Algorithmes de descente

1) Algorithmique de optimisation

Considérons le problème sans contrainte

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

où $f : \mathbb{R}^n \longrightarrow \mathbb{R}$

Remarques

(1) Si l'on veut minimiser un polynôme $P : \mathbb{R} \to \mathbb{R}$, une première approche consistant à calculer les racines de P'

Si degré $P \geq 6 \Rightarrow$ degré $P' \geq 5$

$\Rightarrow$ pas de formule par radicaux

- On s'intéresse donc ici à des méthodes approchées qui génèrent des suites $(x_n) \to$ solution
- Ces algorithmes requièrent une analyse de convergence des suites qu'ils génèrent

(2) Autres limitations : les algorithmes proposés ne pourront trouver que des minima locaux et même des points stationnaires ($\nabla f(x) = 0$)

$\to$ trouver un minimum global est proche d'un problème d'optimisation combinatoire (pensez à la minimisation d'un polynôme d'ordre 1000 sur $\mathbb{R}^{1000}$!!)

$\to$ pour la minimisation globale de polynômes, il existe des approches, mais gourmandes en place mémoire (relaxation de Lasserre)

(3) Si x_* est un min local de (P), on a

$$\nabla f(x_*) = 0 . \qquad (1)$$

C'est un système de n equations à n inconnues. On connaît des algorithmes pour résoudre ces problèmes. Très souvent, les algorithmes qui résolvent (1) trouvent des min. locaux ; mais ce n'est pas garanti.

(4) Il y a une différence importante entre (1) et la recherche d'un zéro de $F : \mathbb{R}^n \to \mathbb{R}^n$ c-à-d. trouver x tel que

$$F(x) = 0 .$$

Ici $F = \nabla f$ et donc $F'(x) = \nabla^2 f(x)$ est symétrique $\Rightarrow$ plus facile.

2) Algorithmes à directions de descente

[A] Principes algorithmiques

Déf d est une direction de descente de f en x si

$$\boxed{f'(x).d = \langle \nabla f(x), d \rangle < 0}$$

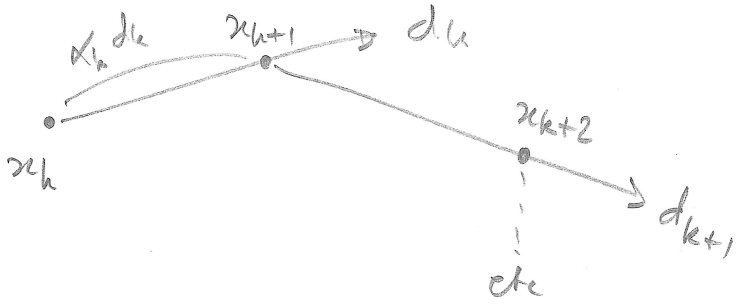
Dans ce cas, $f(x + \alpha d) < f(x)$ pour tout $\alpha > 0$ petit. En effet

$$0 > f'(x).d = \lim_{\alpha \downarrow 0} \frac{1}{\alpha} \underbrace{\left[f(x+\alpha d) - f(x) \right]}_{\text{donc strictement négatif si } \alpha > 0 \text{ petit}}$$

Algorithme à direction de descente

On passe de x_k à x_{k+1} comme suit

1) [test d'arrêt] arrêt si x_k convient.
2) choix (et calcul !) d'une direction de descente d_k
3) [Recherche linéaire] Calcul d'un pas $\alpha_k > 0$ pas trop petit tel que
$$f(x_k + \alpha_k d_k) \leq f(x_k) + (\text{terme négatif})$$
4) [nouvel itéré] $x_{k+1} = x_k + \alpha_k d_k$



Remarques

1) Le test d'arrêt en x_k dépendra de l'algorithme. Si on utilise que des directions de descente comme dans le schéma algorithmique ci-dessus, on ne peut espérer mieux que $\nabla f(x) = 0$; donc on prend comme test d'arrêt

$$\nabla f(x_k) \simeq 0$$

2) On verra à la section suivante quelques exemples de directions de descente

3) Voir la section 5 pour la recherche linéaire

B Exemples de directions de descente

(la direction donne son nom à l'algorithme)

