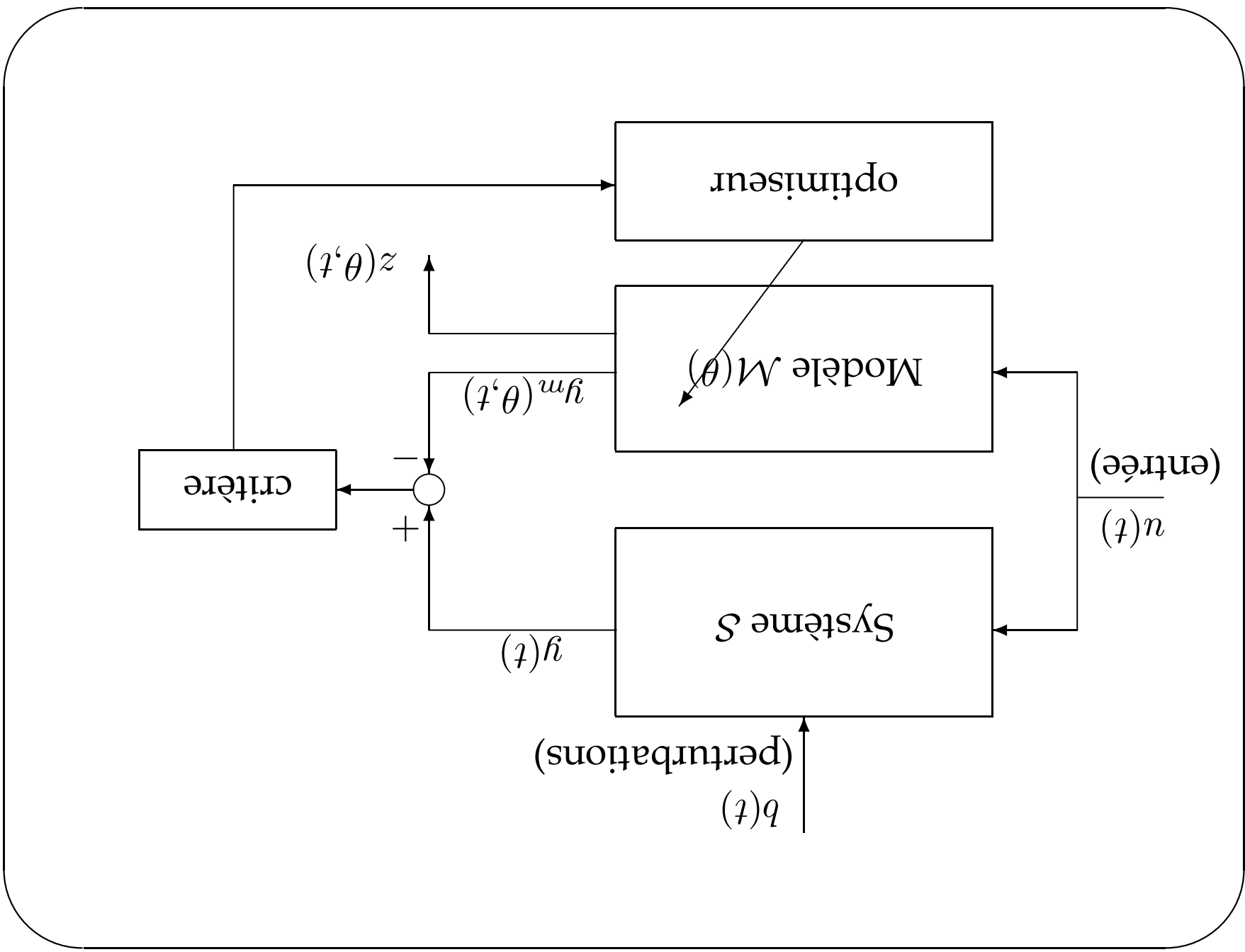


# OPTIMISATION — MÉTHODES GÉNÉRALES

Luc Pronzato, 2004

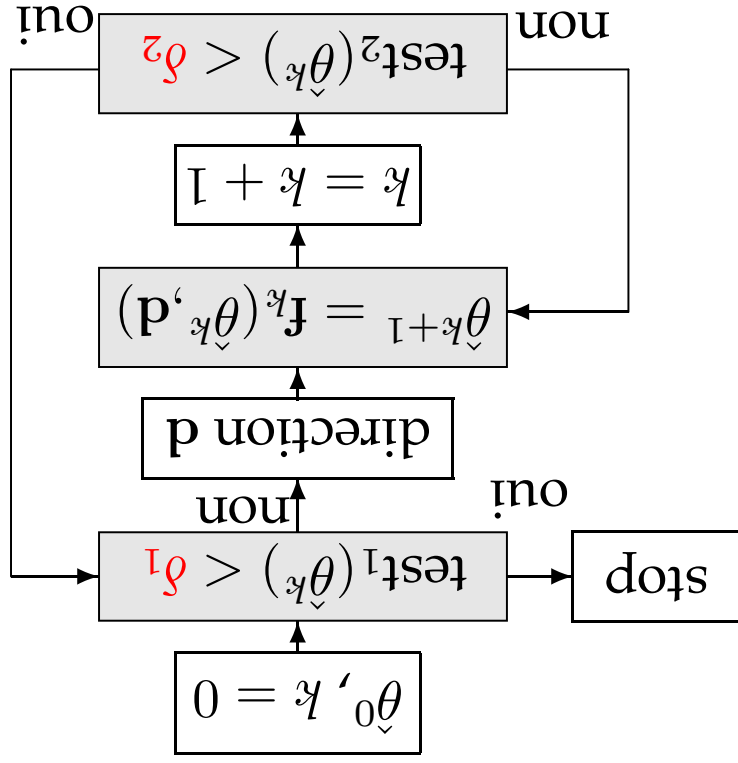
- 1 Optimisation à une dimension
- 2 Combinaison de recherches à une dimension
- 3 Gradient
- 4 Newton & Gauss-Newton
- 5 Quasi-Newton, gradients conjugués
- 6 Optimisation à une dimension (2) : le retour
- 7 Optimisation sous contraintes
- 8 Critères non différentiables
- 9 Techniques récursives
- 10 Optimisation globale





## 0) Généralités

Toujours critère  $j(\theta)$  à minimiser (sinon,  $j(\theta) \leftarrow -j(\theta)$ )



Difficulté : réglage de  $\delta_1$  et  $\delta_2$ !

# 1) Optimisation à une dimension

$$j(\theta), p = \dim(\theta) = 1$$

1.1) Définir un intervalle de recherche :

Au départ  $\hat{\theta}_0$ , calculer la dérivée  $j'(\hat{\theta}_0)$ . Si  $j'(\hat{\theta}_0) < 0$ , calculer  $\hat{\theta}_k = \hat{\theta}_0 + k\Delta$ ,  $\Delta > 0$ , jusqu'à ce que  $j(\hat{\theta}_k) > j(\hat{\theta}_0)$ .  
←  $\exists$  un minimum entre  $\hat{\theta}^{k-2}$  et  $\hat{\theta}^k$

Si  $j'(\hat{\theta}_0) > 0$ , même chose avec  $\Delta < 0$

**Rq :** Direction opposée au signe de la dérivée... voir + loin

1.2) Réduire sa longueur : On part de  $\mathcal{I}^0 = [a_0, b_0]$ ,  $k = 0$

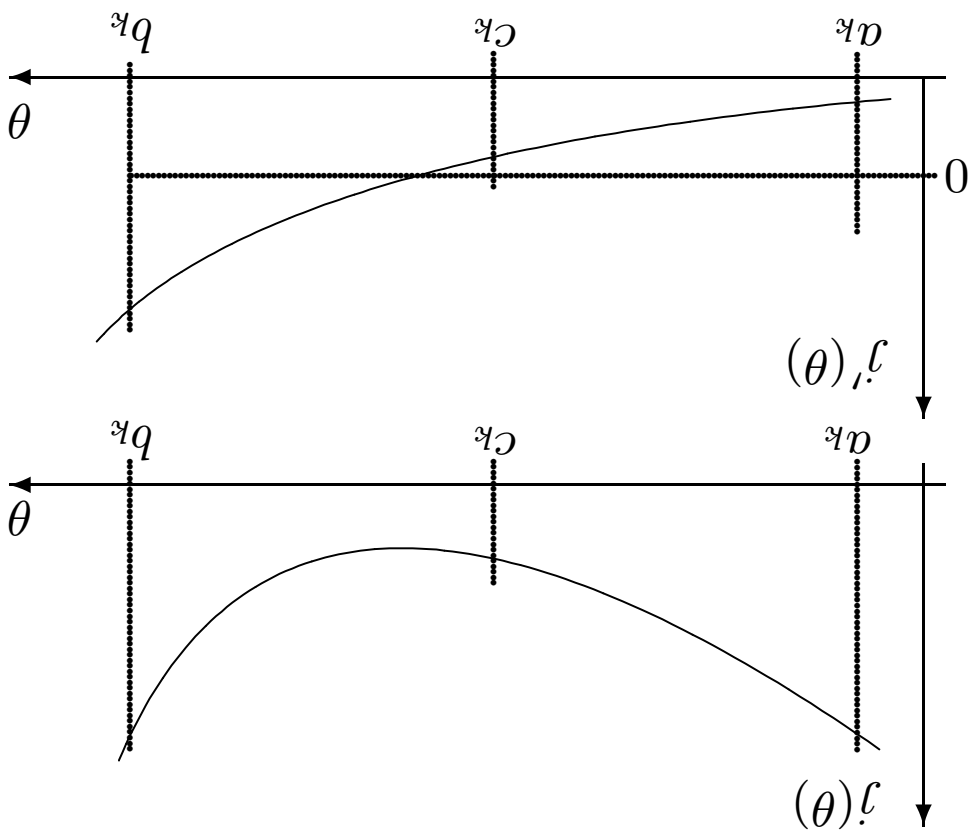
**Dichotomie :** On évalue la dérivée au centre  $c_k = (a_k + b_k)/2$ ,

$$\mathcal{I}^{k+1} = \begin{cases} [a_k, c_k] & \text{si } j'(c_k) > 0 \\ [c_k, b_k] & \text{sinon} \end{cases}$$

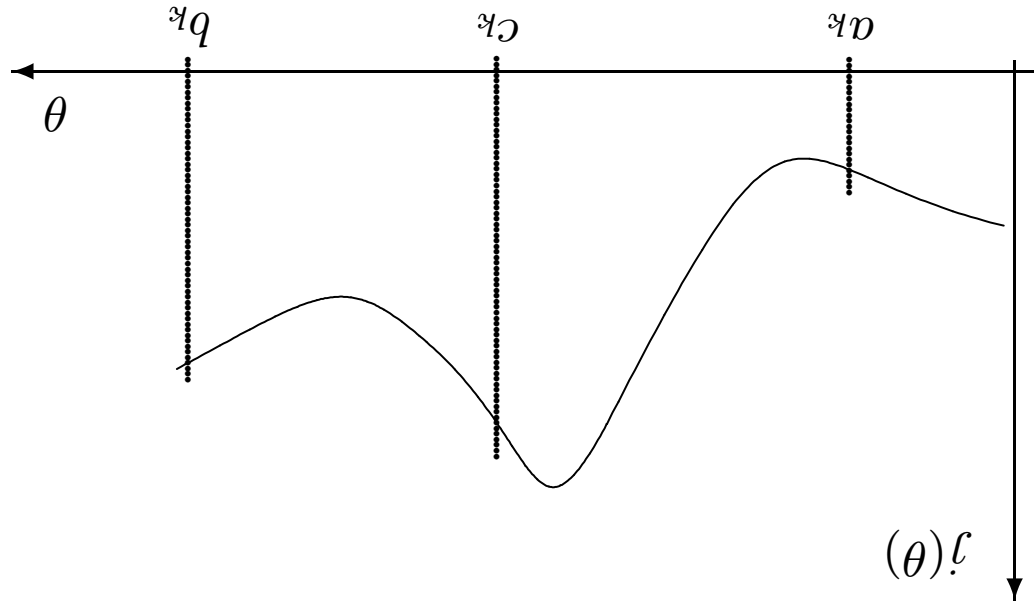
← très rapide !

$$L_n = b_n - a_n = \frac{2^n}{L_0}$$

$n$  calculs de dérivée  $j'(\cdot)$  :



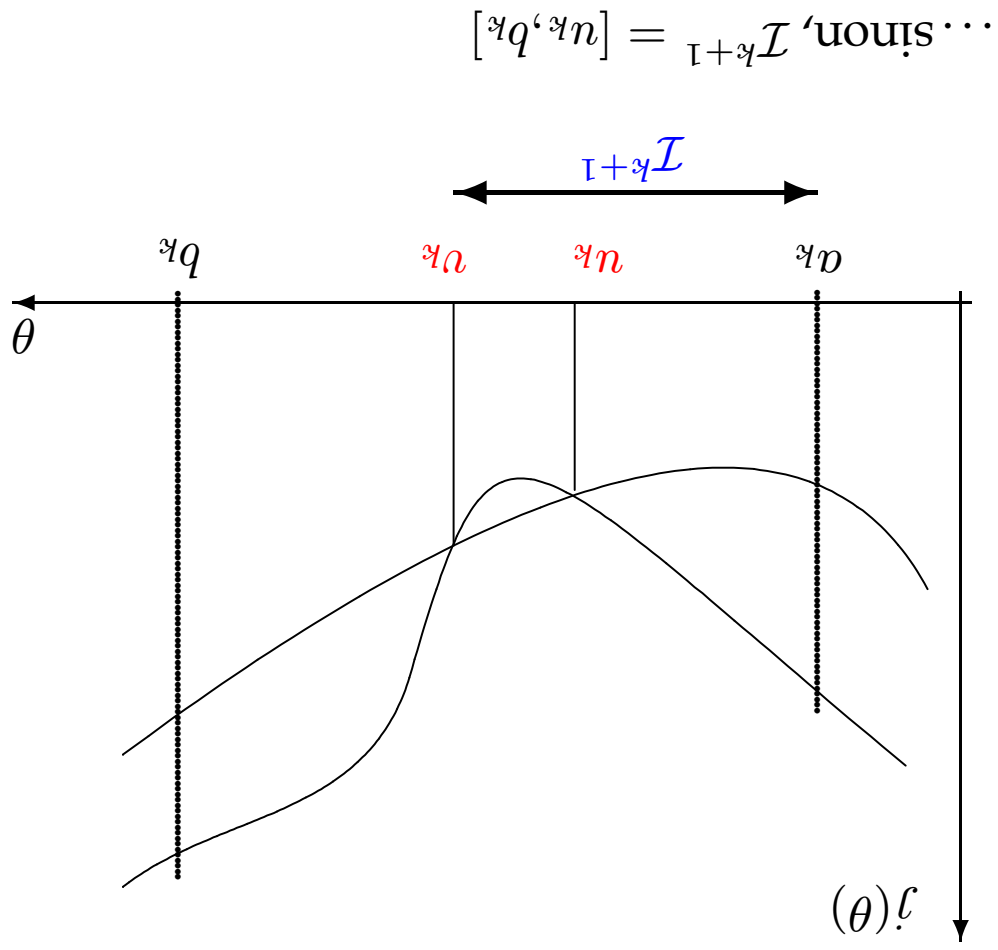
Fibonacci et section-dorée : on n'évalue pas de dérivée



Mais attention : peut converger vers un minimum local

→ 2 calculs de critère, en  $u_k, v_k, (u_k > v_k)$  pour un calcul de dérivée

Si  $j(u_k) > j(v_k), \mathcal{I}_{k+1} = [a_k, v_k]$

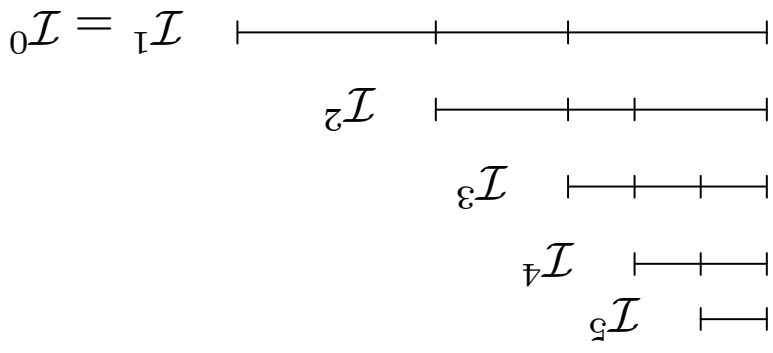


## Méthode de Fibonacci (Kiefer, 1953) : optimale dans le pire cas

Comment choisir  $u_k, v_k$  ?

$\mathcal{I}_{k+1}$  contient  $u_k$  ou  $v_k$  !  
 ← une seule évaluation de  $j(\cdot)$  pour passer à  $\mathcal{I}_{k+2}$

On part de  $\mathcal{I}_0 = [a_0, b_0]$ ,  $L_0 = b_0 - a_0$ ,  $k = 0$   
 Le nombre d'évaluations de  $j(\cdot)$  autorisées est fixé :  $N$   
 $N = 1 : L_1 = L_0$   
 $N = 2 : 2$  évaluations en  $c_0 - \epsilon, c_0 + \epsilon$ , avec  $c_0 = (a_0 + b_0)/2 \rightarrow L_1 \approx L_0/2$   
 ensuite... programmation dynamique  
 pire cas  $\Rightarrow$  symétrie





$$\rightarrow L_{k-1} = L_k + L_{k+1}$$

Dans  $\mathcal{I}^k$ , 2 évaluations en  $a_k + z_k L_k, a_k + (1 - z_k) L_k$ , avec  $z_k = L_{k+1}/L_k$

Finalement

$$z_N = z_{N-1} = 1/2, z_{N-2} = 2/3, z_{N-3} = 3/5, z_{N-4} = 5/8 \dots$$

et

$$L_{N-k} = F_{k+2} L_N$$

avec  $(F_i)$  la suite de Fibonacci:  $F_0 = F_1 = 1, F_k = F_{k-1} + F_{k-2}, k \geq 2$

Mais les points d'évaluation dépendent de la valeur  $N$  choisie...

