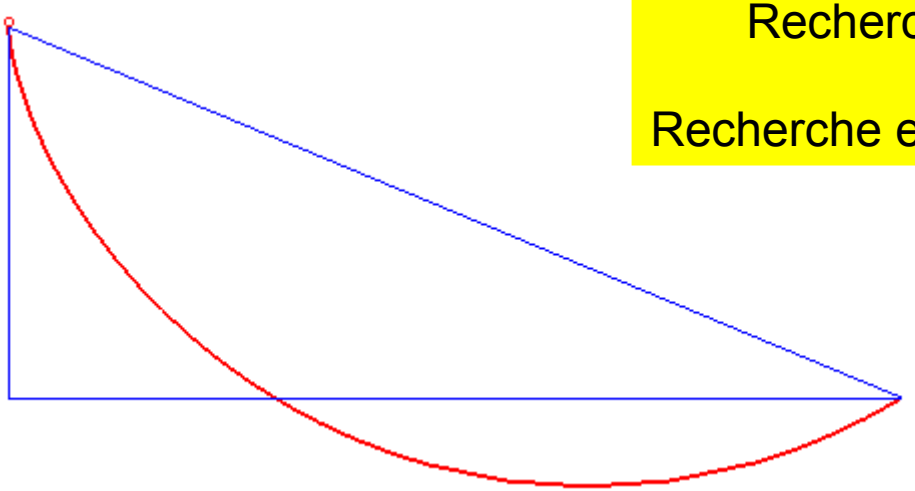
Optimisation continue

Y. Collette

Introduction Historique

Recherche opérationnelle
=
Recherche en opérations militaires



- 1696: Pose du problème du Brachistochrone
- 1696 + 1 jours: Résolution du problème du Brachistochrone par I. Newton (Invention du calcul différentiel par Leibniz)

En fait, résolut simultannément par Leibniz, Newton l'Hôpital et Bernouilli (à qui on attribue la paternité de la résolution)

Introduction
Historique

- 1948: Méthode du SIMPLEX pour la programmation linéaire
- 1950: Plusieurs méthode de recherche aléatoires. Méthode à base de gradient: fin 1950
- 1960: Technique d'optimisation séquentielle non-contrainte, Programmation séquentielle linéaire, Méthode des directions réalisables
- 1970: Méthodes des multiplicateurs (SUMT)
- 1980: Méthodes à métrique variable, Méthodes de Programmation Séquentielle Quadratiaque (SQP)

Introduction
Historique

- 1970: Algorithmes Génétiques
- 1980: Recuit simulé
- 2000: Optimisation par essaim particulaire
- 2000: Optimisation par estimation de distribution
- 2000: CMAES

- Plus Gros problème Test Connu
  - 250,000 Variables Avec 250,000 Contraintes Actives
- Plus Gros problème d'Optimisation de structure connu
  - 190,000 Variables d'épaisseur avec contraintes de fréquence
  - 2,000,000 Variables Topologiques

Plan

- Optimisation sans constraintes
- Optimisation avec constraintes
- Optimisation par Programmation séquentielle
- Programmation linéaire
- Transformation de fonction objectif