(1) Algorithme de la plus profonde descente ou du gradient

$\boxed{d_k = -\nabla f(x_k)}$ On note $g_k := \nabla f(x_k)$

Descente : $f'(x_k) \cdot d_k = -\|\nabla f(x_k)\|^2 < 0$ si $\nabla f(x_k) \neq 0$

$= \langle \nabla f(x_k), -\nabla f(x_k) \rangle$

Rmq : algorithme du 1er ordre, lent en pratique, mais avec un résultat de convergence global fort ; redevenu intéressant pour les données massives.

(2) Algorithme du gradient conjugué

$\boxed{d_k = -g_k + \beta_k d_{k-1}}$ si $k \geq 2$ ($\beta_k = 0$ si $k = 1$)

Descente : $f'(x_k) \cdot d_k = -\|g_k\|^2 + \beta_k \underbrace{\langle g_k, d_{k-1} \rangle}_{=0} < 0$

si $g_k \neq 0$ et Recherche linéaire exacte

(si on minimise $\alpha \longmapsto f(x_{k-1} + \alpha d_{k-1})$ pour trouver α_{k-1})

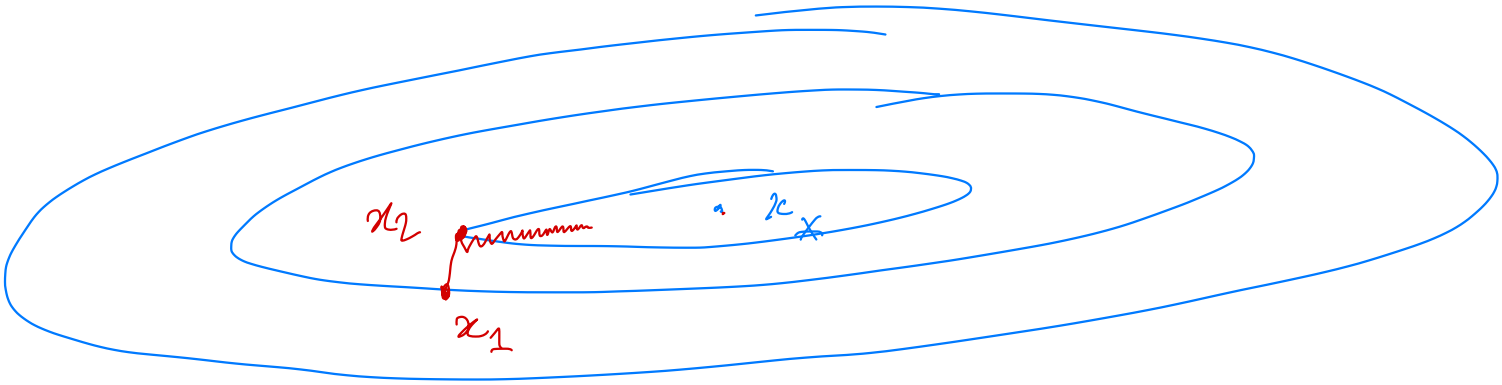
→ OK si problème quadratique sinon très mauvais.

Rmq : - beaucoup de contributions sur des variantes de cet algorithme depuis ~1990 (peut-être à cause des données massives).

- algorithmes très peu gourmands en temps de calcul interne et en place mémoire.

$\alpha_{k-1} \in \operatorname{argmin}_{\alpha > 0} f(x_{k-1} + \alpha d_{k-1})$

$\Rightarrow \langle \nabla f(x_k), d_{k-1} \rangle = 0$



Différentiation automatique (mode inverse)

$$\frac{T(f(x), \nabla f(x))}{T(f(x))} \leq 5$$

(3) Algorithme de Newton

$$\boxed{d_k = -\nabla^2 f(x_k)^{-1} g_k} \quad \text{(si possible)}$$

Descente : $f'(x_k) \cdot d_k = -\langle \nabla^2 f(x_k)^{-1} g_k, g_k \rangle < 0$

si $g_k \neq 0$ et $\nabla^2 f(x_k) > 0$

Remq : - algorithme très rapide mais coûteux (il faut calculer $\nabla^2 f(x_k)$ et résoudre un système linéaire),
- convergence locale seulement,
- algorithme très étudié (pour beaucoup de problèmes différents).