→ Section dorée

représentation d'état:  $\mathbf{x}_k = (F_k, F_{k+1})^\top$

$$\mathbf{x}_{k+1} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \mathbf{x}_k, \mathbf{x}_0 = (0, 1)^\top$$

→ 2 valeurs propres  $1 + \alpha, -\alpha$ , avec  $\alpha = (\sqrt{5} - 1)/2$  le nombre d'or

Pour  $k \rightarrow \infty$  : le mode instable domine

$$\frac{L_0}{L_N} \approx \frac{L_0}{(1 + \alpha)^{N+1} \sqrt{5}}$$

et  $\lim_{k \rightarrow \infty} z_k = 1/(1 + \alpha) = \alpha$

→ méthode de la section dorée :

Dans  $\mathcal{I}_k$ , 2 évaluations en  $a_k + \alpha L_k$  et  $a_k + (1 - \alpha)L_k$

après  $N$  évaluations

$$\frac{L_0}{L_N} \approx \frac{1}{1 + \alpha^{N-1}} = (1 + \alpha)^{N-1}$$

**Rq :** On peut aller + vite que Fibonacci pour  $N$  grand si  $j(\cdot)$  est localement symétrique autour de son minimum (algorithme d'optimisation → système dynamique)

**Comparison :**  $N = 10$  évaluations de  $j(\cdot)$ , valeur de  $L_0/L_N$

Dichotomie (1)	Dichotomie (2)	Fibonacci	Section dorée
$2^{10} = 1024$	$2^5 = 32$	89	76

Dichotomie (1) : 1 calcul de dérivée  $j'(\cdot) \approx 1$  calcul de critère  $j(\cdot)$

Dichotomie (2) : 1 calcul de dérivée  $j'(\cdot) \approx 2$  calculs de critère  $j(\cdot)$

(différences finies)

### 1.3) Interpolation polynomiale :

On utilise plusieurs évaluations de  $j(\cdot)$  ou  $j'(\cdot)$  pour construire une interpolation

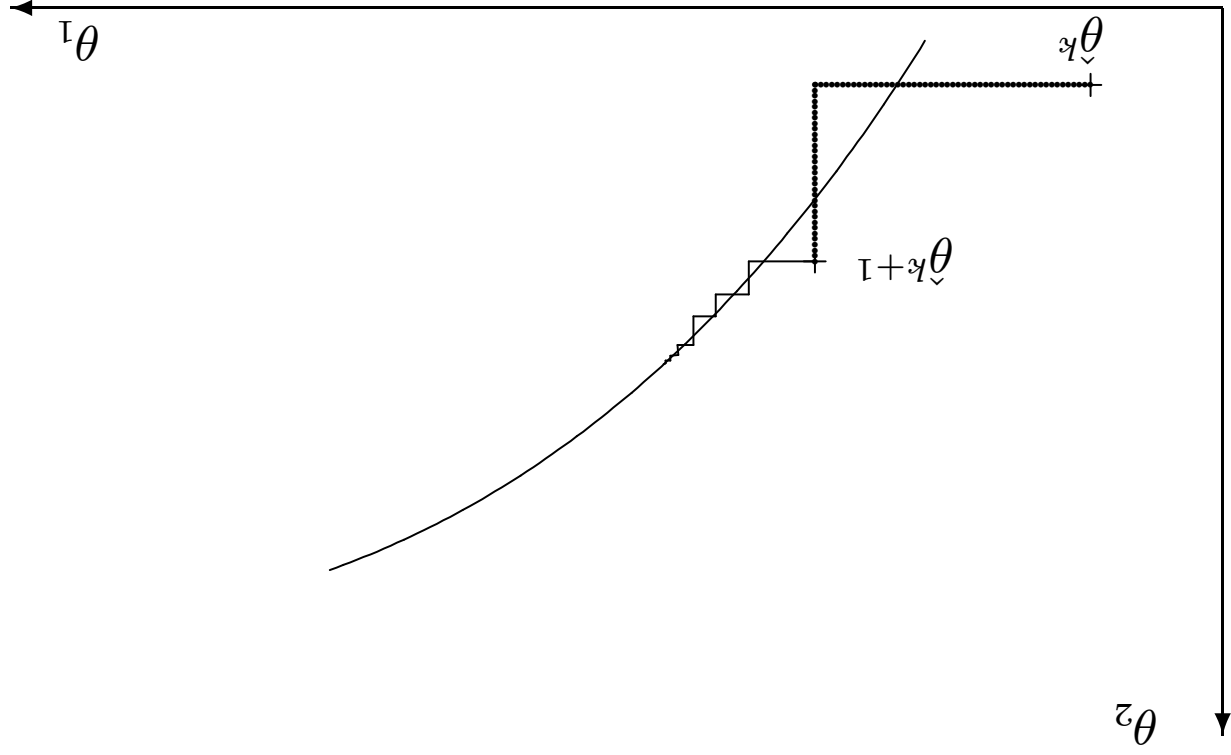
3 évaluations de  $j(\cdot) \rightarrow$  parabole

3 de  $j(\cdot)$  et 1 de  $j'(\cdot)$ , ou 2 de  $j(\cdot)$  et 2 de  $j'(\cdot) \rightarrow$  polynôme de degré 3, etc.

## 2) Combinaison de recherches à une dimension

2.1) Exploration cyclique des paramètres

$$\hat{\theta}^k \leftarrow \hat{\theta}^{k+1}, \hat{\theta}^{k+1} = \arg \min_{\theta^i} j(\theta^i) \left( \hat{\theta}_1^{k+1}, \dots, \hat{\theta}_{i-1}^{k+1}, \theta^i, \hat{\theta}_{i+1}^k, \dots, \hat{\theta}_d^k \right)$$



Convergence très lente si «vallée» non orientée suivant l'un des axes

## 2.2) Méthode de Powell

- 1)  $\hat{\theta}^k \rightarrow \hat{\theta}^{k+1}$  par minimisation suivant  $p$  directions  $\mathbf{d}_i$  indépendantes
- 2)  $\hat{\theta}^k \rightarrow \hat{\theta}^{k+1}$  par minimisation suivant  $\mathbf{d}^{p+1} = \hat{\theta}^{k+1} - \hat{\theta}^k$
- 3) remplacer la «meilleure» direction  $\mathbf{d}_i, i = 1, \dots, p$ , par  $\mathbf{d}^{p+1}$ ,  
retourner en 1)

**Rq1 :** «Meilleure direction» : plus forte décroissance de  $j(\cdot)$   
→ assurer l'indépendance des  $\mathbf{d}_i$

**Rq2 :** Initialisation des  $\mathbf{d}_i$  par les axes  $\mathbf{e}_i$  de l'espace

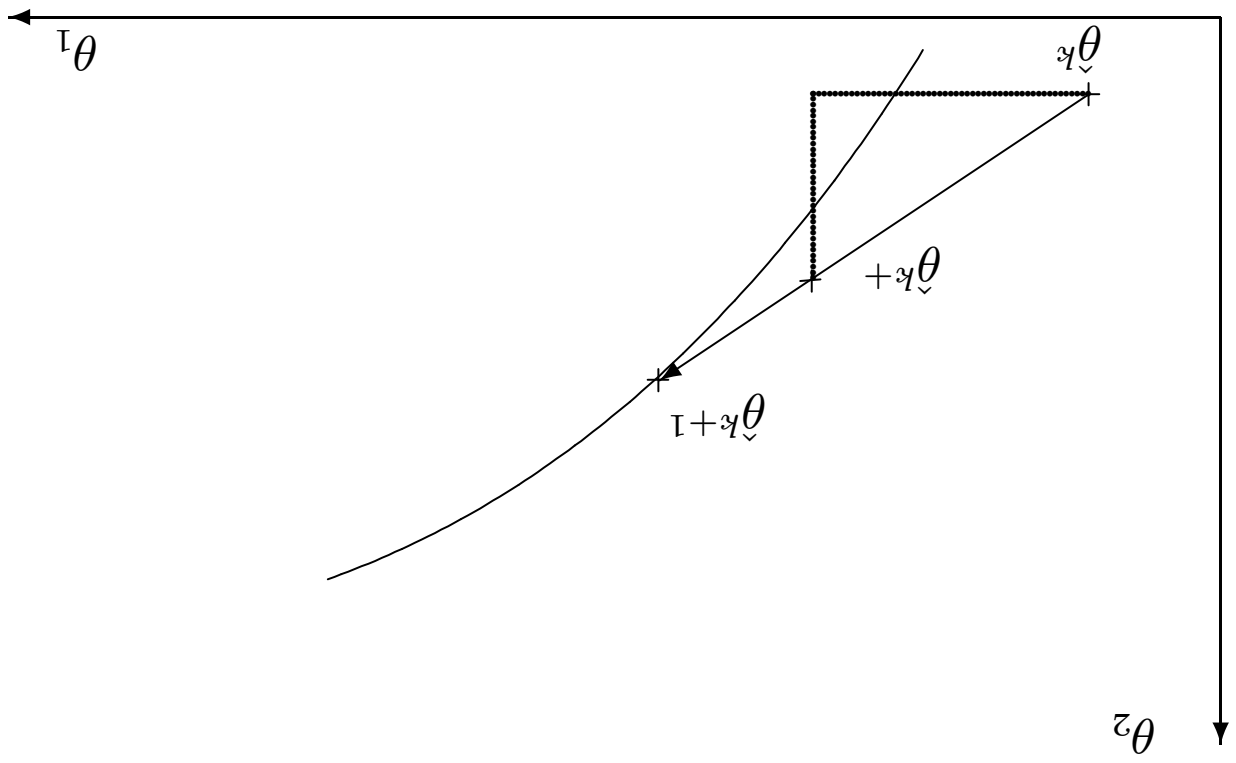
**Mais il faut que  $j(\cdot)$  soit dérivable !**

**Rq5 :** On peut donc minimiser  $j(\cdot)$  sans jamais calculer sa dérivée !

$j(\cdot)$  est quadratique, voir + loin)

**Rq4 :**  $\exists$  variantes plus sophistiquées (génère des directions conjuguées si

**Rq3 :** On peut aussi ré-initialiser les  $d_i$  par les  $e_i$  périodiquement



### 3) Gradient

#### 3.1) Algorithme

Pas recommandé, mais étape nécessaire pour la suite

Développement limité de  $j(\cdot)$  au 1er ordre

$$j(\hat{\theta}_{k+1}) = j(\hat{\theta}_k + \Delta\theta) \approx j(\hat{\theta}_k) + \mathbf{g}_\top(\hat{\theta}_k) \Delta\theta$$

avec  $\mathbf{g}(\hat{\theta}_k) = \left. \frac{\partial j(\theta)}{\partial \theta} \right|_{\theta=\hat{\theta}_k}$  le gradient de  $j(\cdot)$  en  $\hat{\theta}_k$

Minimiser l'approximation linéaire : à  $\|\Delta\theta\|$  fixé  $\Rightarrow \Delta\theta = -\lambda \mathbf{g}(\hat{\theta}_k)$ ,  $\lambda > 0$

Algorithme du gradient :  $\hat{\theta}_{k+1} = \hat{\theta}_k - \lambda \mathbf{g}(\hat{\theta}_k)$

Choix de  $\lambda$  ?

$\lambda$  constant

**Théorème :** Si  $j(\theta) > -\infty, \forall \theta$ , et si  $\mathbf{g}(\cdot)$  satisfait une condition de Lipschitz

$$\|\mathbf{g}(\theta_a) - \mathbf{g}(\theta_b)\| \leq L \|\theta_a - \theta_b\|$$

alors convergence vers un point stationnaire ( $\mathbf{g}(\theta) = \mathbf{0}$ ) et  $j(\cdot)$  décroît de façon monotone si  $0 < \lambda < 2/L$

Si de plus

$$L_m \mathbf{I}_p \preceq \frac{\partial^2 j(\theta)}{\partial \theta \partial \theta^\top} \preceq L_M \mathbf{I}_p, \forall \theta,$$

avec  $L_m > 0$ , alors  $\|\hat{\theta}^k - \hat{\theta}^\infty\| \leq \|\hat{\theta}^0 - \hat{\theta}^\infty\| q^k$  (convergence linéaire — ou exponentielle), avec

$$q = \max\{|1 - \lambda L_m|, |1 - \lambda L_M|\}$$

Vitesse maximum :  $q^* = \frac{L_M - L_m}{L_M + L_m}$ , obtenue pour  $\lambda^* = \frac{(L_M + L_m)}{2}$

En pratique, pas très utile...



Donc, prendre  $\lambda$  variable

$$\lambda_k \geq 0, \lambda_k \rightarrow 0 \text{ et } \sum_{k=0}^{\infty} \lambda_k = \infty \text{ convient}$$

... mais très lent

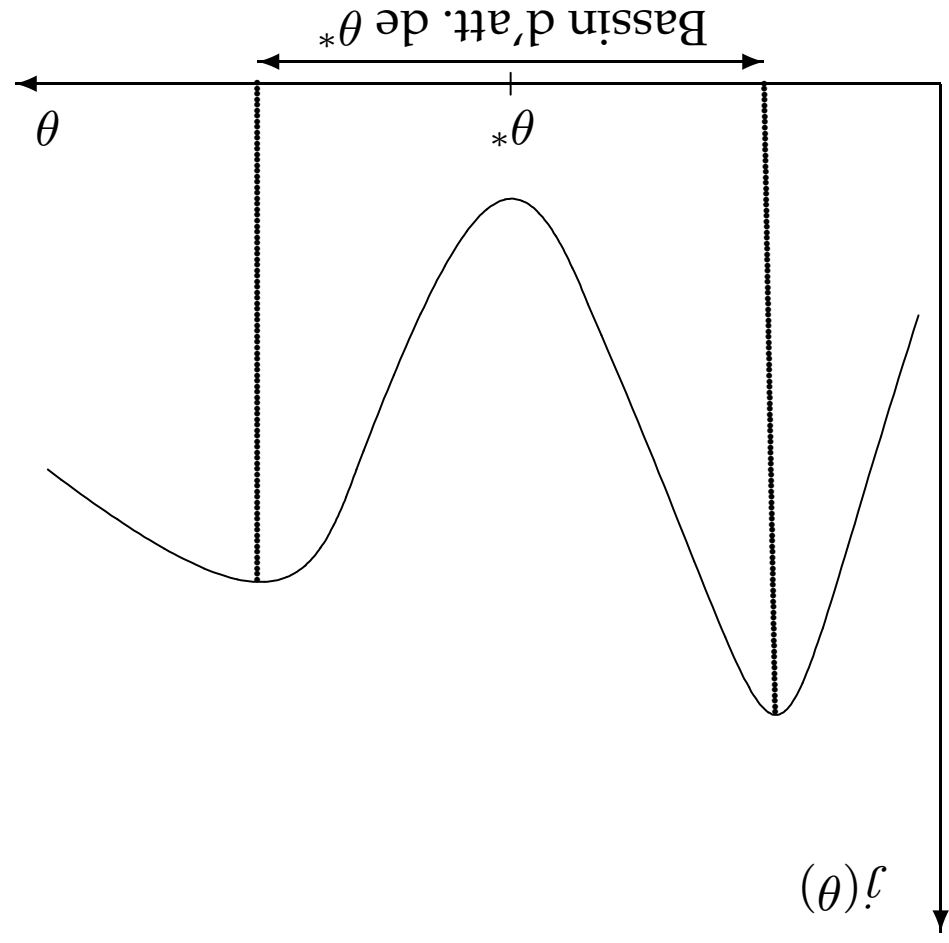
Optimiser par rapport à  $\lambda$  à chaque itération (voir paragraphe 1)

Adapter  $\lambda$  ( $\lambda_k$  à l'itération  $k$ )

