t = \tan(\theta)$. On a alors

$$\begin{cases} b_{ij} = b_{ji} = a_{ij} & \text{si } i \notin \{p,q\} \text{ et } j \notin \{p,q\} \\ b_{pi} = b_{ip} = c \cdot a_{pi} - s \cdot a_{qi} & \text{si } i \notin \{p,q\} \\ b_{qi} = b_{iq} = s \cdot a_{pi} + c \cdot a_{qi} & \text{si } i \notin \{p,q\} \\ b_{pp} = a_{pp} - t \cdot a_{pq} & \\ b_{qq} = a_{qq} + t \cdot a_{pq} & \end{cases} .$$

Th. *Soit $A \in \mathcal{S}_n^+(\mathbf{R})$ et B la matrice obtenue par la construction précédente. On a alors les relations*

$$\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}^2 = \sum_{i=1}^{n}\sum_{j=1}^{n} b_{ij}^2 \qquad \sum_{i=1}^{n} a_{ii}^2 + 2a_{pq}^2 = \sum_{i=1}^{n} b_{ii}^2$$

(le poids de la matrice se déporte sur la diagonale).

Th. *La suite des matrices obtenues par la méthode de Jacobi est convergente et converge vers une matrice diagonale contenant les valeurs propres de A.*

Pour accélérer la convergence il vaut mieux prendre $|a_{pq}|$ maximum.

Th. *Si toutes les valeurs propres de A sont distinctes, alors la suite des produits des matrices Ω converge vers une matrice orthogonale dont les vecteurs colonnes constituent unn ensemble orthnormal de vecteurs propres de A.*

4 Programmation linéaire : l'algorithme du simplexe

La forme standard d'un problème d'optimisation est la suivante :

- on maximise une forme linéaire $z = \sum_{j=1}^{n} c_j x_j$, la **fonction objectif**,
- avec m contraintes linéaires : $\forall i \in [\![1\,;m]\!], \sum_{j=1}^{n} a_{ij} x_j \leqslant b_i$,
- et n contraintes de positivité : $\forall j \in [\![1\,;n]\!], x_j \geqslant 0$.

Ces équations déterminent un **polyèdre des contraintes** qui est convexe. Les n-uplets $(x_1, \ldots, x_n)$ qui satisfont ces contraintes s'appellent **solutions réalisables** du problème. Ce sont les points intérieurs (au sens large) du polyèdre des contraintes. La **solution optimale** est celle qui maximise z.

S'il n'existe aucune solution réalisable, le problème est dit infaisable. S'il existe des solutions réalisables mais que z n'y admet pas de maximum fini alors le problème est dit non-borné.

Th. *Soit un problème de programmation linéaire dont le polyèdre des contraintes est non vide et dont la fonction à maximiser est majorée sur ce polyèdre. Alors le problème admet un maximum fini atteint en au moins un sommet du polyèdre est contraintes.*

Pour trouver le maximum on passe alors itérativement d'un sommet à un sommet adjacent de façon à augmenter la valeur de z.

À partir des données initiales on construit un *dictionnaire* de variables $x_1, \dots, x_{n+m}$ où $x_1, \dots, x_n$ sont appelées *varaibles de décision*, *varaibles de choix* ou encore *varaibles principales* ou *initiales* et $x_{n+1}, \dots, x_{n+m}$ (aussi positives) s'appellent *variables d'écart* telles que $x_{n+i} = b_i - \sum_{j=1}^{n} a_{ij} x_j$.

Un dictionnaire est un système d'équations linéaires liant $x_1, \dots, x_{n+m}$ et satisfaisant :

- les équations expriment z et m des $n+m$ variables (**variables de base**) en fonction des n autres variables (**variables hors-base**),
- équivalence avec le dictionnaire définissant les variables d'écart et la fonction objectif.

Pour une base fixée on obtient une **solution basique** en mettant à 0 toutes les variables hors base. Une solution est dite **dégénérée** lorsque des variables principales sont nulles.