(4) Algorithme de quasi-Newton

$\boxed{d_k = -M_k^{-1} g_k}$ où $M_k \simeq \nabla^2 f(x_k)$ est aussi généré par l'algorithme

Descente si $g_k \neq 0$ et $M_k > 0$

Remq - très bonne famille d'algorithmes,
- moins coûteux que Newton,
- il existe des versions pour très grands problèmes ($n \simeq 10^8 \ldots$).

(5) Algorithme de Gauss-Newton

Adapté aux problèmes de moindres-carrés, c-à-d. qui s'écrivent

$$\inf_{x \in \mathbb{R}^n} f(x) \quad \text{où} \quad f(x) = \frac{1}{2} \| r(x) \|_2^2$$

$$r : \mathbb{R}^n \to \mathbb{R}^m \quad (\text{résidu})$$

Si on note $J_k := r'(x_k)$ $(m \times n)$

$r_k := r(x_k)$,

on calcule d_k comme solution de

$$\boxed{(J_k^T J_k)\, d_k = - J_k^T r_k}$$

($\exists$ toujours une solution)

$J_k^T J_k$: une approximation de $\nabla^2 f(x_k)$

$-J_k^T r_k$ = $-g_k$

$f'(x) d = \langle r(x), r'(x) d \rangle$

$\Rightarrow \nabla f(x) = r'(x)^T r(x)$

Descente : $f'(x_k) \cdot d_k = - \| J_k d_k \|_2^2 < 0$

si $g_k \neq 0$

Rmq : - très utilisé pour les problèmes de moindres-carrés car il ne faut pas calculer $r''(x_k)$ (contrairement à l'algorithme de Newton),

- convergence rapide si $r(x_*) \simeq 0$ ou $r''(x_*) \simeq 0$.