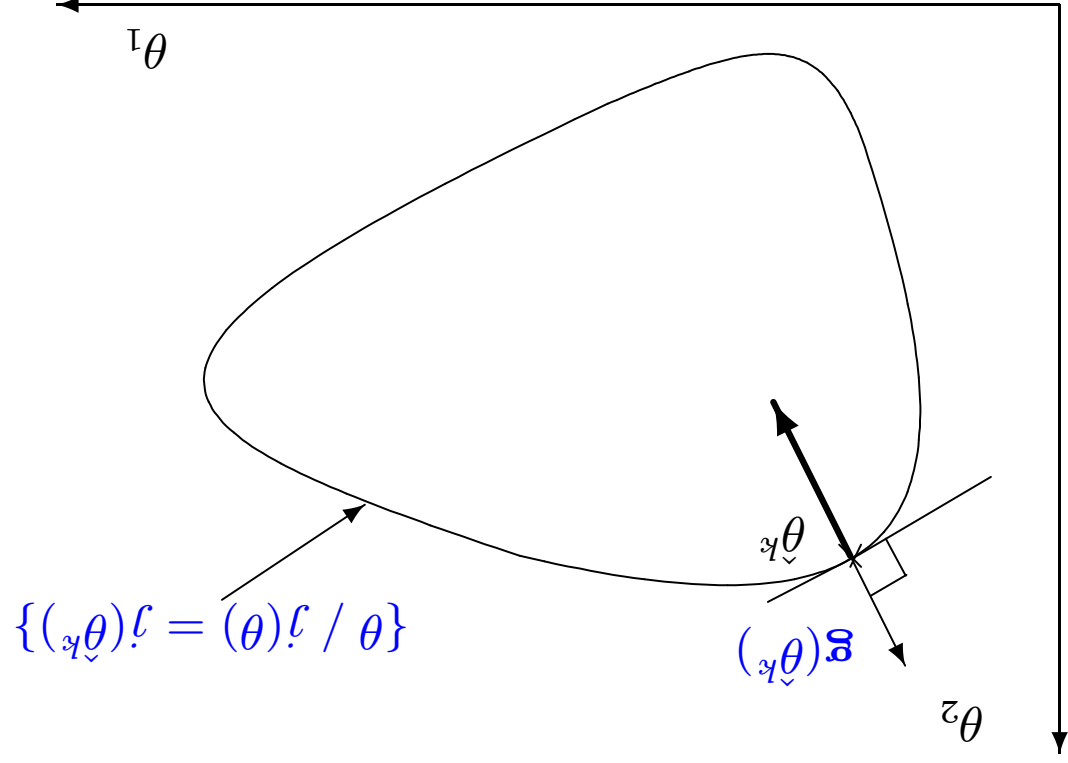
- ☞ si  $j(\hat{\theta}_{k+1}) > j(\hat{\theta}_k)$ , accepter  $\hat{\theta}_{k+1}$ , prendre  $\lambda_{k+1} = 1.5\lambda_k$  (accélérer)
- ☞ si  $j(\hat{\theta}_{k+1}) > j(\hat{\theta}_k)$ , rejeter  $\hat{\theta}_{k+1}$ , prendre  $\lambda_{k+1} = 0.5\lambda_k$  (essayer plus près)

# Propriétés de l'algorithme du gradient

PG1 : Simple, grand domaine de convergence (bassin d'attraction)

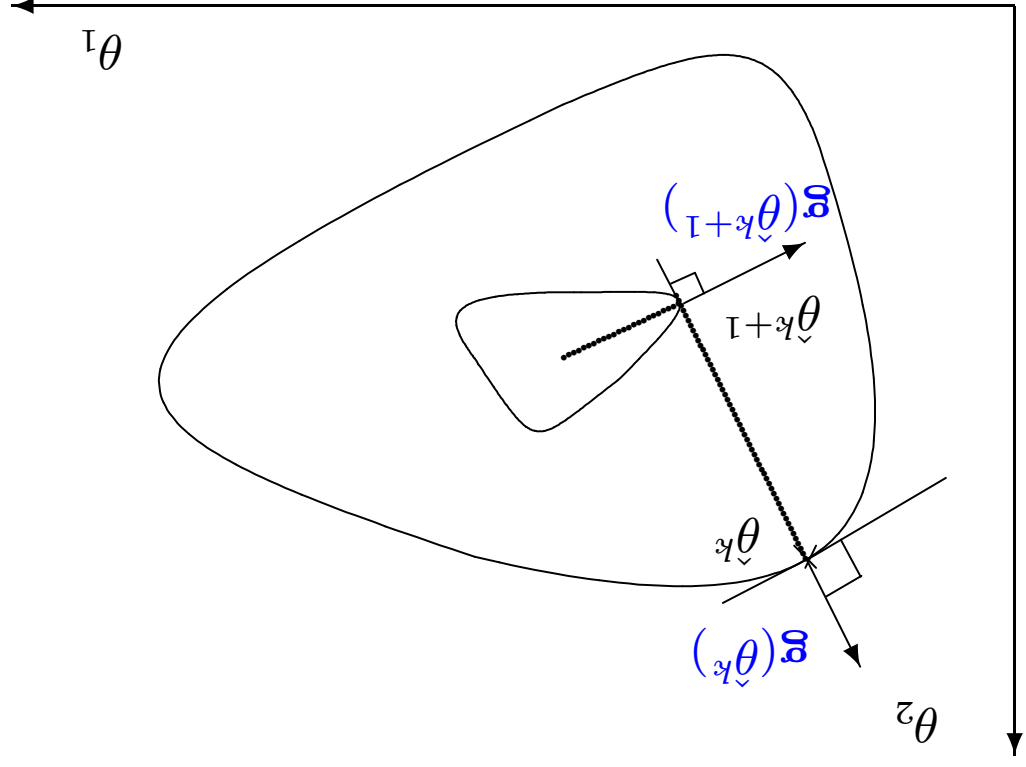


**PG2:** Itération  $k \rightarrow$  direction de recherche  $-\mathbf{g}_k \perp$  surface isocritère (courbe de niveau)  $j(\hat{\theta}_k)$



**PG3:** Toute direction faisant un angle  $> \pi/2$  avec  $-\mathbf{g}(\hat{\theta}_k)$  est acceptable (direction de descente)

**PG4:** Si minimisation précise à chaque itération, les directions de recherche successives sont  $\perp$



**PG5**: La trajectoire suivie dépend de la paramétrisation

$$\text{Exemple: } j(\theta) = \theta_1^2 + \theta_2^2$$

$$\theta \rightarrow \theta' = (\theta_1, \theta_2/10) \Leftrightarrow j(\theta') = \theta'^2_1 + 100\theta'^2_2$$

Convergence + rapide quand les isocritères sont approximativement sphériques ( $\rightarrow$  changement de métrique, voir + loin quasi-Newton)

Utile loin de l'optimum, très lent quand on s'en rapproche : phase initiale de l'optimisation

## 3.2) Calcul du gradient

### 3.2.1) Différences finies :

$$[\mathbf{g}(\theta)]_i = \frac{1}{\Delta_i} [j(\theta) + \Delta_i \mathbf{e}_i] - j(\theta), \quad i = 1, \dots, p$$

avec  $\Delta_i$  «petit»

$\rightarrow p$  (+1) calculs de  $j(\cdot)$  et résultat approximatif

### 3.2.2) Fonctions de sensibilité :

$j(\theta)$  s'écrit comme une fonction des erreurs  $e(\theta, i)$  (par ex.  $e(\theta, i) = y(i) - y_m(\theta, i)$ )

$\rightarrow \mathbf{g}(\theta)$  fait intervenir les dérivées  $\frac{\partial [e(\theta, i)]}{\partial \theta} = \mathbf{s}^e(\theta, i)$  = fonctions de sensibilité

$$\text{Ex: } j(\theta) = \sum_{i=1}^n w_i e^2(\theta, i) \text{ (MC pondérés)}$$
$$\rightarrow \mathbf{g}(\theta) = 2 \sum_{i=1}^n w_i e(\theta, i) \mathbf{s}^e(\theta, i)$$

Comment calculer  $\mathbf{s}^e(\theta, i)$  ?

Erreur de sortie :  $e(\theta, i) = y(i) - y_m(\theta, i) \rightarrow s_e(\theta, i) = -s_y(\theta, i)$

Modèle LI, eq. diff. d'ordre  $m$ , conditions initiales d'effet négligeable  $\rightarrow y_m$  et  $s_y(\theta, i), i = 1, \dots, p$ : simulation d'une eq. diff. d'ordre  $2m$  !

**Ex1 :**

$$d^2 y_m(\theta, t) / dt^2 + \theta_1 dy_m(\theta, t) / dt + \theta_2 y_m(\theta, t) = \theta_3 \frac{du(t)}{dt} + \theta_4 u(t)$$

avec  $y_m(\theta, 0) = 0, \left. \frac{dy_m(\theta, t)}{dt} \right|_{t=0} = 0$

On dérive par rapport à  $\theta_i \rightarrow s_i(\theta, t), i = 1, \dots, 4$

$$d^2 s_{s1}(\theta, t) / dt^2 + \theta_1 ds_{s1}(\theta, t) / dt + \theta_2 s_{s1}(\theta, t) = ds_{s1}(\theta, t) / dt$$

$$d^2 s_{s2}(\theta, t) / dt^2 + \theta_1 ds_{s2}(\theta, t) / dt + \theta_2 s_{s2}(\theta, t) = ds_{s2}(\theta, t) / dt$$

$$d^2 s_{s3}(\theta, t) / dt^2 + \theta_1 ds_{s3}(\theta, t) / dt + \theta_2 s_{s3}(\theta, t) = ds_{s3}(\theta, t) / dt$$

$$d^2 s_{s4}(\theta, t) / dt^2 + \theta_1 ds_{s4}(\theta, t) / dt + \theta_2 s_{s4}(\theta, t) = ds_{s4}(\theta, t) / dt$$

et toutes les conditions initiales sont nulles

5 eq. diff. d'ordre 2 → système d'ordre 10?

Non, car elles sont linéaires et ont toutes le même premier membre

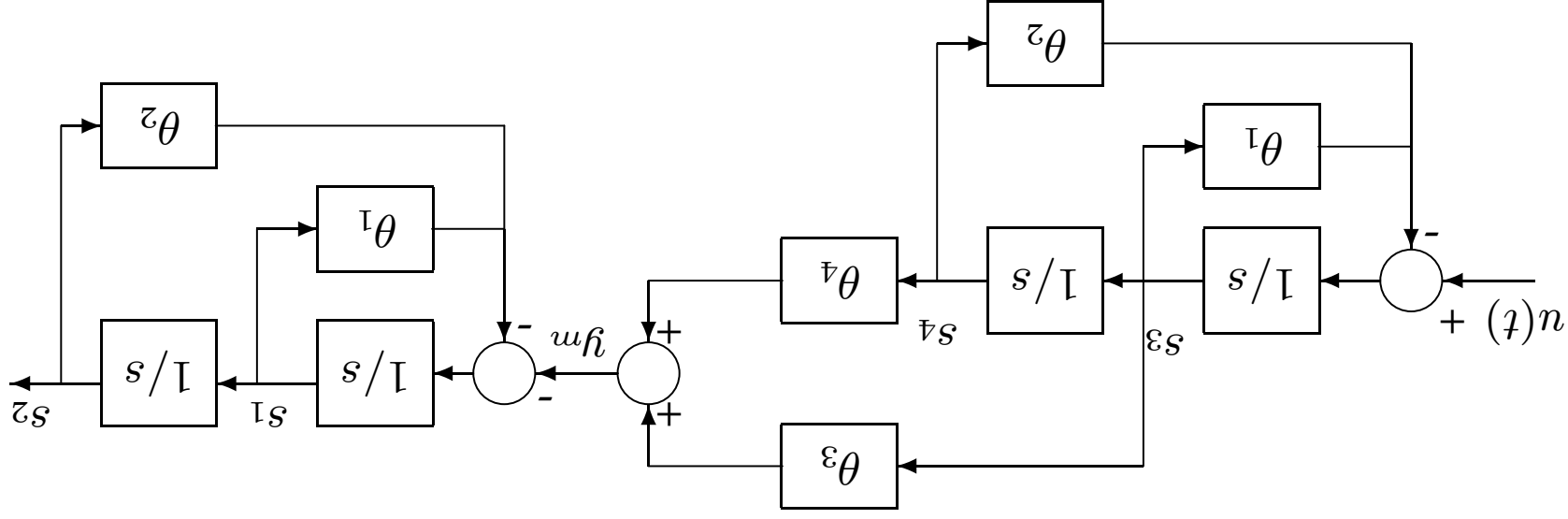
On simule celle donnant  $s_4(\theta, t)$

→ on obtient  $s_3(\theta, t)$  en dérivant par rapport à  $t$

→ puis  $y_m(\theta, t)$  en combinant  $s_3(\theta, t)$  et  $s_4(\theta, t)$

On simule celle donnant  $s_2(\theta, t)$

→ on obtient  $s_1(\theta, t)$  en dérivant par rapport à  $t$





Représentation d'état :  $\mathbf{x}(t) = [s_3, s_4, s_1, s_2]^T$  (sorties des 4 intégrateurs)

$$\frac{d\mathbf{x}(t)}{dt} = \mathbf{A}\mathbf{x}(t) + \mathbf{B}u(t), \mathbf{x}(0) = \mathbf{0}$$

avec

$$\mathbf{A} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ -\theta_3 & 1 & 0 & 0 \\ -\theta_4 & 0 & 0 & 0 \\ -\theta_1 & -\theta_2 & 0 & 0 \end{pmatrix}, \mathbf{B} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

Les réponses d'intérêt :  $\mathbf{z}(\theta, t) = (y_m, s_1, s_2, s_3, s_4)$ , soit

$$\mathbf{z}(\theta, t) = \mathbf{C}\mathbf{x}(t), \text{ avec } \mathbf{C} = \begin{pmatrix} \theta_3 & 0 & 0 & 0 \\ \theta_4 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Plus généralement, erreur de prédiction...

$$\text{Ex2: «ARMAX» } A(\theta, q)y(k) = B(\theta, q)u(k) + \frac{C(\theta, q)}{D(\theta, q)}\epsilon(k)$$

→ erreur de prédiction (utilisée pour max. de vraisemblance)

$$e(\theta, k) = \frac{D(\theta, q)}{C(\theta, q)} [A(\theta, q)y(k) - B(\theta, q)u(k)]$$

On dérive par rapport aux paramètres  $a_i$  dans  $A(\theta, q)$ ,  $b_i$  dans  $B(\theta, q)$ , etc.

$$\begin{aligned} \frac{\partial e(\theta, k)}{\partial a_i} &= \frac{D(\theta, q)}{C(\theta, q)} y(k-i) \\ &= \frac{\partial b_i}{\partial e(\theta, k)} - \frac{D(\theta, q)}{C(\theta, q)} u(k-i) \end{aligned}$$

Puis  $C(\theta, q)e(\theta, k) = D(\theta, q)[A(\theta, q)y(k) - B(\theta, q)u(k)]$  donne

$$\frac{\partial e(\theta, k)}{\partial c_i} = - \frac{C(\theta, q)}{1} e(\theta, k-i)$$

et enfin

$$\frac{\partial e(\theta, k)}{\partial d_i} = \frac{C(\theta, q)}{1} [A(\theta, q)y(k-i) - B(\theta, q)u(k-i)]$$

Plus généralement, modèle NLI ...

Représentation d'état, temps continu

$$\frac{dx(t)}{dt} = \mathbf{f}[\mathbf{x}(t), \theta], \quad \mathbf{x}(0) = \mathbf{x}_0$$

$$\mathbf{h}[\mathbf{x}(t), \theta] = \mathbf{y}_m(t, \theta)$$

(f et h peuvent aussi dépendre de  $u(t)$  et de  $t...$ )

$$\frac{\partial \theta_i}{\partial \mathbf{h}[\mathbf{x}(t), \theta]} + \frac{\partial \theta_i}{\partial \mathbf{x}(t)} \frac{\partial \mathbf{x}(t)}{\partial \mathbf{h}[\mathbf{x}(t), \theta]} = \frac{\partial \theta_i}{\partial \mathbf{y}_m(t, \theta)} = [\mathbf{s}_y(t, \theta)]_i$$

← calculer

$$\frac{\partial \theta_i}{\partial \mathbf{x}(t)} = [\mathbf{s}_x(t, \theta)]_i$$

On dérive...

$$\frac{\partial \theta_i}{\partial \mathbf{x}_0} = [\mathbf{s}_x(0, \theta)]_i, \quad \frac{\partial \theta_i}{\partial \mathbf{x}(t)} \frac{\partial \mathbf{x}(t)}{\partial \mathbf{x}_0} + \frac{\partial \theta_i}{\partial \mathbf{f}[\mathbf{x}(t), \theta]} \frac{\partial \mathbf{x}(t)}{\partial \mathbf{x}_0} = \frac{d}{dt} [\mathbf{s}_x(t, \theta)]_i$$

→ Une simulation pour avoir  $\mathbf{x}(t)$   
 →  $p$  simulations pour avoir les  $[s^x(\theta, t)]_i$  (**Rq** : chaque eq. diff. est linéaire,  
 non stationnaire car  $\frac{\partial \mathbf{f}[\mathbf{x}, \theta]}{\partial \mathbf{x}}$  dépend de  $t$ , et seuls le terme de commande  
 $\frac{\partial \mathbf{f}[\mathbf{x}(t), \theta]}{\partial \theta_i}$  et les conditions initiales changent avec  $i$ )

⇒ pas plus compliqué que par différences finies, et pas d'approximation !

3.2.3) Etat adjoint : (encore plus fort !)