Optimisation Classique

Optimisation sans contraintes

Introduction

Un problème d'optimisation

- Minimiser $f(x)$

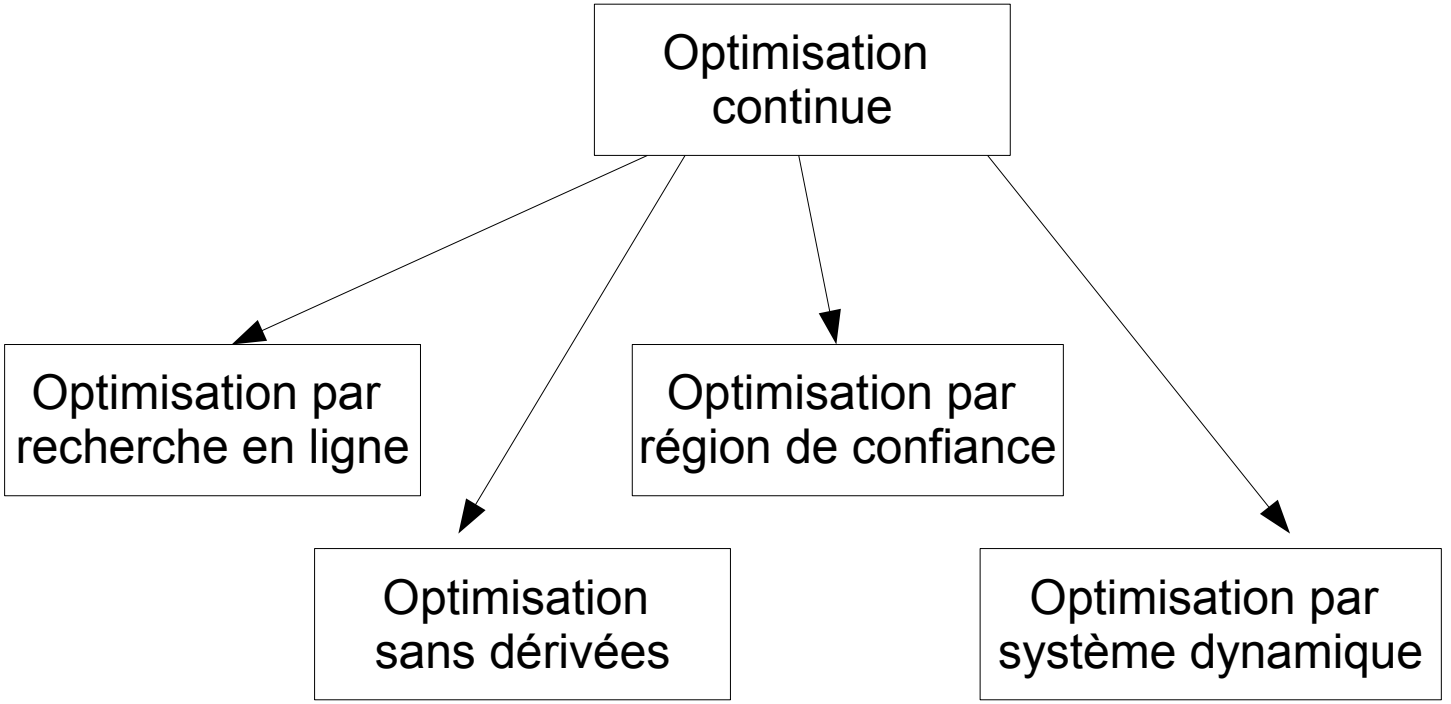
- Sachant que $g_i(x) \leq b_i, i = 1, \ldots, M$

$$h_j(x) = c_j, j = 1, \ldots, L$$

$$X_k^L \leq x_k \leq X_k^U, k = 1, \ldots, N$$

- Avec $f(x), g_i(x), h_j(x)$ des fonctions pouvant être linéaires, non linéaires ou implicites, ayant une dérivée première définie ou non;
- x une variable continue ou non

Différentes approches



Le Processus d'Optimisation

(optimisation par recherche en ligne)

1. Choisir un point initial
2. Trouver une direction de recherche
3. Effectuer une recherche en ligne
4. Vérifier la convergence
5. Si convergence: arrêt sinon, on repart à l'étape 2

Notes:

- Le même algorithme codé par différentes personnes se comportera de façon différente
- Partir d'un point initial différent produira une solution différente (bien que la valeur de la fonction objectif soit à peu près la même) – Méthode Multistart (Méthode du « Couillon » selon J. Gondrand)

Introduction

La stratégie générale

- Etant donné x_0 et $x = x_0$
- A l'itération n, mettre à jour x de la façon suivante : $x_n = x_{n-1} + \alpha . S_n$

- S_n = vecteur de la direction de recherche
- α = taille du pas $\alpha_n = \min_{\alpha} f(x_n + \alpha \cdot S_n)$
- Le calcul de S_n peut nécessiter les gradients
- Le calcul de α (recherche en ligne) va nécessiter le calcul de plusieurs fonctions objectifs