Algorithme 2 : Itération de l'algorithme du simplexe

Entrées : Un dictionnaire :
$$I \uplus J = [\![1\,;n+m]\!], z = z^* + \sum_{j\in J} c_j x_j, \forall i \in I, x_i = b_i + \sum_{j\in J} a_{ij} x_j, b_i \geqslant 0$$

si $\forall j \in J, c_j \leqslant 0$ **alors**
  Fin de l'algorithme, on retourne z^* qui est une solution optimale.

sinon
  $j_0 \leftarrow \arg\max_j \{c_j, j \in J\}$ (choix autre que le maximum possible tant que supérieur à 0) ;
  `//` x_{j_0} `est la variable qui va rentrer en base`
  $i_0 \leftarrow \arg\min_i \left\{ \frac{-b_i}{a_{i,j_0}}, i \in I, \mid a_{i,j_0} < 0 \right\}$;
  `//` x_{i_0} `est la variable sortante`
  On extrait x_{j_0} de l'expression courante de x_{i_0} ;
  On remplace x_{j_0} par sa nouvelle expression dans z et dans l'expression des autres x_i ;
  On réitère avec le nouveau dictionnaire construit ;

Th (Théorème de Bland). *Il ne peut y avoir cyclage lorsque, à toute itération effectuée à partir d'un dictionnaire dégénéré, on choisit les variables entrante et sortante comme celles de plus petit indice parmi les candidats possibles.*

On obtient le **problème auxiliaire** en ajoutant une variable x_0 positive telle que $\forall i \in [\![1\,;m]\!], \sum_{j=1}^{n} a_{ij} x_j - x_0 \leqslant b_i$. Ce problème toujours réalisable en prenant x_0 assez grand et s'il n'était pas réalisable initialement (i.e. avec $x_0 = 0$) on peut chercher le x_0 minimum.

Méthode à deux phases : pour un dictionnaire non réalisable a priori on introduit x_0 et une fonction cible $w = -x_0$. On fait rentrer x_0 en base et on itère pour essayer de trouver un dictionnaire réalisable.

5 Dualité en programmation linéaire

S'il existe m réels $y_i \geqslant 0$ tels que $\forall j \in [\![1\,;n]\!], \sum_{i=1}^{m} a_{i,j} y_i \geqslant c_j$ alors on a, pour toute solution réalisable de (P) : $z \leqslant \sum_{i=1}^{m} b_i y_i$. Le **problème dual** (D) du problème primal (P) s'écrit : minimiser $\sum_{i=1}^{m} b_i y_i$ avec les contraintes $\forall j \in [\![1\,;n]\!], \sum_{i=1}^{m} a_{i,j} y_i \geqslant c_j$ et $\forall i \in [\![1\,;m]\!], y_i \geqslant 0$.

Rem. Le problème dual de (D) est (P).

Prop. *Soit $(x_1^*, \dots, x_n^*)$ une solution réalisable de (P) et $(y_1^*, \dots, y_m^*)$ une solution réalisable de (D). On a $\sum_{j=1}^{n} c_j x_j^* \leqslant \sum_{i=1}^{m} b_i y_i^*$. En cas d'égalité les deux solutions sont optimales pour leurs problèmes respectifs.*

Cor. *Si (P) admet une solution réalisable et est non bornée alors (D) est infaisable.*

Th (de la dualité). *Si (P) a une solution optimale $(x_1^*, \dots, x_n^*)$ alors (D) a une solution optimale $(y_1^*, \dots, y_m^*)$ et $\sum_{j=1}^{n} c_j x_j^* = \sum_{i=1}^{m} b_i y_i^*$.*

Prop. *Si (P) admet une solution primale telle que, dans son dernier dictionnaire obtenu par la méthode du simplexe, $z = z^* + \sum_{k=1}^{n+m} d_k x_k$ alors une solution optimale de (D) est donnée par $\forall i, y_i^* = -d_{n+i}$.*

Th (des écarts complémentaires). *Une solution $(x_1^*, \dots, x_n^*)$ de (P) est optimale si et seulement s'il existe $(y_1^*, \dots, y_m^*)$ une solution de (D) vérifiant :*

- $\forall i \in [\![1\,;m]\!], \left(\sum_{j=1}^n a_{ij}x_j^* < b\right) \implies (y_i^* = 0)$
- $\forall i \in [\![1\,;n]\!], \left(x_j^* > 0\right) \implies (\sum_{i=1}^m a_{ij}y_i^* = c_j)$

De plus $(y_1^*, \dots, y_m^*)$ constitue une solution optimale de (D).