Toujours représentation d'état, temps discret

$$\mathbf{x}(k+1) = \mathbf{f}[\mathbf{x}(k), \theta], \quad \mathbf{x}(0) = \mathbf{x}_0(\theta)$$

$$\mathbf{y}^m(\theta, k) = \mathbf{h}[\mathbf{x}(k), \theta]$$

On suppose le critère  $J(\theta)$  **additif** (pas très restrictif) :

$$J(\theta) = \sum_n^{i=0} r_i[\mathbf{x}(i), \theta]$$

→ on le transforme en coût terminal en introduisant une variable d'état supplémentaire  $x_0(k)$

$$\text{Evolution: } \mathbf{x}^e(k+1) = \begin{pmatrix} \theta \\ \mathbf{f}[\mathbf{x}(k), \theta] \\ x_0(k) + r_k[\mathbf{x}(k), \theta] \end{pmatrix} = \mathbf{f}^e[\mathbf{x}^e(k)]$$

$$\text{Critère: } j(\theta) = [1 \ 0 \ \dots \ 0 \ 1] \mathbf{x}^e(n+1)$$

$$\text{État étendu: } \mathbf{x}^e(k) = \begin{pmatrix} \theta \\ \mathbf{x}(k) \\ x_0(k) \end{pmatrix} \text{ et } \mathbf{x}^e(0) = \begin{pmatrix} \theta \\ \mathbf{x}_0(\theta) \\ 0 \end{pmatrix}$$

de telle sorte que  $j(\theta) = x_0(n+1)$

$$= x_0(k) + r_k[\mathbf{x}(k), \theta], \quad x_0(0) = 0$$

$$= x_0(k+1) = \sum_{i=k}^{0=?} r_i[\mathbf{x}(i), \theta]$$

$$\begin{aligned} \frac{\partial \theta}{\partial \mathbf{x}_\perp^e(1)} &= \frac{\partial \theta}{\partial \mathbf{x}_\perp^e(0)} \frac{\partial \mathbf{x}_\perp^e(0)}{\partial \mathbf{J}_\perp^e(0)} \\ \dots & \\ \frac{\partial \theta}{\partial \mathbf{x}_\perp^e(n)} &= \frac{\partial \theta}{\partial \mathbf{x}_\perp^e(n-1)} \frac{\partial \mathbf{x}_\perp^e(n-1)}{\partial \mathbf{J}_\perp^e(n-1)} \\ \frac{\partial \theta}{\partial \mathbf{x}_\perp^e(n+1)} &= \frac{\partial \theta}{\partial \mathbf{x}_\perp^e(n)} \frac{\partial \mathbf{x}_\perp^e(n)}{\partial \mathbf{J}_\perp^e(n)} \end{aligned}$$

On continue, avec  $\mathbf{x}_\perp^e(n+1) = \mathbf{f}_\perp^e[\mathbf{x}_\perp^e(n)]$ , etc.

$$\frac{\partial j}{\partial \mathbf{x}_\perp^e(n+1)} = [1 \ 0 \ \dots \ 0]_\perp$$

avec

$$\mathbf{g}(\theta) = \frac{\partial j(\theta)}{\partial \theta} = \frac{\partial \theta}{\partial \mathbf{x}_\perp^e(n+1)} \frac{\partial \mathbf{x}_\perp^e(n+1)}{\partial j} = \mathbf{g}(\theta)$$

→ règle de dérivation en chaîne (fonction composée)

$$\left( \begin{array}{c} \mathbf{0} \\ \frac{\partial \theta}{\partial \mathbf{x}_\perp^e(0)} \\ \mathbf{I}^p \end{array} \right) = \frac{\partial \theta}{\partial \mathbf{x}_\perp^e(0)} \mathbf{p}^e(0), \text{ avec } \frac{\partial \theta}{\partial \mathbf{x}_\perp^e(0)} = \mathbf{g}(\theta)$$

$$\mathbf{p}^e(k-1) = \frac{\partial \mathbf{f}_\perp^e[\mathbf{x}^e(k-1)]}{\partial \mathbf{x}^e(k-1)} \mathbf{p}^e(k), \quad k = n+1, \dots, 1$$

satisfait l'équation de récurrence à temps rétrograde

$$\left( \begin{array}{c} 0 \\ \vdots \\ 0 \\ 1 \end{array} \right) = \text{Etat adjoint : initialisé par } \mathbf{d}^e(n+1)$$

$$\left( \begin{array}{c} 0 \\ \vdots \\ 0 \\ 1 \end{array} \right) = \frac{\partial \theta}{\partial \mathbf{x}_\perp^e(0)} \frac{\partial \mathbf{f}_\perp^e[\mathbf{x}^e(0)]}{\partial \mathbf{x}^e(0)} \dots \frac{\partial \mathbf{f}_\perp^e[\mathbf{x}^e(1)]}{\partial \mathbf{x}^e(1)} \frac{\partial \mathbf{f}_\perp^e[\mathbf{x}^e(n-1)]}{\partial \mathbf{x}^e(n-1)} \frac{\partial \mathbf{f}_\perp^e[\mathbf{x}^e(n)]}{\partial \mathbf{x}^e(n)}$$

soit finalement

Une simulation à temps direct : évolution de  $\mathbf{x}^e(k) \rightarrow \mathbf{j}(\theta)$   
 Une simulation à temps rétrograde : évolution de  $\mathbf{d}^e(k) \rightarrow \mathbf{g}(\theta)$   
 $\dots \forall p = \dim(\theta)$  et sans approximation !

**Rq1 :** L'état adjoint satisfait

$$\mathbf{d}^e(k+1) = \left[ \frac{\partial \mathbf{f}^e[\mathbf{x}^e(k)]}{\partial \mathbf{x}^e(k)} \right]^{-1} \mathbf{d}^e(k)$$

Linearisation de l'évolution de l'état (étendu) :

$$\delta \mathbf{x}^e(k+1) = \frac{\partial \mathbf{f}^e[\mathbf{x}^e(k)]}{\partial \mathbf{x}^e(k)} \delta \mathbf{x}^e(k)$$

et donc  $\delta \mathbf{x}^e(k+1) \perp \mathbf{d}^e(k+1) = \delta \mathbf{x}^e(k) \perp \mathbf{d}^e(k) = \text{cte le long de la}$   
 trajectoire

← utile pour vérifier les calculs

**Rq2 :** Modèle à temps continu → chaque simulation (temps direct et rétrograde) implique des approximations, attention !



### 3.2.4) Code adjoint: (de plus en plus fort !)

Idee: **code informatique calculant  $j(\theta) \Leftrightarrow$  système à temps discret**

$\mathbf{v}$ : vecteur de toutes les variables utilisées

la ligne  $k$  du code modifie *une variable*, disons celle de numéro  $\mu(k)$

$$v_{\mu(k)} = \phi_k(v_i, i \in \mathcal{I}_k)$$

$\mathcal{I}_k \rightarrow$  ensemble des variables utilisées pour calculer  $v_{\mu(k)}$

$\Leftrightarrow$  transformation  $\Phi_k$  appliquée à  $\mathbf{v}$

$$[\Phi_k(\mathbf{v})]_j = v_j, \forall j \neq \mu(k)$$

$$[\Phi_k(\mathbf{v})]_{\mu(k)} = \phi_k(v_i, i \in \mathcal{I}_k)$$

Finallement, on distingue les valeurs de  $\mathbf{v}$  avant et après l'exécution de la

ligne  $k$ :

$$\mathbf{v}(k) = \Phi_k[\mathbf{v}(k-1)], k = 1, \dots, f$$

avec  $f$  la dernière ligne du code

Convention :

$p$  premières composantes de  $\mathbf{v} = \theta$

ensuite, les variables indépendantes (nécessaires pour le calcul de  $j(\theta)$ , par ex.  $y^{(i)}, i = 1, \dots, n$ )

ensuite les variables dépendantes (variables intermédiaires)

enfin, la valeur de  $j(\theta)$ , dernière composante de  $\mathbf{v}$

... et donc,  $j(\theta) = [0 \ 0 \ \dots \ 0 \ 1] \mathbf{v}(f)$

Même technique qu'en 3.2.3 : règle de dérivation en chaîne

$$\mathbf{g}(\theta) = \frac{\partial \mathbf{v}^{\top}(0)}{\partial \Phi_1^{\top}} \frac{\partial \mathbf{v}(0)}{\partial \Phi_1^{\top}} \frac{\partial \mathbf{v}(1)}{\partial \Phi_2^{\top}} \dots \frac{\partial \mathbf{v}(f-2)}{\partial \Phi_{f-1}^{\top}} \frac{\partial \mathbf{v}(f-1)}{\partial \Phi_f^{\top}} \frac{\partial \mathbf{v}(f)}{\partial j}$$

← variables duales  $\mathbf{d}$ , initialisées par  $\mathbf{d}(f) = \frac{\partial \mathbf{v}(f)}{\partial j} = [0 \ 0 \ \dots \ 0 \ 1]^{\top}$ , puis

$$\mathbf{d}(k-1) = \frac{\partial \Phi_k^{\top}}{\partial \mathbf{v}(k-1)} \mathbf{d}(k), \quad k = f, \dots, 1$$

et  $\frac{\partial \Phi_{\perp}^{\Lambda}(k)}{\partial \gamma}$  est donné par

$$\begin{aligned}
 & \left( \begin{array}{cccc} \mathbb{1} & & & \\ & \ddots & & \\ & & \mathbb{1} & \\ & & & \ddots & \\ f_{\alpha} & \dots & & & \end{array} \right) \left[ \begin{array}{c} \vdots \\ \frac{(1-\gamma)\Lambda e}{\gamma \Phi e} \\ \vdots \end{array} \right] \left( \begin{array}{cccc} & & & \\ & & & \mathbb{1} \\ & \mathbb{1} & & \\ & & \ddots & \\ & & & \mathbb{1} \end{array} \right) = \frac{(1-\gamma)\Lambda e}{\gamma \Phi_{\perp} e} \\
 & \qquad \qquad \qquad \parallel \\
 & \qquad \qquad \qquad (\gamma \mathcal{I} \ni v_i, i \in \mathcal{I}_k) \gamma \phi \qquad \qquad \qquad \leftarrow (\gamma)_{\perp}^{\Lambda}
 \end{aligned}$$

Ceci donne les récurrences

$$d_i(k-1) = d_i(k) + \frac{\partial v_i}{\partial \phi_k} p^{(k)\mu(k)} (1) \text{ pour } i \neq \mu(k)$$

$$\text{et } d_{\mu(k)}(k-1) = \frac{\partial v_{\mu(k)}}{\partial \phi_k} p^{(k)\mu(k)} (1)$$

et enfin

$$\mathbf{g}(\theta) = \frac{\partial \mathbf{v}}{\partial \boldsymbol{\theta}}(0), \text{ avec } \frac{\partial \mathbf{v}}{\partial \boldsymbol{\theta}}(0) = \mathbf{I}^d \mathbf{0}$$

← les  $p$  premières composantes de  $\mathbf{d}(0)$  correspondent à  $\mathbf{g}(\theta)$

← composantes suivantes : dérivées de  $j(\cdot)$  par rapport aux variables

indépendantes (par ex.  $y(i)$ )

Il est inutile de mémoriser les valeurs successives de  $\mathbf{d}$  :

$$v_{\mu(k)} = \phi_k(v_i, i \in \mathcal{I}_k) \text{ donne dans le code adjoint}$$

$$\forall i \in \mathcal{I}_k, i \neq \mu(k), d_i = d_i + \frac{\partial \phi_k}{\partial v_i} d_{\mu(k)}$$

$$d_{\mu(k)} = \frac{\partial \phi_k}{\partial v_{\mu(k)}} d_{\mu(k)}$$

**Initialisation :**  $\phi_k$  ne dépend pas de  $v_{\mu(k)} \leftarrow d_{\mu(k)} = 0$

**Incrément :**  $v_{\mu(k)} = v_{\mu(k)} + \phi_k(v_i, i \neq \mu(k)) \leftarrow d_{\mu(k)} = d_{\mu(k)}$

**Ex :** code direct :  $j = j + [y(k) - y_m(k)]^2$

$\leftarrow$  3 variables  $j, y(k), y_m(k)$  et 3 variables duales  $d_j, d_y(k), d_{y_m(k)}$

$\leftarrow$  code adjoint :

$$\begin{aligned} dy(k) &= dy(k) + 2[y(k) - y_m(k)] d_j \\ dy_m(k) &= dy_m(k) - 2[y(k) - y_m(k)] d_j \\ d_j &= d_j \text{ (bien sûr, inutile)} \end{aligned}$$

**Rq1** : Le code adjoint est  $\approx 4$  fois plus «gros» au plus que le code direct  
**Rq2** : Boucle `FOR` du code direct  $\rightarrow$  boucle `FOR` du code adjoint *mais en sens inverse* !

**Rq3** : Se souvenir du chemin suivi dans le code direct ( $\perp F$ )  $\rightarrow$  même chemin dans le code adjoint

**Rq4** : Se souvenir de la valeur des variables du code direct intervenant dans des expressions non linéaires

**Rq5** : Créer un sous-programme adjoint pour chaque sous-programme direct

**Rq6** : Développer (et modifier !) les deux codes en parallèle

Gradient  $g(\theta)$  obtenu en deux simulations (et sans approximation) !

## 4) Newton & Gauss-Newton

### 4.1) Newton :

Développement limite de  $j(\theta)$  au deuxième ordre en  $\hat{\theta}_k$

$$j(\hat{\theta}_{k+1}) = j(\hat{\theta}_k + \Delta\theta) \approx j(\hat{\theta}_k) + \mathbf{g}_\perp(\hat{\theta}_k) \Delta\theta + \frac{1}{2} \Delta\theta^\top \mathbf{H}(\hat{\theta}_k) \Delta\theta$$

avec  $\mathbf{g}(\hat{\theta}_k)$  le gradient de  $j(\cdot)$  en  $\hat{\theta}_k$  (vecteur de dim.  $p$ ) et

$$\mathbf{H}(\hat{\theta}_k) = \frac{\partial^2 j(\theta)}{\partial \theta \partial \theta^\top} \Big|_{\theta = \hat{\theta}_k}$$

le Hessien de  $j(\cdot)$  en  $\hat{\theta}_k$  (matrice symétrique  $p \times p$ )  
 $\Delta\theta$  assurant la plus forte décroissance de  $j(\cdot)$  ?

← condition de stationnarité :

$$\mathbf{0} = \frac{e[\Delta\theta]}{\partial \{j(\hat{\theta}_k) + \mathbf{g}_\perp(\hat{\theta}_k) \Delta\theta + \frac{1}{2} \Delta\theta^\top \mathbf{H}(\hat{\theta}_k) \Delta\theta\}}$$

→ déplacement (*pas*)  $\Delta\theta = -\mathbf{H}^{-1}(\hat{\theta}_k)\mathbf{g}(\hat{\theta}_k)$

Algorithme de Newton:  $\hat{\theta}_{k+1} = \hat{\theta}_k - \mathbf{H}^{-1}(\hat{\theta}_k)\mathbf{g}(\hat{\theta}_k)$

(Rappel : algorithme du gradient  $\rightarrow \hat{\theta}_{k+1} = \hat{\theta}_k - \lambda\mathbf{g}(\hat{\theta}_k)$ )

**PN1** : A présent, direction et pas suggérés simultanément

**PN2** : Calculs beaucoup + lourds (dérivées secondes...)

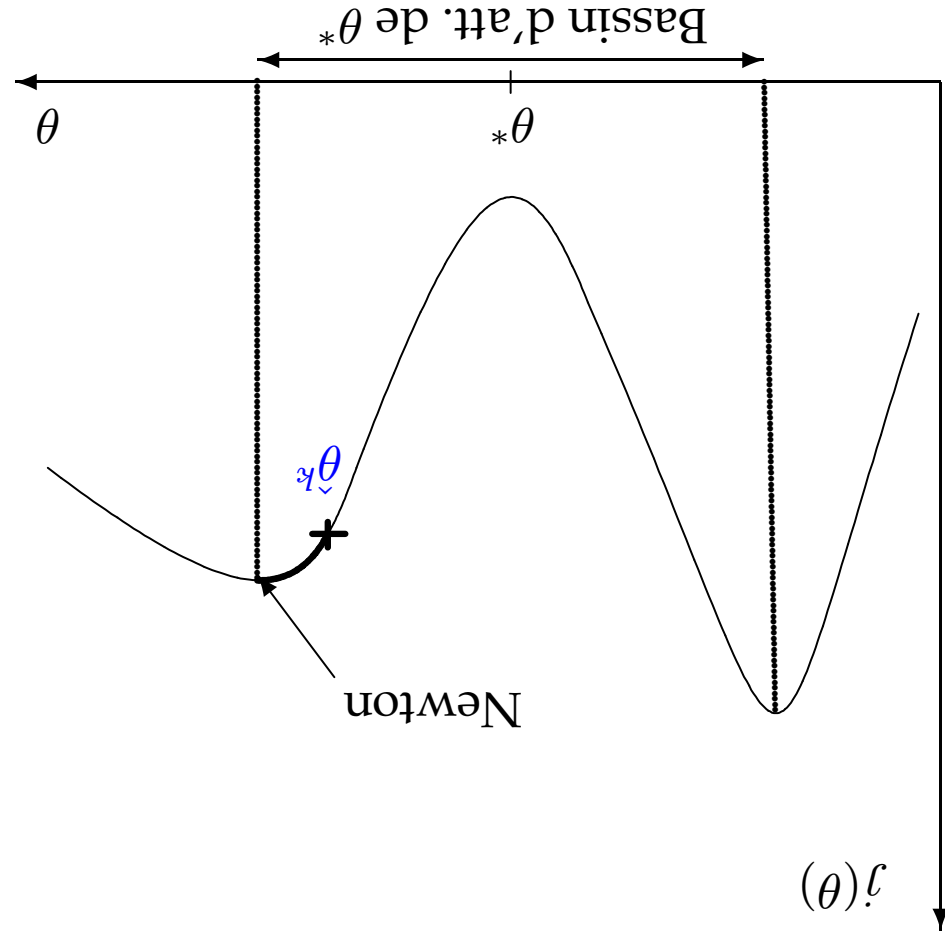
**PN3** : **H** donne une idée de la précision de l'estimation (voir + loin)

**PN4** : Pas d'accumulation d'erreurs

**PN5** : Correct si  $\mathbf{H}(\hat{\theta}_k) \succ \mathbf{O}$  (on n'a pas utilisé le fait que l'on minimise  $J(\cdot)$  !)