La Recherche en ligne

- A l'itération n, mise à jour de x:
- α est l'inconnue, on cherche donc α minimisant:

$$f\left(x_{n-1}+\alpha.S_n\right)$$

- On prend plusieurs valeurs de α et on évalue $f\left(x_{n-1}+\alpha.S_n\right)$ pour chaque α

Ou on optimise f en fonction de α

  - On estime le α qui minimise $f(x)$
- Plusieurs approches possibles:
  - Interpolation polynomiale (2nd ordre, 3e ordre)
  - Méthode du nombre d'or
  - Dichotomie
  - Sécante
  - Backtracking
  - etc.

La Recherche en ligne

Interpolation polynômiale (1/4)

- Soit

$$f(\alpha)=a_0+a_1.\alpha+a_2.\alpha^2+a_3.\alpha^3$$

Utilisé par BigDot[tm] dans Genesis[tm] http://www.vrand.com

- On a

$$\frac{df}{d\alpha}=a_1+2.a_2.\alpha+3.a_3.\alpha^2$$

$$\frac{d^2f}{d\alpha^2}=2.a_2+6.a_3.\alpha$$

- Donc

$$\frac{df}{d\alpha}=0\rightarrow\alpha^{opt}=\frac{-2.a_2\pm\sqrt{4.a_2^2-12.a_1.a3}}{6.a_3}$$

La Recherche en ligne

Interpolation polynômiale (2/4)

- Exemple:

$$\nabla f = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \quad S = \begin{bmatrix} -1 \\ -2 \end{bmatrix}$$

On recherche l'optimum suivant α.d

$$\frac{df}{d\alpha} = \frac{\partial f}{\partial x_1} \cdot \frac{\partial x_1}{\partial \alpha} + \frac{\partial f}{\partial x_2} \cdot \frac{\partial x_2}{\partial \alpha} = \nabla^t S = -5$$

- Calcul de f pour $\alpha = 0,1 \quad et \quad 2$

α	f	df/dα
0	10	-5
1	6	*
2	8	*

La Recherche en ligne

Interpolation polynômiale (3/4)

- Création d'un système de 4 équations et 4 inconnues:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 8 \end{bmatrix} \cdot \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 10 \\ -5 \\ 6 \\ 8 \end{bmatrix}$$

- La résolution donne

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 10 \\ -5 \\ 0 \\ 1 \end{bmatrix}$$

La Recherche en ligne

Interpolation polynômiale (4/4)

- Recherche du α optimal

$$f(\alpha) = 10 - 5.\alpha^2 + \alpha^2 \rightarrow \alpha^{opt} = \sqrt{(\frac{5}{3})} \simeq 1.291$$

$$\frac{d^2 f}{d\alpha^2} = 7.746 > 0$$

C'est donc un minimum ($d^2f/d\alpha^2>0$)

- Notes:
  - α doit être positif (sinon la direction ne pointe plus vers le minimum)
  - Se limiter à l'interpolation cubique
  - D'abord trouver une borne sur α puis interpoler

La Recherche en ligne

Le Nombre d'Or (1/3)

Méthode de recherche du α basée sur le nombre d'or:

- Trouver des bornes pour α
- Calculer deux points intérieurs de façon à réduire les bornes le plus rapidement possible

$$\alpha_1 = \alpha_U - \tau.(\alpha_U - \alpha_B) \quad \alpha_2 = \alpha_L + \tau.\alpha_U - \alpha_B$$

$$\tau = \frac{3-\sqrt{(5)}}{2} \simeq 0.38197$$

U: upper
L: lower
B: best

α1,α2: deux valeurs de α qui vont être calculées

La Recherche en ligne

Le Nombre d'Or (2/3)