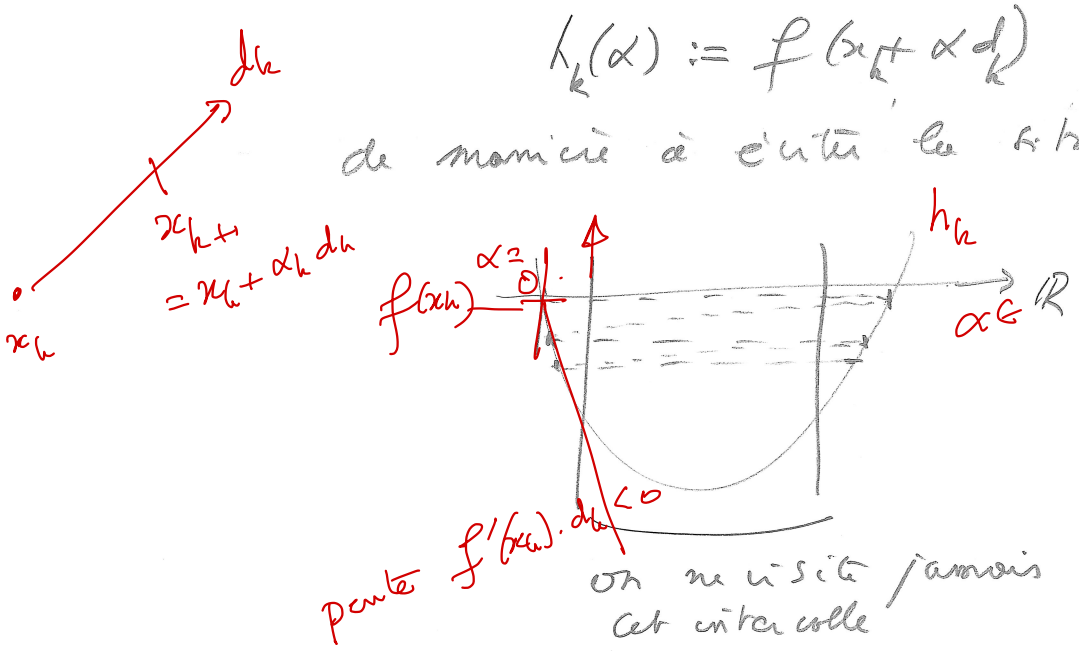
C Règles de recherche linéaire

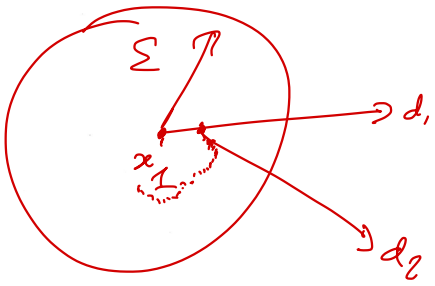
On cherche un pas $\alpha_k > 0$ en x_k avec 2 objectifs

— faire décroître suffisamment $h_k : \mathbb{R}_+ \to \mathbb{R}$

$$h_k(\alpha) := f(x_k + \alpha d_k)$$

de manière à éviter les situations





— ne pas prendre $\alpha_k > 0$ trop petit

si $k \geq 1$ $\begin{cases} 0 < \alpha_k \leq \dfrac{\varepsilon}{2^k \|d_k\|} \\ f(x_k + \alpha_k d_k) < f(x_k) \end{cases}$ toujours possible

on a (x_k) est une suite de Cauchy car

$$x_k - x_\ell = \sum_{i=\ell}^{k-1} \alpha_i d_i$$

$$\|x_k - x_\ell\| \leq \sum_{i=\ell}^{k-1} \alpha_i \|d_i\| \leq$$

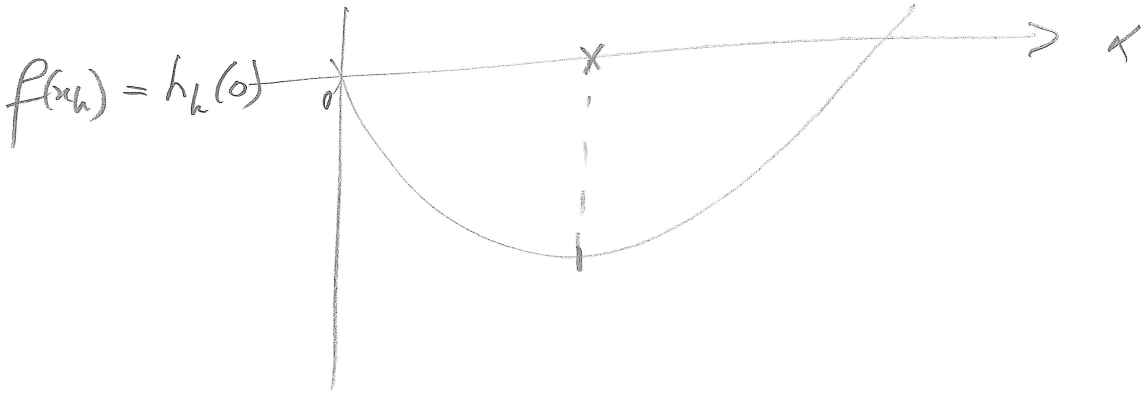
$$\leq \varepsilon \sum_{i \geq \ell} \frac{1}{2^i} = \frac{\varepsilon}{2^{\ell-1}} \to 0 \text{ si } \ell \to \infty$$

Donc $x_k \to \bar{x} \in \bar{B}(x_1, \varepsilon)$

si x_1 n'est pas solution et $\varepsilon > 0$ petit, $\bar{x}$ n'est pas solution ($\Rightarrow$ fausse convergence).