PN6: Domaine de convergence plus réduit que pour l'alg. du gradient



**PN7:**  $j(\theta)$  quadratique  $\Rightarrow$  convergence en *une* itération !

$$[\text{Ex.: } j(\theta) = \mathbf{e}^\top(\theta) \mathbf{Q} \mathbf{e}(\theta), \mathbf{e}(\theta) = \mathbf{y} - \mathbf{R}\theta]$$

Si  $j(\theta)$  non quadratique, mais  $\hat{\theta}_k$  proche de l'optimum, convergence très rapide (*quadratique* — linéaire pour le gradient)

**PN8:** Si  $\mathbf{H}(\hat{\theta}_k) \succ \mathbf{O}$ ,  $\mathbf{H}^{-1}(\hat{\theta}_k) \succ \mathbf{O}$  et la direction suggérée par Newton

fait un angle  $< \pi/2$  avec celle suggérée par l'alg. du gradient  $\rightarrow j(\theta)$  peut

décroître dans cette direction (si on ne va pas trop loin)

$\leftarrow$  coefficient de relaxation  $\lambda_k$

Algorithme de Newton relaxé:  $\hat{\theta}_{k+1} = \hat{\theta}_k - \lambda_k \mathbf{H}^{-1}(\hat{\theta}_k) \mathbf{g}(\hat{\theta}_k)$

Commencer avec  $\lambda_k = 1$ , rejeter  $\hat{\theta}_{k+1}$  et réduire  $\lambda_k$  tant que  $j(\hat{\theta}_{k+1}) > j(\hat{\theta}_k)$

**PN9:** Implémentation  $\rightarrow$  plutôt résoudre  $\mathbf{H}(\hat{\theta}_k) \Delta \theta = -\mathbf{g}(\hat{\theta}_k)$

## Critère quadratique (Moindres carrés)

$$J(\theta) = \frac{1}{2} \sum_{i=1}^n w_i e^2(\theta, i) \text{ donne}$$

$$\text{gradient } \mathbf{g}(\theta) = \sum_{i=1}^n w_i \frac{\partial e(\theta, i)}{\partial \theta} e(\theta, i)$$

$$\text{Hessien } \mathbf{H}(\theta) = \sum_{i=1}^n w_i \frac{\partial e(\theta, i)}{\partial \theta} \frac{\partial \theta}{\partial \theta^\top} + \sum_{i=1}^n w_i \frac{\partial^2 e(\theta, i)}{\partial \theta \partial \theta^\top} e(\theta, i)$$

← fonctions de sensibilité du deuxième ordre  $\frac{\partial^2 e(\theta, i)}{\partial \theta \partial \theta^\top}$

1) lourdes à calculer

2) pas sûr que  $\mathbf{H}(\hat{\theta}_k)$  soit  $\succ \mathbf{O}$  !

simplifier !

## 4.2) Gauss-Newton : Pour un critère

$$j(\theta) = \frac{1}{2} \sum_{i=1}^n w_i e_2^2(\theta, i)$$

remplacer  $\mathbf{H}(\hat{\theta}^k)$  par  $\mathbf{H}^a(\hat{\theta}^k)$  dans Newton, avec

$$\mathbf{H}^a(\theta) = \sum_{i=1}^n w_i \frac{\partial e(\theta, i)}{\partial \theta} \frac{\partial e(\theta, i)}{\partial \theta^\top}$$

↔ **matrice d'information de Fisher** (← précision sur  $\hat{\theta}$ )

Utilise seulement les fonctions de sensibilité déjà utilisées pour calculer  $\mathbf{g}(\hat{\theta}^k)$  !

Approximation de  $\mathbf{H}$  par  $\mathbf{H}^a$  d'autant plus justifiée que les erreurs  $e(\theta, i)$  sont petites, et leur dérivée seconde faible (modèle presque linéaire)

Si le modèle est localement identifiable en  $\hat{\theta}^k$ ,  $\mathbf{H}^a(\hat{\theta}^k) \succ \mathbf{O}$  → la direction proposée est une direction de descente

**Très efficace !**

## 4.2) Levenberg-Marquardt:

Newton  $\rightarrow$  résoudre  $\mathbf{H}(\hat{\theta}_k) \Delta \theta = -\mathbf{g}(\hat{\theta}_k)$

$(\mathbf{H}^a(\hat{\theta}_k) \Delta \theta = -\mathbf{g}(\hat{\theta}_k))$  pour Gauss-Newton)

Levenberg (1944) Marquardt (1963): remplacer par

$$[\mathbf{H}(\hat{\theta}_k) + \mu_k \mathbf{I}^p] \Delta \theta = -\mathbf{g}(\hat{\theta}_k), \mu_k \geq 0$$

(revient à ajouter la pénalité  $(\mu_k/2) \|\theta - \hat{\theta}_k\|_2^2$  à  $j(\theta)$ )

$\mu_k = 0$ : Newton (ou Gauss-Newton) non relaxé

$\mu_k \rightarrow \infty$ : gradient avec pas  $1/\mu_k \rightarrow 0$

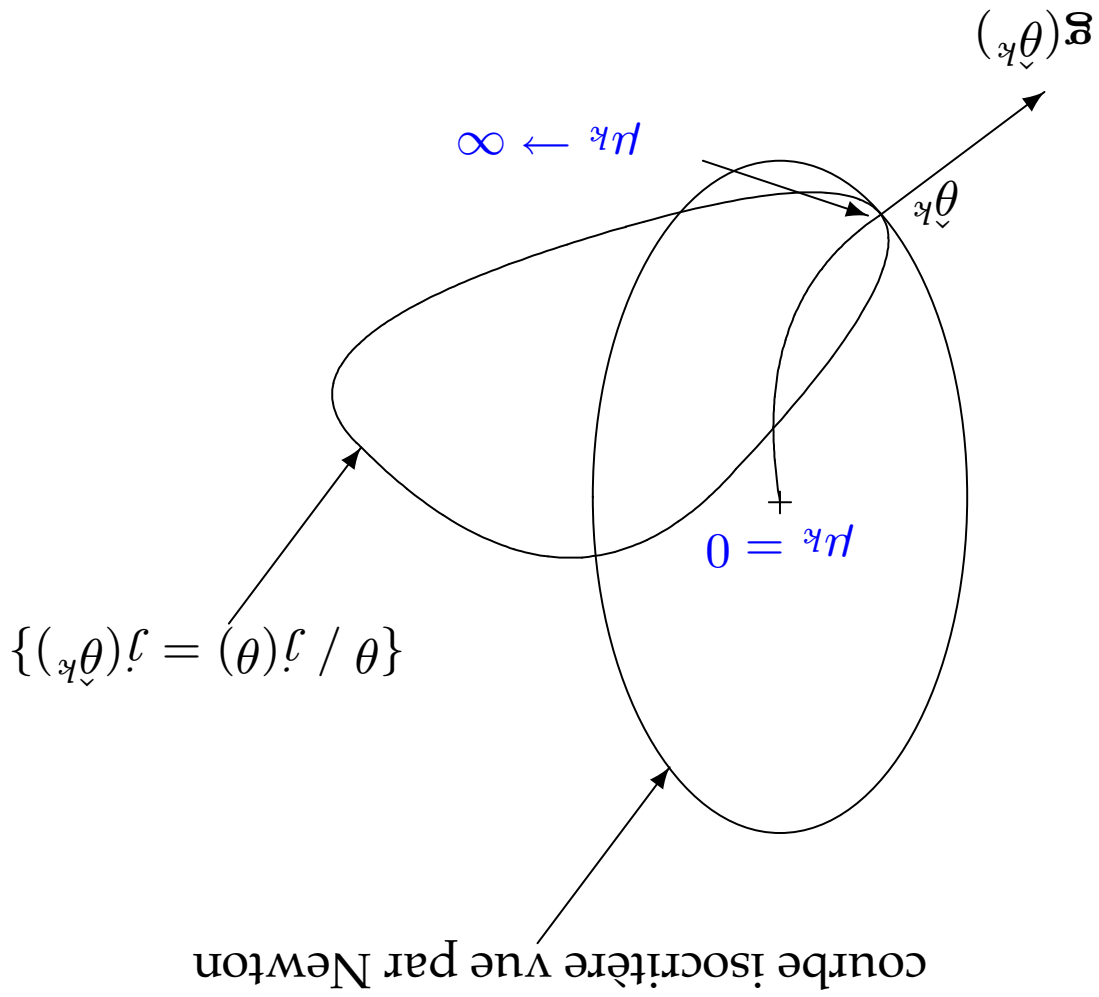
Adapter  $\mu_k$ :

si  $j(\hat{\theta}_{k+1}) < j(\hat{\theta}_k)$  diviser  $\mu_k$  par 10 (tout va bien...)

si  $j(\hat{\theta}_{k+1}) > j(\hat{\theta}_k)$ , rejeter  $\hat{\theta}_{k+1}$ , essayer avec  $10\mu_k$  (le gradient finit

toujours par marcher)

Pour chaque  $\mu_k$ , résoudre  $\mathbf{H}(\hat{\theta}_k) + \mu_k \mathbf{I}^d \Delta \theta = -\mathbf{g}(\hat{\theta}_k)$  ?



## 5) Quasi-Newton, gradients conjugués

← Diagonaliser  $\mathbf{H}(\hat{\theta}^k) : \mathbf{H}(\hat{\theta}^k) = \mathbf{T}\mathbf{\Lambda}\mathbf{T}^\top$ ,

avec  $\mathbf{T}\mathbf{T}^\top = \mathbf{I}_p$  et  $\mathbf{\Lambda} = \text{diag}\{\lambda_i, i = 1, \dots, p\}$ , puis

$$\Delta\theta = -\mathbf{T}\text{diag}\{\lambda_i + \mu_k\}^{-1} \mathbf{T}^\top \mathbf{g}(\hat{\theta}^k)$$

### 5.1) Quasi-Newton

Essayer de combiner les avantages du gradient (peu de calculs) et de Newton (convergence rapide — si convergence !)

→ construire une approximation de  $\mathbf{H}^{-1}$  sans calculer de dérivées

secondes...

Supposons  $j(\theta)$  quadratique en  $\theta$  :

$$j(\theta) = j(\hat{\theta}^k) + \mathbf{g}^\top(\hat{\theta}^k)(\theta - \hat{\theta}^k) + \frac{1}{2}(\theta - \hat{\theta}^k)^\top \mathbf{H}(\hat{\theta}^k)(\theta - \hat{\theta}^k)$$

$j(\theta)$  quadratique  $\Leftrightarrow \mathbf{H}$  ne dépend pas de  $\theta$

$$\mathbf{g}(\hat{\theta}_{k+1}) = \mathbf{g}(\hat{\theta}_k) + \mathbf{H}(\hat{\theta}_{k+1} - \hat{\theta}_k)$$

soit,  $\Delta \mathbf{g}_k = \mathbf{H} \Delta \theta_k$

Si  $M_k$  est l'approximation courante de  $\mathbf{H}^{-1}$ , quasi-Newton relaxé :

$$\hat{\theta}_{k+1} - \hat{\theta}_k = \lambda^k M^k \mathbf{g}(\hat{\theta}_k)$$

avec  $\lambda_k$  obtenu par optimisation unidimensionnelle (approximation polynomiale par ex.)

Nouvelle approximation de  $\mathbf{H}^{-1}$  :  $M_{k+1} = M_k + C_k$

avec  $C_k$  matrice de correction

Calcul d'une correction de rang 1 :

Si  $M_{k+1}$  était l'inverse de  $\mathbf{H} \rightarrow M_{k+1} \Delta \mathbf{g}_k = \Delta \theta_k$ , ou encore

$$(M_k + C_k) \Delta \mathbf{g}_k = \Delta \theta_k$$



$$C^k \Delta \mathbf{g}^k = \Delta \theta^k - M^k \Delta \mathbf{g}^k = \frac{(\Delta \theta^k - M^k \Delta \mathbf{g}^k)(\Delta \theta^k - M^k \Delta \mathbf{g}^k)^\top}{(\Delta \theta^k - M^k \Delta \mathbf{g}^k)(\Delta \theta^k - M^k \Delta \mathbf{g}^k)^\top} \overbrace{\Delta \mathbf{g}^k}^{C^k}$$

On obtient

$C^k$  symétrique et de rang 1

Initialisation :  $M_0 = \mathbf{I}^p \Leftrightarrow$  gradient

En pratique, correction de rang 2,  $\exists$  différentes formules :  
Davidon-Fletcher-Powell (DFP) :

$$C^k = \frac{\Delta \theta^k \Delta \theta^k{}^\top}{M^k \Delta \mathbf{g}^k \Delta \mathbf{g}^k M^k} - \frac{\Delta \theta^k \Delta \mathbf{g}^k{}^\top}{M^k \Delta \mathbf{g}^k \Delta \mathbf{g}^k M^k}$$

ou Broyden-Fletcher-Goldfarb-Shanno (BFGS) :

$$C^k = \left( 1 + \frac{\Delta \mathbf{g}^k{}^\top M^k \Delta \mathbf{g}^k}{\Delta \theta^k \Delta \theta^k{}^\top} \right) \frac{\Delta \theta^k \Delta \theta^k{}^\top}{\Delta \theta^k \Delta \theta^k{}^\top} - \frac{\Delta \theta^k \Delta \mathbf{g}^k{}^\top}{\Delta \theta^k \Delta \theta^k{}^\top} \frac{\Delta \mathbf{g}^k \Delta \theta^k{}^\top}{\Delta \theta^k \Delta \theta^k{}^\top} + M^k + M^k \frac{\Delta \mathbf{g}^k \Delta \theta^k{}^\top}{\Delta \theta^k \Delta \theta^k{}^\top} \frac{\Delta \theta^k \Delta \mathbf{g}^k{}^\top}{\Delta \theta^k \Delta \theta^k{}^\top} M^k$$

PQÑ1 : Calculs simples (seulement le gradient)

PQÑ2 : Si  $j(\theta)$  quadratique, convergence en  $p$  itérations (Newton : 1 itération, gradient :  $\infty$  itérations !)

PQÑ3 : Accumulation d'erreurs  $\rightarrow M_k$  peut devenir singulière  $\rightarrow$

ré-initialiser  $M_k$  par  $I_p$  (gradient) [peut être fait chaque  $p$  itérations]

PQÑ4 :  $M_k \rightarrow$  caractérisation de la précision de l'estimation

PQÑ5 : Domaine de convergence = bassin d'attraction (comme gradient)

PQÑ6 : Convergence lente au début, puis s'accélère (superlinéaire, et même quadratique si  $H(\theta)$  satisfait une condition de Lipschitz au

voisinage de l'optimum)

PQÑ7 : Métrique variable

## 5.2) Gradients conjugués

Essayer de combiner les avantages du gradient (peu de calculs) et de Newton (convergence rapide — si convergence !)

Ne pas manipuler de matrice (utile si  $p$  très grand)

Supposons de nouveau  $j(\theta)$  quadratique

$$j(\theta) = \frac{1}{2} (\hat{\theta}_k - \theta)^\top \mathbf{H} (\hat{\theta}_k - \theta) + \mathbf{g}^\top (\hat{\theta}_k - \theta)$$

$\hat{\theta}_{k+1}$  obtenu par minimisation dans une certaine direction  $\mathbf{d}_k$  :

$$\hat{\theta}_{k+1} = \hat{\theta}_k + \lambda_k \mathbf{d}_k$$

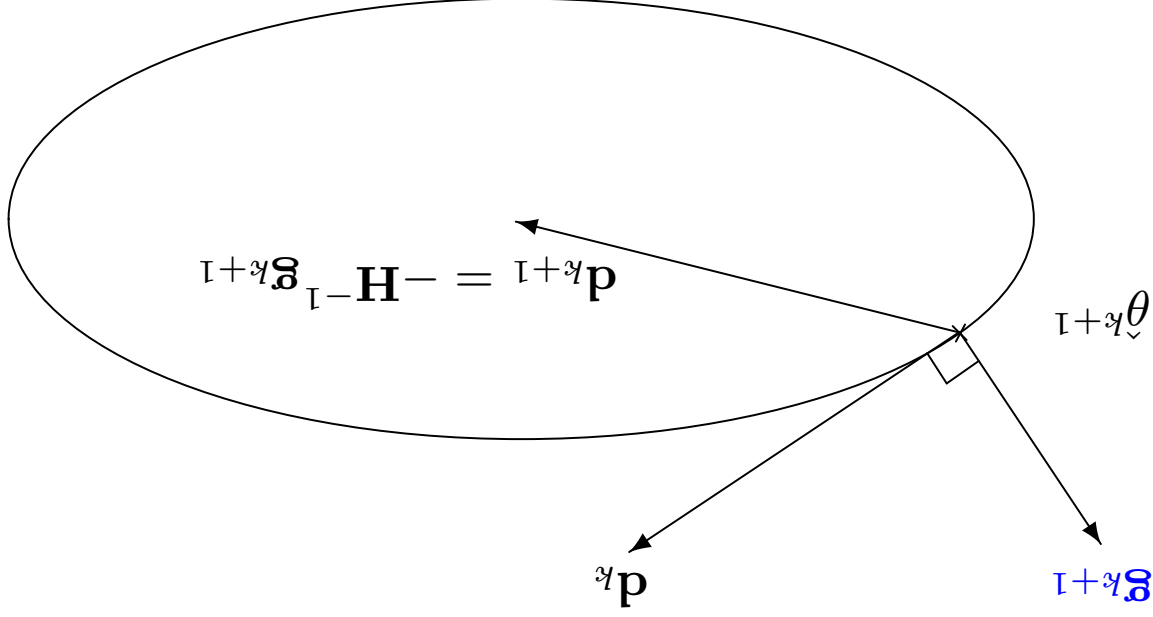
Minimisation précise suivant  $\mathbf{d}_k \Leftrightarrow \mathbf{g}_{k+1} = \mathbf{g}^\top (\hat{\theta}_{k+1})^\perp \mathbf{d}_k$ , soit

$$\mathbf{g}_{k+1}^\perp \mathbf{d}_k = 0$$

Direction suivante  $\mathbf{d}^{k+1}$  : celle de Newton  $\mathbf{d}^{k+1} = -\mathbf{H}^{-1} \mathbf{g}^{k+1}$ , soit  $\mathbf{g}^{k+1} = -\mathbf{H} \mathbf{d}^{k+1}$ , et donc  $\mathbf{d}_\perp^{k+1} \mathbf{H} \mathbf{d}^k = 0$ .