Th. *Supposons que la base optimale de (P) est non dégénérée. Pour des variations δb_i des b_i on considère le problème (P_δ) avec les contraintes linéaires $\forall i \in [\![1\,;m]\!], \sum_{j=1}^n a_{ij}x_j \leqslant b_i + \delta b_i$. On suppose que les δb_i sont suffisamment faibles pour que la base optimale de (P) soit encore réalisable pour (P_δ). La variation de z vaut alors $\sum_{i=1}^m \delta b_i yi^*$ où $(y_1^*, \dots, y_m^*)$ est solution optimale de (D).*

L'utilisation du problème dual permet de résoudre un problème où la solution nulle n'est pas réalisable mais où $\forall j, c_j \leqslant 0$. Un tel problème est dit **dual-réalisable**.

6 Optimisation non linéaire sans contrainte

Def. Soit E et F des espaces vectoriels normés de dimensions finies et U un ouvert de E. Une fonction $f\colon U \to F$ est différentiable en $a \in U$ s'il existe une application linéaire $\mathrm{d}f(a)\colon E \to F$ appellée différentielle de f en a, et une fonction $r\colon U \to F$ continue et nulle en a telles que

$$\forall x \in U, f(x) = f(a) + \mathrm{d}f(a)\cdot(x-a) + r(x)\cdot\|x-a\| \ .$$

Def. On appelle matrice jacobienne de f en x relativement aux bases $\mathcal{B}$ et $\mathcal{B}'$ la matrice $\mathrm{Jac}_{f;\mathcal{B},\mathcal{B}'}(x) = \mathrm{Mat}(\mathrm{d}f(x); \mathcal{B}, \mathcal{B}')$.

Prenons maintenant $f\colon \mathbf{R}^n \to \mathbf{R}$. On cherche à déterminer ses extrema, locaux ou globaux.

Def. Soit $x \in \mathbf{R}^n$ tel que f est dérivable en x. Le **gradient** de f en x est $\nabla f(x) = \left(\frac{\delta f}{\delta x_1}(x), \dots, \frac{\delta f}{\delta x_n}(x)\right)^\mathsf{T}$.

Prop. *Soit $A \in \mathfrak{M}_n(\mathbf{R})$ et u et v deux fonctions de $\mathbf{R}^n$. Alors $\mathrm{d}u^\mathsf{T}A) = \mathrm{d}(u^\mathsf{T})A$ et $\nabla(\langle u \mid v\rangle) = \langle \mathrm{d}u \mid v\rangle + \langle u \mid \mathrm{d}v\rangle$.*

Def. Si f admet des dérivées partielles d'ordre en x, on a sa **matrice hessienne** dans $\mathcal{B}$ la base canonique, $\mathcal{H}f(x) = \mathrm{Jac}_{f;\mathcal{B}}(x) = \left(\frac{\partial^2 f}{\partial x_i \partial x_j}(x)\right)_{1\leqslant i,j\leqslant n}$. Si f est de classe $\mathcal{C}^2$, la matrice hessienne est symétrique.

Prop. *Si f est de classe $\mathcal{C}^2$ en a, on peut écrire la formule de Taylor d'ordre 2 :*

$$f(a+h) = f(a) + \langle \nabla f(a) \mid h\rangle + \frac{1}{2}\langle \mathcal{H}f(a)\cdot h \mid h\rangle + o\left(\|h\|^2\right) \ .$$

Th (condition nécessaire d'optimalité). *Si f admet un minimum local en x^*, alors $\nabla f(x^*) = 0$ et $\mathcal{H}f(x^*)$ est une matrice positive (i.e. $\forall h \in \mathbf{R}^n, h^\mathsf{T}\mathcal{H}f(x^*)h \geqslant 0$).*

Th (condition suffisante d'optimalité). *Si f vérifie en x^* $\nabla f(x^*) = 0$ et $\mathcal{H}f(x^*)$ est une matrice* définie *positive (i.e. $\forall h \in \mathbf{R}^n \setminus \{0\}, h^\mathsf{T}\mathcal{H}f(x^*)h > 0$).*

Def. Soit $A \in \mathcal{S}_n(\mathbf{R})$, $b \in \mathbf{R}^n$ et $c \in \mathbf{R}$. Une application $q\colon \mathbf{R}^n \to \mathbf{R}$ de la forme $x \mapsto c + \langle x \mid c\rangle + \frac{1}{2}\langle Ax \mid x\rangle$ est appellée **fonction quadratique**.