① Règle de Cauchy (ou recherche linéaire exacte)

$$\alpha_k \in \underset{\alpha \in \mathbb{R}_+}{\operatorname{argmin}} \; h_k(\alpha)$$



Rmq

- ça semble très bien, mais c'est à proscrire car
  - ⊗ c'est un problème de minimisation en soi ⇒ demande beaucoup d'itérations donc coûteux (en calcul de gradients)
  - ⊗ il faut un test d'arrêt (minimisation exacte impossible) ; donc mieux vaut avoir un test d'arrêt qui assure la convergence globale de l'algorithme sans devoir faire une minimisation précise.
  - ⊗ le minimum peut ne pas exister
- à réserver pour les problèmes quadratiques

② Règle d'Armijo

- on fait décroître f suffisamment en demandant

$$f(x_k + \alpha_k d_k) \leq f(x_k) + \omega \alpha_k \underbrace{\langle g_k, d_k \rangle}_{<0} \qquad (A)$$

toujours possible

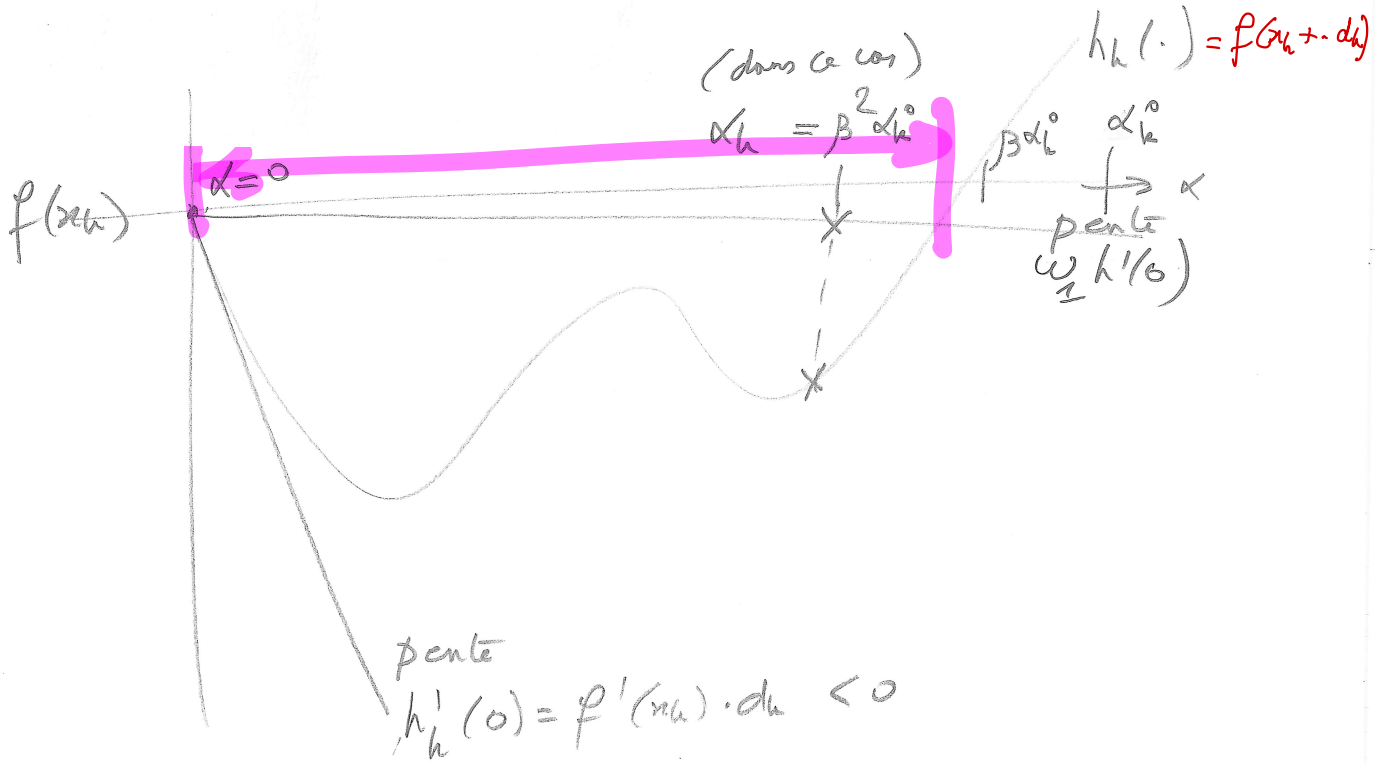
$\omega \in]0, \frac{1}{2}]$ (typiquement $\omega \simeq 10^{-4}$)