On dit que les directions  $\mathbf{d}^k$  et  $\mathbf{d}^{k+1}$  sont conjuguées par rapport à  $\mathbf{H}$

← générer des directions mutuellement conjuguées



$\mathbf{d}_0 = -\mathbf{g}_0$  (gradient), puis (Polak-Ribière)

$$\mathbf{d}_{k+1} = -\mathbf{g}_{k+1} + \frac{\mathbf{g}_k^\top \mathbf{g}_k}{(\mathbf{g}_{k+1} - \mathbf{g}_k)^\top \mathbf{g}_{k+1}} \mathbf{d}_k$$

avec  $\mathbf{g}_{k+1} = \mathbf{g}(\hat{\theta}_{k+1})$  et  $\hat{\theta}_{k+1}$  obtenu par minimisation dans la direction  $\mathbf{d}_k$  en partant de  $\hat{\theta}_k$

$$\hat{\theta}_{k+1} = \hat{\theta}_k + \lambda_k \mathbf{d}_k, \quad \lambda_k = \arg \min_{\lambda} j(\hat{\theta}_k + \lambda \mathbf{d}_k)$$

Si  $j(\theta)$  quadratique, et  $\mathbf{H}$  connu, alors  $\lambda_k = -\frac{\mathbf{g}_k^\top \mathbf{d}_k}{\mathbf{d}_k^\top \mathbf{H} \mathbf{d}_k}$ ,  $\mathbf{g}_{k+1} = \mathbf{g}_k + \lambda_k \mathbf{H} \mathbf{d}_k$  et on montre que les directions  $\mathbf{d}_k$  générées satisfont  $\mathbf{d}_i^\top \mathbf{H} \mathbf{d}_j = 0, \forall i \neq j$

**PGC1** : Calculs simples (seulement le gradient)

**PGC2** : Si  $J(\theta)$  quadratique, convergence en  $p$  itérations (Newton : 1 itération, gradient :  $\infty$  itérations !)

**PGC3** : Accumulation d'erreurs  $\rightarrow$   $\rightarrow$  ré-initialiser  $d_k$  par  $-\mathbf{g}_k$  (gradient)  
[peut être fait chaque  $p$  itérations]

**PGC4** : Pas de matrice à manipuler (mais pas de caractérisation de la précision...), très utile en grandes dimensions

**PGC5** : Plus d'itérations que quasi-Newton, mais itérations plus simples

**PGC6** : Les minimisations unidimensionnelles doivent être précises (pour avoir la propriété de conjugaison)

## 6) Optimisation à une dimension (2) : le retour

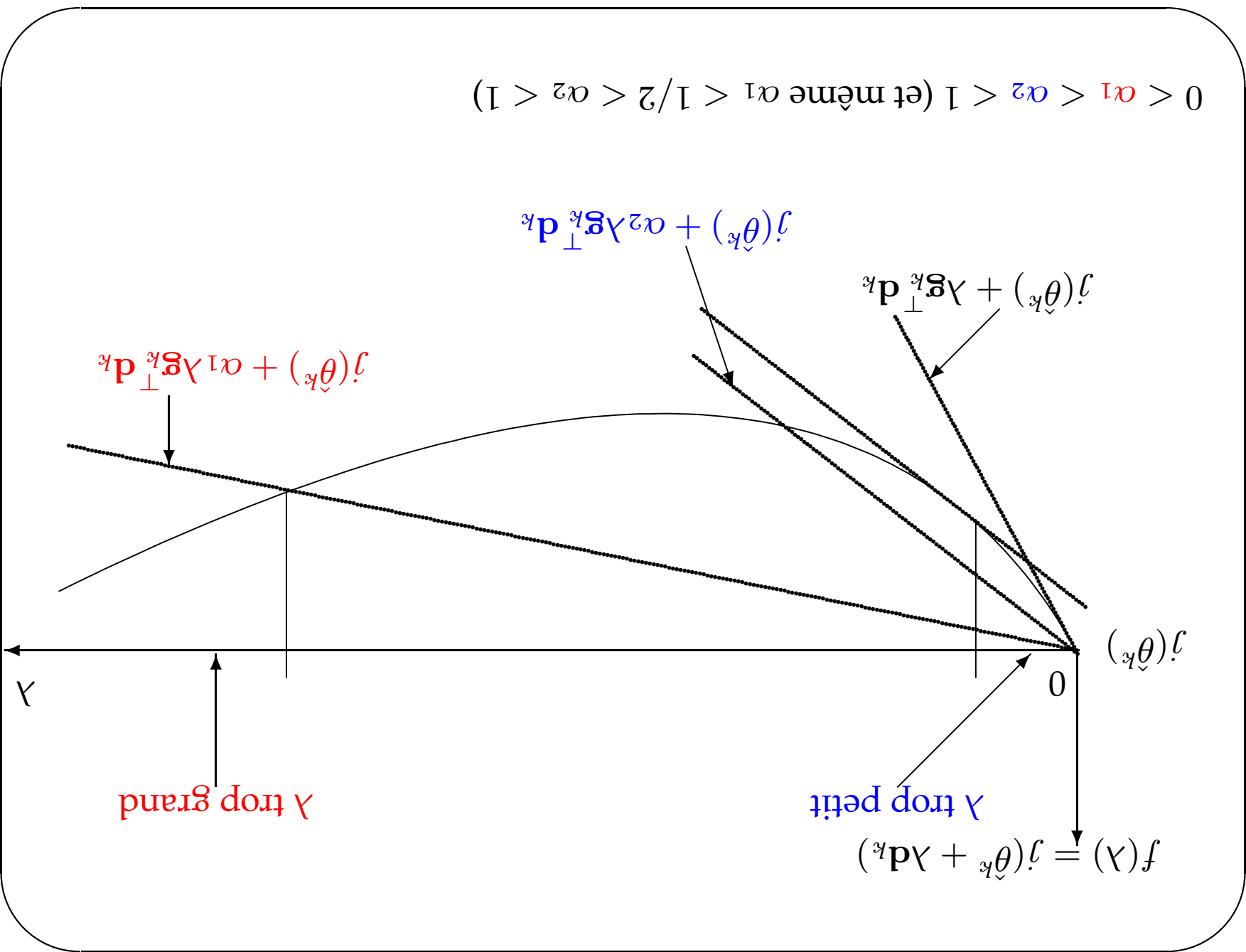
... ou plutôt, choix du pas

Rien ne sert d'optimiser trop précisément dans la direction proposée (sauf pour Powell et gradients conjugués) → seulement assurer une décroissance significative de  $j(\theta)$

**Rq :** Pour Newton, Gauss-Newton, quasi-Newton, un choix raisonnable pour le pas est 1

Méthode de Wolfe :

→ Rejeter les pas **trop petits** et les pas **trop grands**





## Algorithme :

- 1) Choisir  $\lambda_1 > 0, \alpha_1, \alpha_2$ , avec  $0 < \alpha_1 < \alpha_2, \lambda_{\min} = \lambda_{\max} = 0, i = 1$
- 2) Si  $j(\hat{\theta}_k + \lambda_i \mathbf{d}_k) + \lambda_i \mathbf{d}_k^T (\alpha_1 \lambda_i \mathbf{g}_k^T \mathbf{d}_k + \alpha_2 \mathbf{g}_k^T \mathbf{d}_k) > j(\hat{\theta}_k) + \lambda_i \mathbf{d}_k^T \mathbf{g}_k$ ,  $\lambda_i$  trop grand :  $\lambda_{\max} = \lambda_i$ , aller au pas 5
- 3) Si  $\mathbf{g}_k^T (\hat{\theta}_k + \lambda_i \mathbf{d}_k) + \lambda_i \mathbf{d}_k^T \mathbf{g}_k \geq \alpha_2 \mathbf{g}_k^T \mathbf{d}_k$ ,  $\lambda_i$  est acceptable :  
 $\hat{\theta}_{k+1} = \hat{\theta}_k + \lambda_i \mathbf{d}_k$ , stop  $\rightarrow$  direction suivante

Si non,  $\lambda_i$  trop petit,  $\lambda_{\min} = \lambda_i$

- 4) Si  $\lambda_{\max} = 0, \lambda_{i+1} = 2\lambda_i, i \rightarrow i + 1$ , aller au pas 2
- 5)  $\lambda_{i+1} = (\lambda_{\min} + \lambda_{\max})/2, i \rightarrow i + 1$ , aller au pas 2

Rq :  $\exists$  autres méthodes ne calculant pas d'autre gradient que  $\mathbf{g}_k$

Goldstein et Price :

$$j(\hat{\theta}_k) + \alpha_2 \lambda_i \mathbf{g}_k^T \mathbf{d}_k \leq j(\hat{\theta}_k) + \lambda_i \mathbf{d}_k^T \mathbf{g}_k \leq j(\hat{\theta}_k) + \alpha_1 \lambda_i \mathbf{g}_k^T \mathbf{d}_k \rightarrow \lambda_i \text{ est acceptable}$$
$$j(\hat{\theta}_k) + \lambda_i \mathbf{d}_k^T \mathbf{g}_k > j(\hat{\theta}_k) + \alpha_1 \lambda_i \mathbf{g}_k^T \mathbf{d}_k \rightarrow \lambda_i \text{ trop grand}$$
$$j(\hat{\theta}_k) + \lambda_i \mathbf{d}_k^T \mathbf{g}_k > j(\hat{\theta}_k) + \alpha_2 \lambda_i \mathbf{g}_k^T \mathbf{d}_k \rightarrow \lambda_i \text{ trop petit}$$

## 7) Optimisation sous contraintes

Modification du critère (pénalisation) : déjà vu

→ Modification de l'algorithme. On notera  $S$  ensemble des points acceptables (qui satisfont les contraintes, ici uniquement inégalités)

**Beaucoup plus difficile que sans contraintes**

7.1) Programmation linéaire

Minimiser  $c^T \theta$  (linéaire en  $\theta$ ) sous les contraintes (linéaires en  $\theta$ )

$$A\theta \leq b$$

c'est-à-dire,  $a_i^T \theta \leq b_i, i = 1, \dots, m$  ( $S$  est un polyèdre)

Problème rencontré pour l'estimation de norme  $L_1$  et  $L_\infty$ ...

∃ algorithme spécifique et classique : simplexe (Dantzig, 1963)

∃ autres algorithmes (ellipsoïdes, points intérieurs), suite à (Khachiyan, 1979), voir + loin

## 7.2) Programmation quadratique

Contraintes linéaires,  $\mathbf{A}\theta \leq \mathbf{b}$  ( $\mathcal{S}$  est un polyèdre), mais objectif quadratique

$$j(\theta) = \frac{1}{2}\theta^\top \mathbf{C}\theta - \mathbf{d}^\top \theta$$

avec  $\mathbf{C} \succ \mathbf{0}$  ( $\Leftrightarrow j(\cdot)$  est convexe)

Algorithme «standard»: rechercher le minimum de  $j(\cdot)$  sous les contraintes *actives* prises comme des contraintes égales

Soient  $\mathbf{A}_k$  et  $\mathbf{b}_k$  les parties de  $\mathbf{A}$  et  $\mathbf{b}$  correspondant aux contraintes actives  $\leftarrow$  Lagrangien

$$\mathcal{L}(\theta, \mathbf{z}) = \frac{1}{2}\theta^\top \mathbf{C}\theta - \mathbf{d}^\top \theta + \mathbf{z}^\top (\mathbf{A}_k \theta - \mathbf{b}_k)$$

Stationnarité par rapport à  $\theta$  et  $\mathbf{z}$ :

$$\begin{pmatrix} \mathbf{C} & \mathbf{A}_k^\top \\ \mathbf{A}_k & \mathbf{0} \end{pmatrix} \begin{pmatrix} \hat{\theta} \\ \hat{\mathbf{z}} \end{pmatrix} = \begin{pmatrix} \mathbf{d} \\ \mathbf{b}_k \end{pmatrix}$$

- ☞ Si  $\mathbf{A}\hat{\theta}_{k+} \leq \mathbf{b}$  et  $\hat{\mathbf{z}} \geq 0$ ,  $\hat{\theta}_{k+}$  est solution
  - ☞ Si  $\hat{\theta}_{k+}$  viole des contraintes,
    - rechercher sur le segment  $\hat{\theta}_k \rightarrow \hat{\theta}_{k+}$  pour trouver un nouveau point acceptable  $\hat{\theta}_{k+1}$
    - introduire les nouvelles contraintes égalités
  - ☞ si des  $z_i$  sont  $> 0$ , éliminer les contraintes associées (inactives)
- Ici aussi,  $\exists$  autres algorithmes (ellipsoïdes, points intérieurs), suite à (Khachiyan, 1979), voir + loin

### 7.3) Gradient contraint

Algorithme du gradient:  $\hat{\theta}_{k+1} = \hat{\theta}_k - \lambda_k \mathbf{g}(\hat{\theta}_k)$ , avec  $\lambda_k$  choisi pour minimiser  $j(\hat{\theta}_{k+1})$

Problème :  $\mathbf{g}(\hat{\theta}_k)$  peut pointer dans une direction inacceptable (sortant de  $S$ )

← remplacer  $\mathbf{g}(\hat{\theta}_k)$  par  $\mathbf{d}_k = \hat{\theta}_{k+} - \hat{\theta}_k$  avec

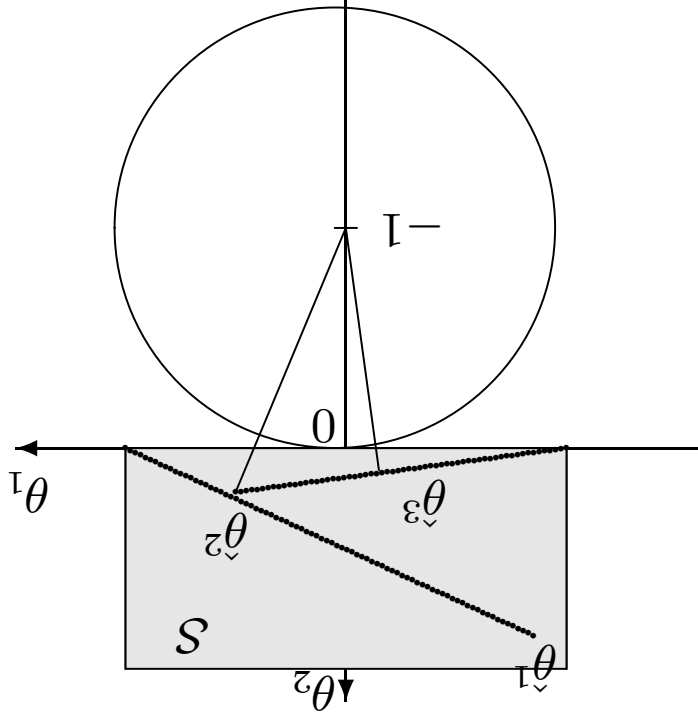
$$\hat{\theta}_{k+} = \arg \min_{\theta \in S} \mathbf{g}^{\top}(\hat{\theta}_k)(\theta - \hat{\theta}_k)$$

mauvaise idée !

Ex1 :

$$j(\theta) = \theta_2^2 + (1 + \theta_2)^2, \theta = (\theta_1, \theta_2)^{\top}, S = \{\theta \mid |\theta_1| \leq 1, 0 \leq \theta_2 \leq 1\}$$

← isocritères : cercles centré en  $(0, -1)$



Convergence vers l'optimum  $(0,0)^T$   
 en **une**  $\infty$  itérations !