$$F_1=\frac{3-\sqrt{5}}{2}, F_2=\frac{\sqrt{5}-1}{2}=1-F_1$$

$$F_1=F_2^2, F_1+F_2=1$$

Définition et propriétés du nombre d'or

$$\Delta^k=y_1^k-y_3^k=x_3^k-F_1\Delta^k$$

$$y_1^k=x_1^k+F_1\Delta^k$$

$$y_2^k=x_1^k+F_2\Delta^k$$

si $f(y_1^k)<f(y_2^k)$ alors $\Delta^{k+1}=(y_2^k-x_1^k)$ et $x_1^{k+1}=x_1^k, x_3^{k+1}=y_2^k$

si $f(y_1^k)>f(y_2^k)$ alors $\Delta^{k+1}=(x_3^k-y_1^k)$ et $x_1^{k+1}=y_1^k, x_3^{k+1}=x_3^k$

si $f(y_1^k)=f(y_2^k)$ alors $\Delta^{k+1}=(y_2^k-x_1^k)=(x_3^k-y_1^k)$ et

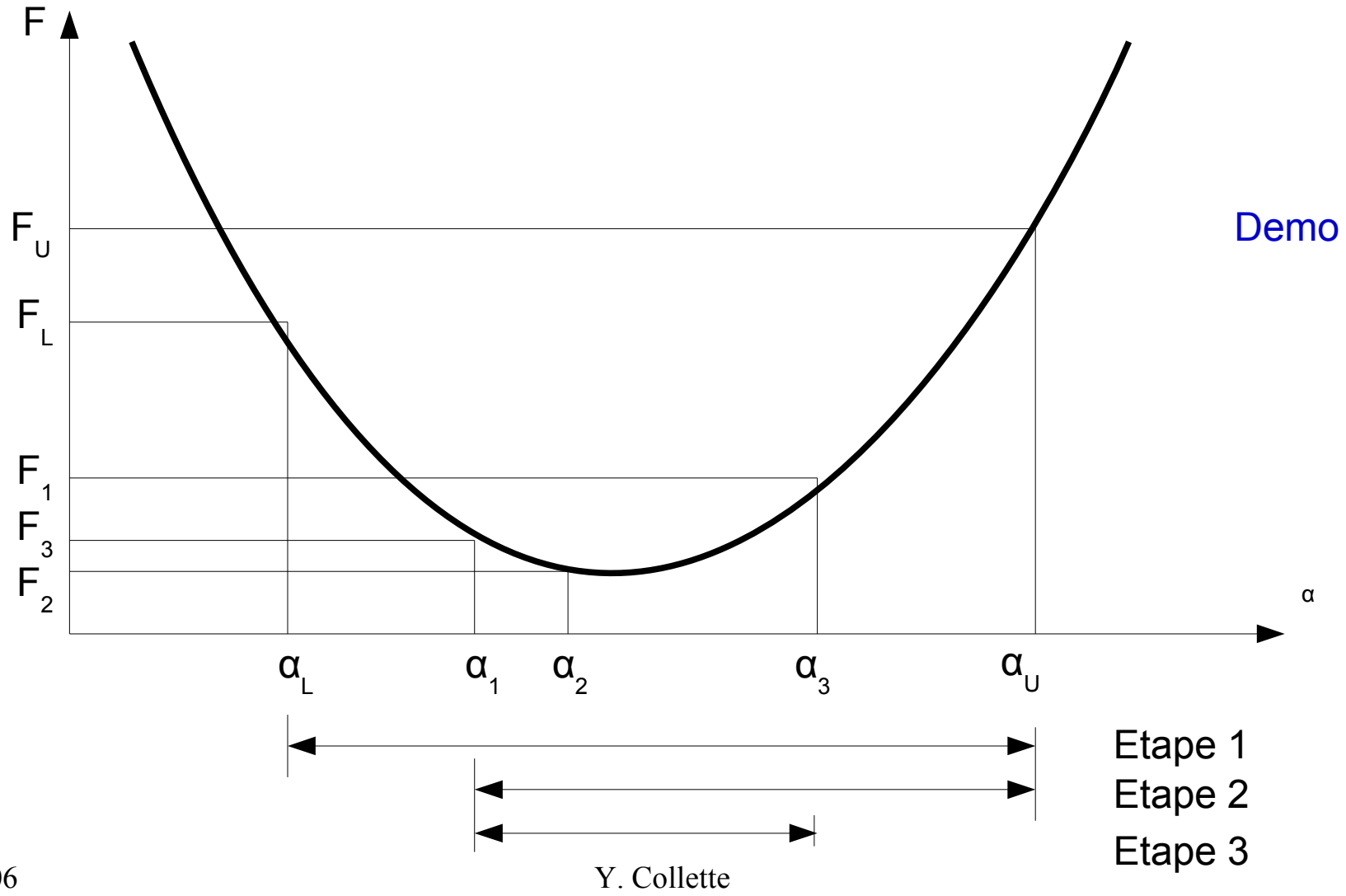
$$x_1^{k+1}=x_1^k, x_3^{k+1}=y_2^k \text{ ou}$$