c'est la forme négative

- on empêche le pas d'être trop petit en prenant

$$\alpha_k = \beta^{i_k} (\alpha_k^0) \longrightarrow \in \text{compact de } \mathbb{R}_{++} \text{ (souvent } \alpha_k^0 = 1)$$

où i_k est le plus petit entier ≥ 0 tel que (A) soit vérifié.



Rmq – souvent utilisé pour l'algorithme de Newton

(3) Règle de Wolfe

- on fait décroître f suffisamment en demandant (comme dans la règle d'Armijo)

$$f(x_k + \alpha_k d_k) \leq f(x_k) + \omega_1 \alpha_k \langle g_k, d_k \rangle \qquad (W_1)$$

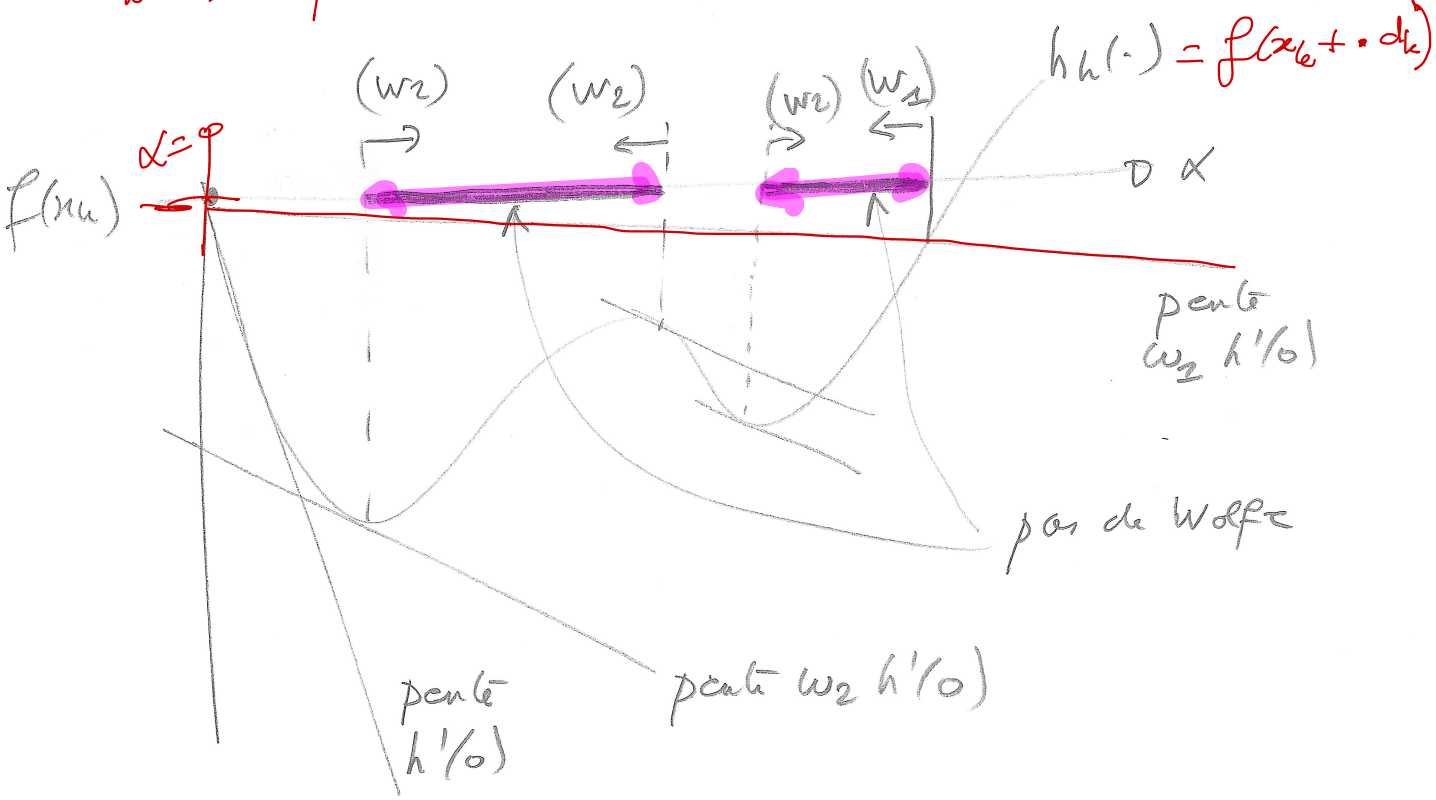
$\omega_1 \in]0, \frac{1}{2}]$ (typiquement $\omega_1 = 10^{-4}$)