### 7.4) Gradient projeté

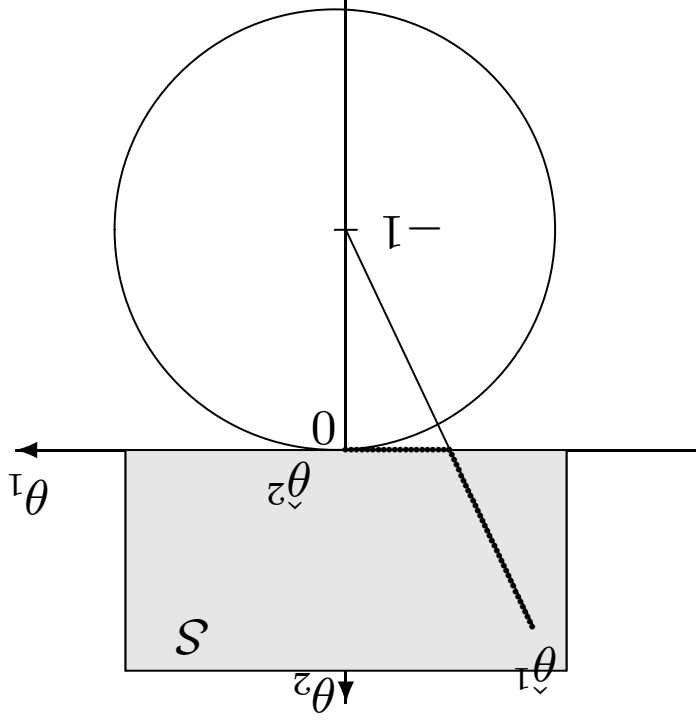
$$\hat{\theta}^{k+1}(\lambda) = \arg \min_{\theta \in S} \|\theta - [\hat{\theta}^k - \lambda \mathbf{g}(\hat{\theta}^k)]\|_2$$

puis minimisation de  $j[\hat{\theta}^{k+1}(\lambda)]$  par rapport à  $\lambda$

Retour à Ex1 ...

$$\left. \begin{array}{l} \hat{\theta}_{k+1}^i(\lambda) = a_i \\ \hat{\theta}_{k+1}^i(\lambda) = b_i \\ \hat{\theta}_{k+1}^i(\lambda) - \lambda g_i(\hat{\theta}_k) = a_i \leq \hat{\theta}_k^i - \lambda g_i(\hat{\theta}_k) \\ \hat{\theta}_{k+1}^i(\lambda) - \lambda g_i(\hat{\theta}_k) = b_i > \hat{\theta}_k^i - \lambda g_i(\hat{\theta}_k) \\ \hat{\theta}_{k+1}^i(\lambda) - \lambda g_i(\hat{\theta}_k) = a_i < \hat{\theta}_k^i - \lambda g_i(\hat{\theta}_k) \\ \hat{\theta}_{k+1}^i(\lambda) - \lambda g_i(\hat{\theta}_k) = b_i > \hat{\theta}_k^i - \lambda g_i(\hat{\theta}_k) \\ \hat{\theta}_{k+1}^i(\lambda) - \lambda g_i(\hat{\theta}_k) = a_i \leq \hat{\theta}_k^i - \lambda g_i(\hat{\theta}_k) \\ \hat{\theta}_{k+1}^i(\lambda) - \lambda g_i(\hat{\theta}_k) = b_i > \hat{\theta}_k^i - \lambda g_i(\hat{\theta}_k) \end{array} \right\}$$

Très pratique quand  $S$  est un hyper-rectangle (orthotope),  
 $S = \{ \theta^i / a_i \leq \theta^i \leq b_i \mid i = 1, \dots, d \}$  ← «simple saturation»



Convergence vers l'optimum  $(0,0)^T$   
 en **une** itération !

## 7.5) Méthodes du second ordre : programmation quadratique séquentielle

Approximation quadratique de  $j(\cdot)$  :

$$\hat{\theta}_{k+1} = \arg \min_{\theta \in S} \left[ (\theta - \hat{\theta}_k)^\top \mathbf{g}(\hat{\theta}_k) + \frac{1}{2} (\theta - \hat{\theta}_k)^\top \mathbf{H}(\hat{\theta}_k) (\theta - \hat{\theta}_k) \right]$$

avec  $\mathbf{H}(\theta)$  le Hessien en  $\theta$  (ou une approximation construite récursivement pour quasi-Newton)

puis  $\hat{\theta}_{k+1} = \hat{\theta}_{k+1}$

ou

$$\hat{\theta}_{k+1} = \arg \min_{\lambda} [\hat{\theta}_k + \lambda(\hat{\theta}_{k+1} - \hat{\theta}_k)]$$

Si  $S$  est un polyèdre : détermination de  $\hat{\theta}_{k+1}$  = programmation quadratique  
→ programmation quadratique séquentielle

Si  $S$  n'est pas un polyèdre (contraintes non linéaires), on les linéarise...



## 8) Critères non différentiables

Par exemple, estimation  $L_1$ ,  $j(\theta) = |\theta_1 - \theta_2| + 0.2|\theta_1 + \theta_2|$

En  $\hat{\theta}_1 = (1, 1)^\top$ ,  $j(\theta)$  augmente quand on se déplace suivant chacun des

axes  $e_1$  et  $e_2 \rightarrow$  la méthode de Powell est inapplicable !

### 8.1) Programmation convexe

$\Leftrightarrow S$  est convexe et  $j(\cdot)$  est convexe

#### 8.1.1) Sous-différentiel et sous-gradient

Sous-gradient de  $j(\cdot)$  en  $\theta$  : vecteur  $\mathbf{a} \in \mathbb{R}^p$  tel que

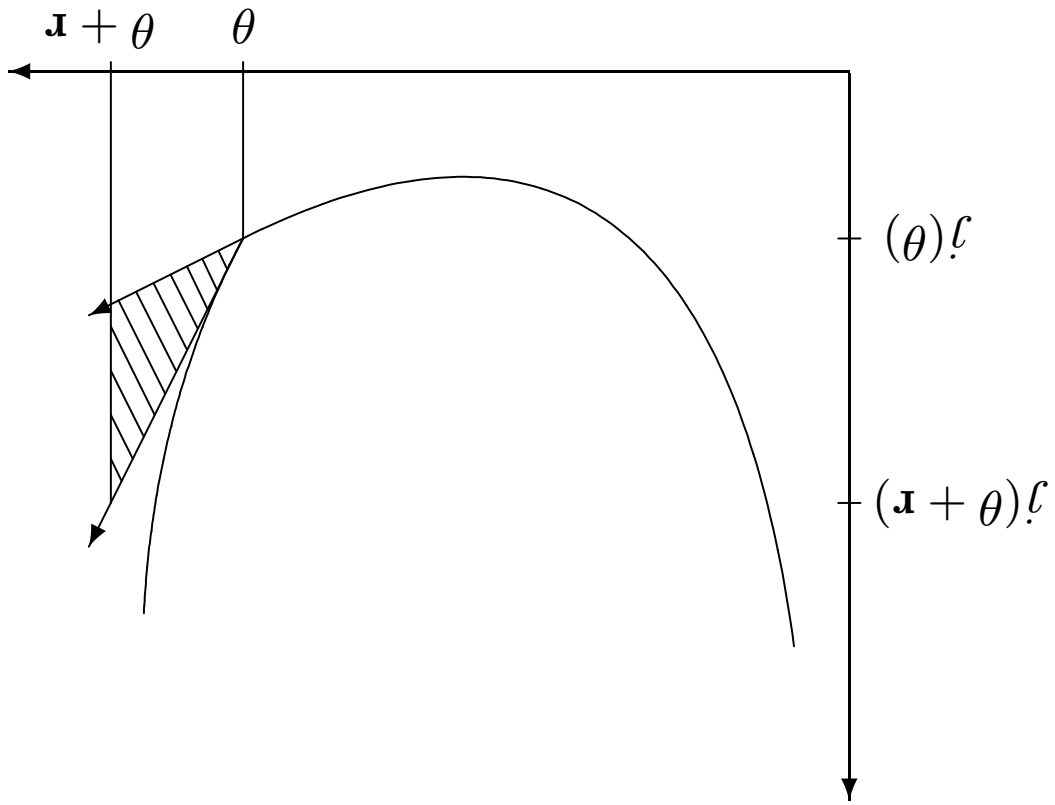
$$\forall \mathbf{r} \in \mathbb{R}^p, j(\theta + \mathbf{r}) \geq j(\theta) + \mathbf{a}^\top \mathbf{r}$$

Si  $j(\cdot)$  est différentiable en  $\theta$ , alors le sous-gradient en  $\theta$  est unique,  $\underline{\mathbf{g}}(\theta) = \mathbf{g}(\theta)$  gradient de  $j(\cdot)$  en  $\theta$

Sinon,  $\partial j(\theta) =$  sous-différentiel = ensemble des sous-gradients en  $\theta$

## Propriétés :

- ↪  $j(\cdot)$  convexe  $\Leftrightarrow \theta \in \partial j(\theta)$  minimise  $j(\cdot)$
- ↪  $j(\cdot)$  convexe  $\Leftrightarrow [\underline{g}(\theta_2) - \tilde{g}(\theta_1)]^\top (\theta_2 - \theta_1) \geq 0$
- ↪  $j(\cdot)$  convexe  $\Leftrightarrow \partial j(\theta)$  borné sur  $\{\theta / j(\theta) \leq \alpha\}$



➔ dérivée directionnelle (Fréchet)

$$g(\theta_1, \theta_2) = \lim_{\epsilon \rightarrow 0} \frac{j(\theta_1 + \epsilon \theta_2) - j(\theta_1)}{\epsilon} = \max_{\mathbf{a} \in \partial j(\theta_1)} \mathbf{a}^\top \theta_2$$

Quelques règles de calcul :  $j_i(\cdot)$  des fonctions convexes, de sous-gradients respectifs  $\tilde{\mathbf{g}}_i(\cdot)$

☞ si  $j(\theta) = \sum_{i=1}^m \alpha_i j_i(\theta)$ , alors  $\tilde{\mathbf{g}}(\theta) = \sum_{i=1}^m \alpha_i \tilde{\mathbf{g}}_i(\theta)$

☞ si  $j(\theta) = \max_{i=1, \dots, m} j_i(\theta)$ , alors  $\partial j(\theta)$  est l'enveloppe convexe de

$$\cup_{i \in \mathcal{I}(\theta)} \{\partial j_i(\theta)\} \text{ avec } \mathcal{I}(\theta) = \{i / j_i(\theta) = j(\theta)\}$$

### 8.1.2) Méthode du sous-gradient

$$\hat{\theta}_{k+1} = \hat{\theta}_k - \lambda_k \tilde{\mathbf{g}}(\hat{\theta}_k)$$

Problème :  $j(\cdot)$  ne décroît pas forcément dans la direction  $-\tilde{\mathbf{g}}(\hat{\theta}_k)$

⇒ impossible d'optimiser par rapport à  $\lambda$  !

Impossible aussi de garder  $\lambda$  constant : pour  $j(\theta) = |\theta|$ ,  $\theta \in \mathbb{R}$ ,  $|\tilde{\mathbf{g}}(\theta)| = 1$   
 $\forall \theta \neq 0$ , et  $|\hat{\theta}_{k+1} - \hat{\theta}_k| = \lambda \forall k$  !

← on fixe  $\lambda_k$  a priori :

$$\lambda_k \geq 0, \lim_{k \rightarrow \infty} \lambda_k = 0, \sum_{k=1}^{\infty} \lambda_k = \infty \quad (\lambda_k = \frac{k}{\alpha} \text{ convient})$$

mais convergence très lente...

### 8.1.3) Méthode du plan sécant

$$j(\cdot) \text{ convexe} \Leftrightarrow \forall \theta, j(\theta) \geq j(\hat{\theta}_k) + \tilde{\mathbf{g}}_{\perp}(\hat{\theta}_k)(\theta - \hat{\theta}_k)$$

On utilise une approximation inférieure de  $j(\cdot)$  donnée par

$$j_k(\theta) = \max_{i=1, \dots, k} j(\hat{\theta}_i) + \tilde{\mathbf{g}}_{\perp}(\hat{\theta}_i)(\theta - \hat{\theta}_i)$$

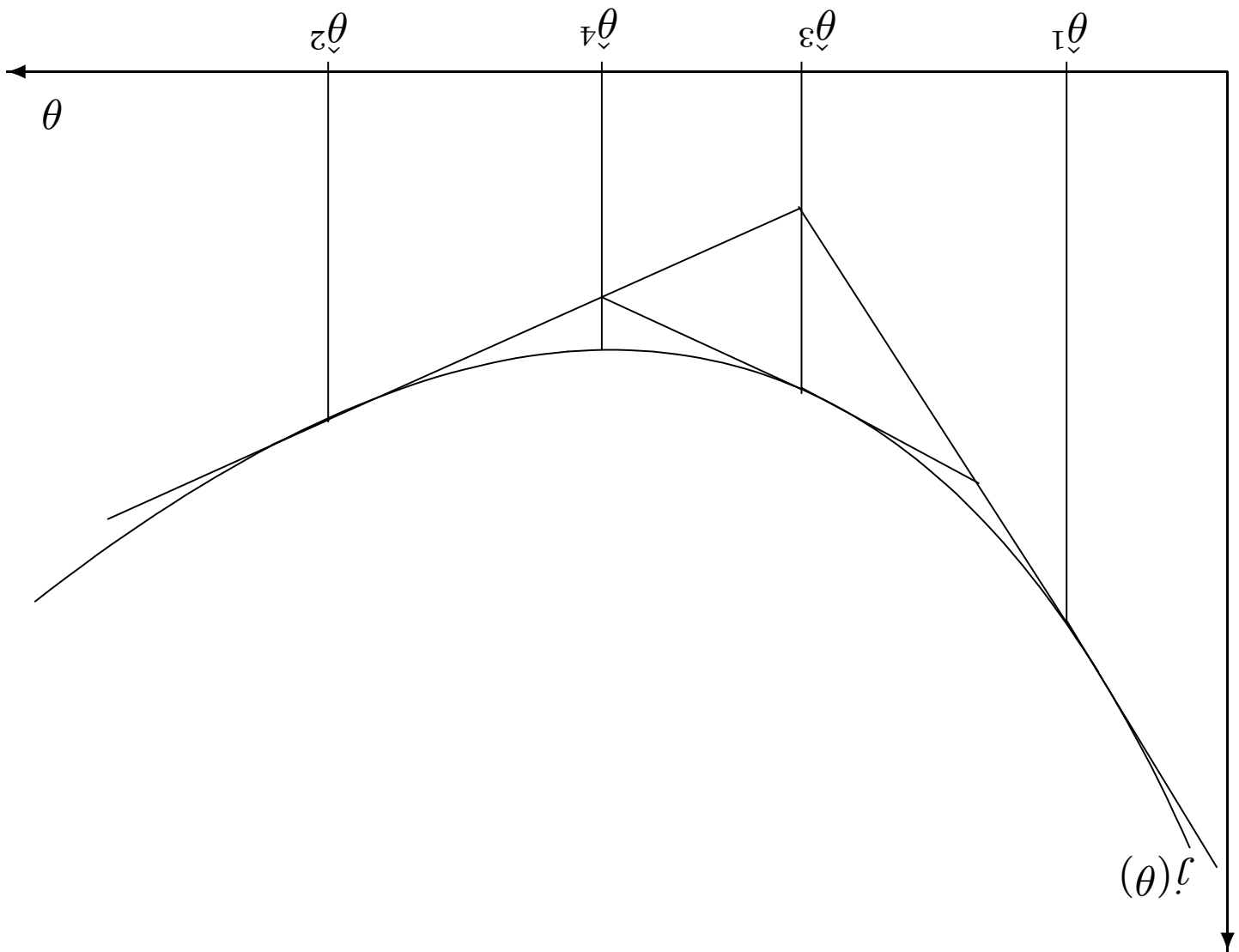
$$\text{et } \hat{\theta}_{k+1} = \arg \min_{\theta \in S} j_k(\theta)$$

Si  $S$  est un polyèdre  $\Leftrightarrow$  programmation linéaire dans  $\mathbb{R}^{p+1}$  :

Minimiser  $\alpha$  sous les contraintes

$$\theta \in S \text{ et } j(\hat{\theta}_i) + \tilde{\mathbf{g}}_{\perp}(\hat{\theta}_i)(\theta - \hat{\theta}_i) \leq \alpha, i = 1, \dots, k$$

Mais la convergence est parfois très lente...



## 8.1.4) Méthodes de coupe

(le plan sécant en fait partie...)

$j(\cdot)$  convexe  $\Rightarrow$  le minimum  $\hat{\theta}$  satisfait

$$\tilde{\mathbf{g}}_{\perp}(\hat{\theta}^k) - j(\hat{\theta}^k) \leq j(\hat{\theta}) - j(\hat{\theta}^k) \leq 0$$

$\leftarrow$  chaque évaluation de sous-gradient apporte une contrainte

supplémentaire

après  $k$  évaluations, on sait que

$$\hat{\theta} \in S^k = S \cap \{\theta / \tilde{\mathbf{g}}_{\perp}(\hat{\theta}^i) - \theta^i \leq 0, i = 1, \dots, k\}$$

point suivant  $\hat{\theta}^{k+1}$  choisi dans cet ensemble (centre de gravité, mais on ne sait pas faire, centre du plus grand ellipsoïde inscrit dans  $S^k$ ...)