Pour cette application quadratique on a $\nabla q(x) = b + Ax$ et $\mathcal{H}q(x) = A$.

Fonctions convexes

Th. *Si f est convexe et admet des dérivées partielles, alors f admet un minimum global en x^* ssi $\nabla f(x^*) = 0$.*

Th. *Si f est convexe et admet un minimum local en x^*, alors c'est un minimum global.*

Th. *Si f est de classe $\mathcal{C}^2$ alors les propositions suivantes sont équivalentes :*

(a) f est convexe,

(b) $\forall a, h \in \mathbf{R}^n, f(a+h) \geqslant f(a) + \langle \nabla f(a) \mid h\rangle$ (la surface de $\mathbf{R}^{n+1}$ d'équation $x_{n+1} = f(x)$ est au-dessus de ses hyperplans tangents),

(c) pour tout $x \in \mathbf{R}^n$, $\mathcal{H}f(x)$ est positive.

Généralités sur les méthodes d'optimisation sans contrainte

Pour déterminer un point où f atteint un minimum local, on construit une suité (x_i) qui doit converger vers un point x^* vérifiant une condition nécessaire d'optimalité. On appelle **méthode de descente** toute méthode telle que $\forall k, x_{k+1} = x_k + s_k d_k$ où $s_k \in \mathbf{R}_+$ et $d_k \in \mathbf{R}^n$ est une direction qui vérifie $\langle d_k \mid \nabla f(x_k) \rangle < 0$. On veut ainsi assurer au minimum $f(x_{k+1}) \leqslant f(x_k)$.

Lorsque la convergence d'un algorithme est établie, on s'intéresse à sa *vitesse de convergence* :

- Si $\frac{\|x_{k+1}-x^*\|}{\|x_k-x^*\|} \leqslant \alpha < 1$ pour k assez grand, on dit que la convergence est *linéaire* de taux α.
- Si $\frac{\|x_{k+1}-x^*\|}{\|x_k-x^*\|} \overset{k\to\infty}{\to} 0$ on dit que la convergence est *superlinéaire*.
- Si $\frac{\|x_{k+1}-x^*\|}{\|x_k-x^*\|^\gamma}$ est borné avec $\gamma > 1$, on dit que la convergence est superlinéaire d'ordre γ. Dans le cas $\gamma = 2$ elle est dite quadratique.

Méthodes de gradient

Soit $x_k \in \mathbf{R}^n$ tel que $\nabla f(x_k) \neq 0$ et $d \in \mathbf{R}^n$. Posons, pour $s \in \mathbf{R}$, $g(s) = f(x_k + sd)$. On dit que d est une *direction de descente* si $g'(0) = \langle d \mid \nabla f(x_k) \rangle < 0$.

La *direction de plus grande pente descendante* est donnée par $d = -\frac{\nabla f(x_k)}{\|\nabla f(x_k)\|}$.

Algorithme 3 : Méthode de gradient

Entrées : Fonction f à minimser.
On choisit un point de départ x_0 ;
$k \leftarrow 0$;
tant que *un test d'arrêt n'est pas vérifié* **faire**
  $d_k \leftarrow -\nabla f(x_k)$;
  $s_k \leftarrow \arg\min_{s \geqslant 0} f(s_k + sd_k)$;
  $x_{k+1} \leftarrow x_k + s_k d_k$;
  $k \leftarrow k + 1$;
Sorties : x_k

La condition d'arrêt peut être $k > N$, $\|\nabla f(x_k)\|_2 < \epsilon$ ou $|f(x_{k+1}) - f(x_k)| \leqslant \epsilon$. Les directions successives empruntées sont ici orthogonales.

Méthode de la plus forte pente accélérée

Soit $p \in \mathbf{N}$. À partir d'un point x_k on effectue p itérations de la méthode de la plus forte pente ; on obtient un point y_k et on pose $d_k = y_k - x_k$. Le point x_{k+1} est le point où $f(x_k + sd_k)$ atteint un minimum pour $s > 0$.

7 Optimisation non linéaire avec contraintes