- on empêche le pas d'être trop petit en demandant

$$\langle g(x_k + \alpha_k d_k), d_k \rangle \geq \omega_2 \langle g_k, d_k \rangle \qquad (W_2)$$

$h_k'(\alpha_k) = f'(x_k + \alpha_k d) \cdot d_k$; $h_k'(0)$

$\omega_2 \in]\omega_1, 1[$ (typiquement $\omega_2 = 0.99$)



Rmq – très utilisé pour l'algorithme de quasi-Newton (car permet d'assurer $H_k \succ 0$)

D Un résultat de convergence globale

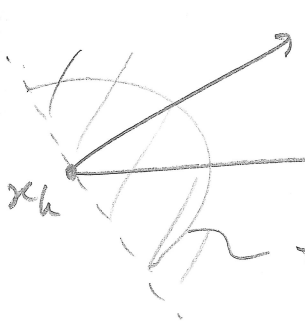
Si • algorithme à directions de descente et recherche linéaire de Wolfe,
• $\{f(x_k)\}$ bornée inférieurement,
• f est $C^{1,1}$, c.-à-d., $\exists L>0$: $\forall x,y \in \mathbb{R}^n$ $\|\nabla f(y)-\nabla f(x)\| \le L\|y-x\|$

alors on a la condition de Zoutendijk

$$\sum_{k\ge 1} \|g_k\|^2 \cos^2\theta_k < +\infty$$

On a noté θ_k = l'angle entre d_k et $-g_k$:

$$\cos\theta_k = -\frac{\langle g_k, d_k\rangle}{\|g_k\|\,\|d_k\|}$$



toutes les directions de descente de f en x_k (pour le p.s. euclidien ici)

Exemples d'utilisation

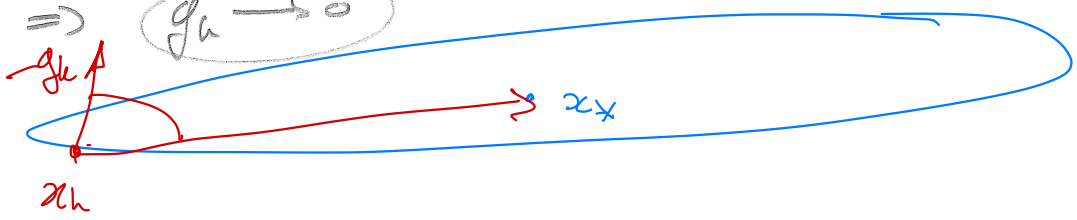
• algorithme du gradient $\Rightarrow d_k = -g_k \Rightarrow \theta_k = 0$

$\Rightarrow \sum_{k\ge 1} \|g_k\|^2 < \infty \Rightarrow$ $g_k \to 0$

(on ne peut pas dire beaucoup plus que ça !)

• plus généralement, si $\cos\theta_k \ge \gamma > 0$

$\Rightarrow$ $g_k \to 0$



Dem