## 8.1.5) Méthode des ellipsoïdes extérieurs

On part d'un grand ellipsoïde  $\mathcal{E}^0$  dont on est sûr qu'il contient  $\hat{\theta}$

Soit  $\mathcal{E}^k$  l'ellipsoïde de l'itération  $k$

Si son centre  $\hat{\theta}^k$  ne satisfait pas l'une des contraintes de  $S^{k-1}$ ,

on coupe par la contrainte la plus violée (*coupe profonde*),

on garde la partie de l'ellipsoïde qui satisfait la contrainte,

on détermine le plus petit ellipsoïde contenant cet ellipsoïde tronqué,

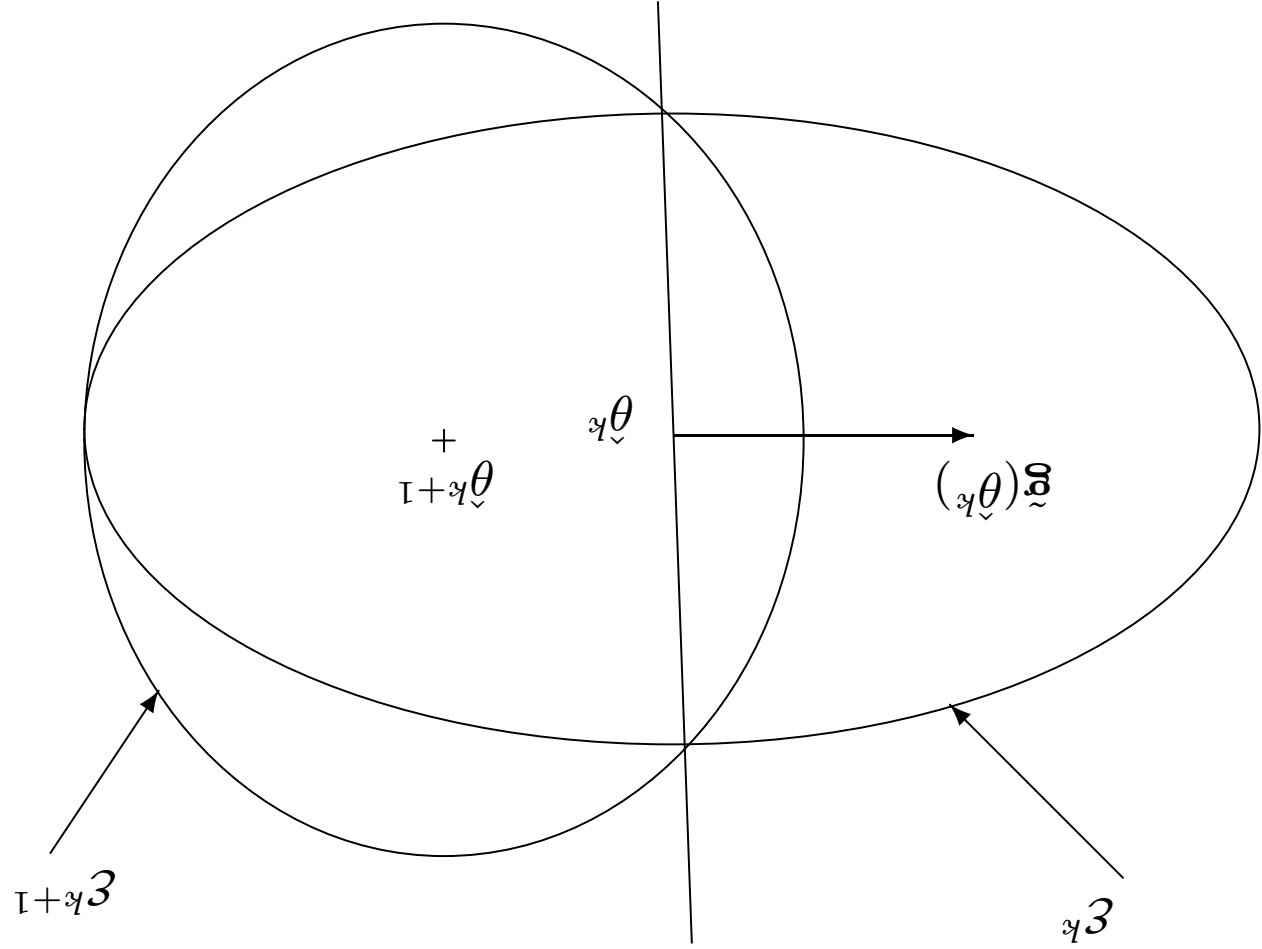
on répète autant que nécessaire

Quand le centre satisfait finalement toutes les contraintes, on coupe par le centre (*coupe centrale*) en utilisant  $\tilde{\mathbf{g}}_{\perp}(\hat{\theta}^k)(\hat{\theta} - \hat{\theta}^k)$

En 1979, Khachiyan → programmation linéaire de complexité polynomiale

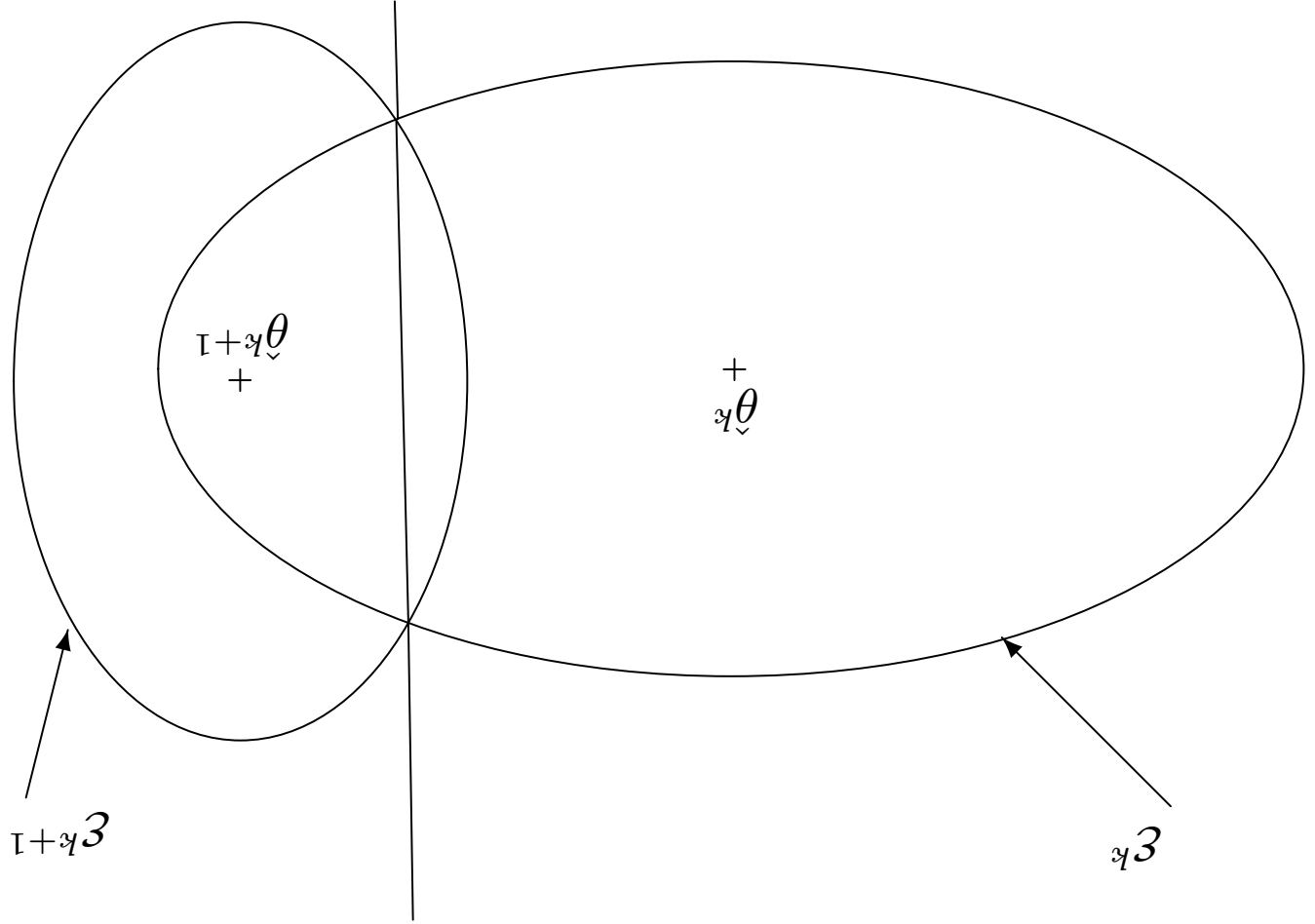
Depuis, méthodes de points intérieurs pour programmation convexe...

Coupe centrale





Coupe profonde



## 8.2) Estimation de norme $L_1$

Si modèle linéaire, déjà vu : programmation linéaire

Si modèle non-linéaire → solution séquentielle de problème convexes

$$j(\theta) = \sum_{i=1}^N w_i |y(i) - y_m(\theta, i)| \text{ avec } y_m(\theta, i) \text{ non linéaire en } \theta$$

0) choisir  $\hat{\theta}_0, k = 0$

1) calculer

$$\delta\hat{\theta}^k = \arg \min_{\delta\theta} \sum_{i=1}^N \left| y(i) - y_m(\hat{\theta}^k, i) - \frac{\partial y_m(\theta, i)}{\partial \theta^\top} \Big|_{\hat{\theta}^k} \delta\theta \right|$$

2) calculer

$$\lambda_k = \arg \min_{\lambda > 0} \sum_{i=1}^N w_i |y(i) - y_m(\hat{\theta}^k + \lambda \delta\hat{\theta}^k, i)|$$

$\hat{\theta}^{k+1} = \hat{\theta}^k + \lambda_k \delta\hat{\theta}^k, k \leftarrow k + 1$ , retourner en 1)

Il faut aussi un test d'arrêt (par exemple sur  $|j(\hat{\theta}_{k+1}) - j(\hat{\theta}_k)|$ )

Pas 1) : problème convexe (outils déjà vus)

Pas 2) : recherche unidimensionnelle (déjà vu aussi)

... mais il vaut mieux quand même rendre le critère différentiable →

*M*-estimateur de Huber

$$\rho_1(e) = \begin{cases} e^2/2 & \text{si } |e| \leq \delta \\ \delta|e| - \delta^2/2 & \text{sinon} \end{cases}$$

## 9) Techniques récursives

Maximum de vraisemblance  $\rightarrow$  erreur de prédiction  $e(\theta, i) \rightarrow$  maximiser

$$j_{MV}(\theta) = \frac{1}{N} \sum_{i=1}^N \log \pi[e(\theta, i) | \theta]$$

Comment faire pour traiter les données l'une après l'autre ?

On suppose que l'on en aura beaucoup :  $N \rightarrow \infty$ , et les  $e(\theta, i)$  sont

supposées i.i.d., alors

$$j_{MV}(\theta) \approx \mathbb{E}_e \{ \log \pi[e(\theta, 1) | \theta] \}$$

**Approximation stochastique** : on peut optimiser une espérance sans jamais

la calculer !

Gradient :

$$\hat{\theta}_{k+1} = \hat{\theta}_k + \lambda_k \frac{\partial \mathbb{E}_e \{ \log \pi[e(\theta, 1) | \theta] \}}{\partial \theta} \Big|_{\theta_k}$$

Gradient stochastique : on remplace  $E\{\cdot\}$  par une évaluation en un point pris au hasard  
 ici  $e(\theta, k+1)$  correspondant à l'observation suivante

$$\hat{\theta}_{k+1} = \hat{\theta}_k + \lambda_k \frac{\partial \log \pi[e(\theta, k+1) | \theta]}{\partial \theta} \quad \hat{\theta}_k = \hat{\theta}_k + \lambda_k \frac{\partial \pi[e(\theta, k+1) | \theta]}{\partial \theta}$$

Conditions sur  $\lambda_k$  :

$$\lambda_k > 0, \sum_{k=1}^{\infty} \lambda_k = \infty, \sum_{k=1}^{\infty} \lambda_k^2 < \infty$$

(un choix possible :  $\lambda_k = \alpha / (k+1)$ )

### Newton stochastique :

$$\hat{\theta}_{k+1} = \hat{\theta}_k + \frac{1}{k+1} \mathbf{M}_{k+1}^{-1}(\hat{\theta}_k, \xi_{k+1}^1) \frac{\partial \theta}{\partial \log \pi[e(\theta, k+1) | \theta]} \Big|_{\theta^k}$$

avec  $\mathbf{M}(\theta, \xi_{k+1}^1)$  la matrice d'information de Fisher (moyenne par

observation)

$$\mathbf{M}(\theta, \xi_{k+1}^1) = \mathbf{E} \left\{ \frac{1}{k+1} \sum_{i=1}^i \frac{\partial \theta}{\partial \log \pi[e(\theta, i) | \theta]} \frac{\partial \theta}{\partial \log \pi[e(\theta, i) | \theta]} \Big| \theta \right\}$$

**Ex :**  $y(i) = y_m(\theta, i) + \epsilon_i$ , avec  $(\epsilon_i)_i$  i.i.d.  $\mathcal{N}(0, \sigma^2)$ . Alors

$$\mathbf{M}(\theta, \xi_{k+1}^1) = \frac{1}{k+1} \sum_{i=1}^i \frac{\partial \theta}{\partial y_m(\theta, i)} \frac{\partial \theta}{\partial y_m(\theta, i)} = \frac{1}{k+1} \frac{\sigma^2}{1}$$

Si  $y_m(\theta, i)$  linéaire en  $\theta$ ,  $y_m(\theta, i) = \mathbf{r}_\perp^\top(i)\theta$ ,  $M(\theta, \xi_{k+1}^1)$  ne dépend pas de  $\theta$  et peut être calculé récursivement :

$$M(\theta, \xi_{k+1}^1) = \frac{k}{k+1} M(\theta, \xi_k^1) + \frac{\mathbf{r}(k+1)\mathbf{r}_\perp^\top(k+1)\sigma_2}{k+1}$$

← redonne exactement l'algorithme des moindres carrés récursifs !

L'algorithme du gradient stochastique donne dans ce cas :

$$\hat{\theta}_{k+1} = \hat{\theta}_k + \lambda_k \mathbf{r}(k+1)[y(k+1) - \mathbf{r}_\perp^\top(k+1)\hat{\theta}_k]$$

appelé algorithme *Least Mean Squares*

Si  $y_m(\theta, i)$  n'est pas linéaire en  $\theta$ , on peut tout de même approximer  $M(\theta, \xi_{k+1}^1)$  récursivement par

$$M_a(\xi_{k+1}^1) = \frac{k+1}{k} M_a(\xi_k^1) + \frac{1}{1+k} \frac{\partial}{\partial \theta} \frac{\partial}{\partial y_m(\theta, i)} \bigg|_{\hat{\theta}_k} + \frac{1}{1+k} \frac{\partial}{\partial \theta} \frac{\partial}{\partial y_m(\theta, i)} \bigg|_{\hat{\theta}_k}$$

et on peut éviter l'inversion de  $M_a(\xi_{k+1}^1) \leftarrow$  «maximum de vraisemblance approche»

## 10) Optimisation globale

On a déjà vu comment supprimer la présence d'optima locaux en répétant des observations sous les mêmes conditions expérimentales  
Mais si les observations sont déjà là .. il faut un algorithme permettant d'échapper aux optimas locaux

Technique simple à mettre en œuvre : recherche aléatoire

- 0) choisir  $\theta^0, k = 0$
- 1) générer  $\hat{\theta}_{k+} \in S : \hat{\theta}_{k+} = \theta_k + r_k$ , avec  $r_k \sim \mathcal{N}(0, \Sigma)$  (on essaie de nouveau tant que  $\hat{\theta}_{k+} \notin S$ )
- 2) si  $j(\hat{\theta}_{k+}) < j(\hat{\theta}_k)$ , accepter  $\hat{\theta}_{k+1} = \hat{\theta}_{k+}$  ;  
sinon  $\hat{\theta}_{k+1} = \hat{\theta}_k$

Difficulté : comment choisir  $\Sigma$  ?

← tirages proches si  $\hat{\theta}_k$  proche de l'optimum  $\theta$ , tirages éloignés sinon...

idée : adapter  $\Sigma \leftarrow$  recherche aléatoire adaptative



On suppose  $S = \text{orthotope } \{ \theta / a_i \leq \theta_i \leq b_i, i = 1, \dots, p \}$  (ou contenu dans un tel orthotope)

☞  $\theta_0$  au centre:  $\hat{\theta}_0^i = (a_i + b_i)/2, i = 1, \dots, p$

☞ 5 valeurs de  $\Sigma$ :  $\Sigma_1 = \text{diag}\{b_i - a_i, i = 1, \dots, p\}$ , puis  $\Sigma_{j+1} = 0.1\Sigma_j, j = 1, \dots, 4$

☞ On alterne 2 phases :

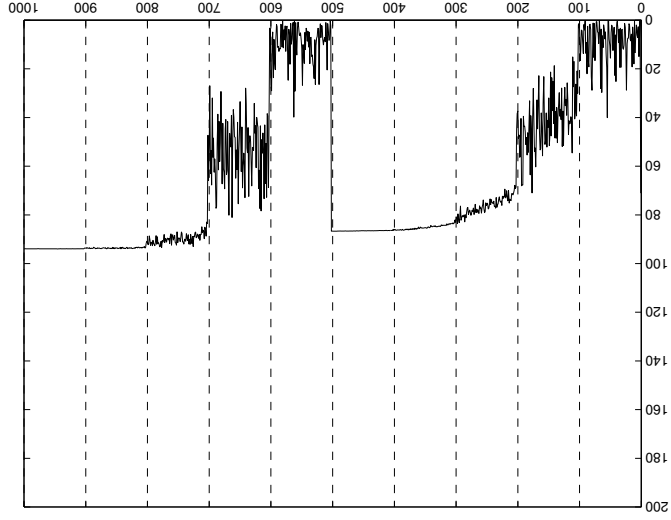
☞ a) exploration : soit  $\hat{\theta}^{k*}$  le meilleur point [+ petite valeur de  $j(\cdot)$ ] pour chaque  $j$ , on fait 100/ $j$  itérations avec  $\Sigma_j$ , et on note  $\hat{\theta}^{k*,j}$  la meilleure valeur obtenue,  $j = 1, \dots, 5$

☞ b) exploitation : on part du meilleur point  $\hat{\theta}^{k*,j^*}$ , obtenu pour  $j = j^*$ , et on fait 100 itérations avec  $\Sigma_{j^*}$

☞ exploration de nouveau, puis exploitation, etc.

☞ on arrête ... quand on en a assez (!) (nb. max. d'itérations dépassé), ou si  $j^* = 5$  ( $\leftrightarrow \Sigma$  petit  $\leftrightarrow$  recherche locale) un nombre consécutif de fois  $\geq 5$

Pas d'alternance  
 exploration/exploitation :  
 $\Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \Sigma_5, \Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4, \dots$



Alternance  
 exploration/exploitation puis  
 exploration/exploitation...