$$x_1^{k+1}=y_1^k, x_3^{k+1}=x_3^k$$

Algorithme de la méthode du nombre d'or

La Recherche en ligne

Le Nombre d'Or (3/3)



Demo

Les Méthodes d'optimisation

Algorithmes Typiques

Ordre de la dérivée utilisée

Méthode de Rosenbrock	Ordre 0
Méthode de Nelder et Mead	Ordre 0
Steepest Descent	Ordre 1
Fletcher-Reeves	Ordre 1
Davidon-Fletcher-Powell	Ordre 1
Broydon-Fletcher-Goldfarb-Shanno	Ordre 1
Méthode de Newton	Ordre 2

Exemple

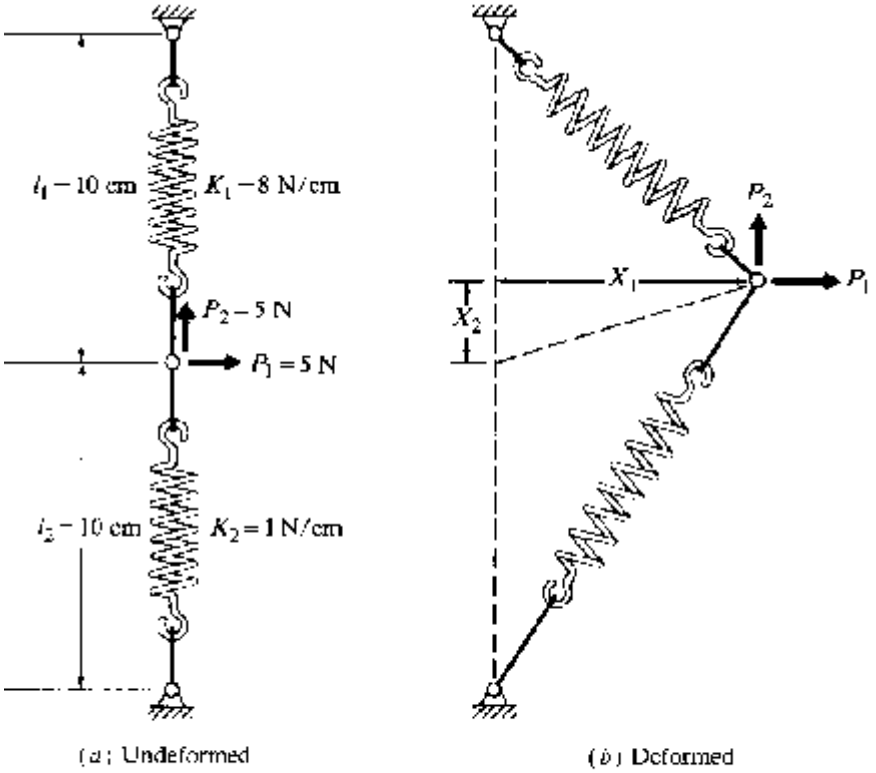
Définition du problème

Minimiser l'énergie potentielle totale PE

$$PE = \frac{1}{2}.K_1(\Delta L_1)^2 + \frac{1}{2}.K_2(\Delta L_2)^2 - P_1 X_1 - P_2 X_2$$

$$\Delta L_1 = \left[(10 - X_2)^2 + X_1^2\right]^{\frac{1}{2}} - 10$$

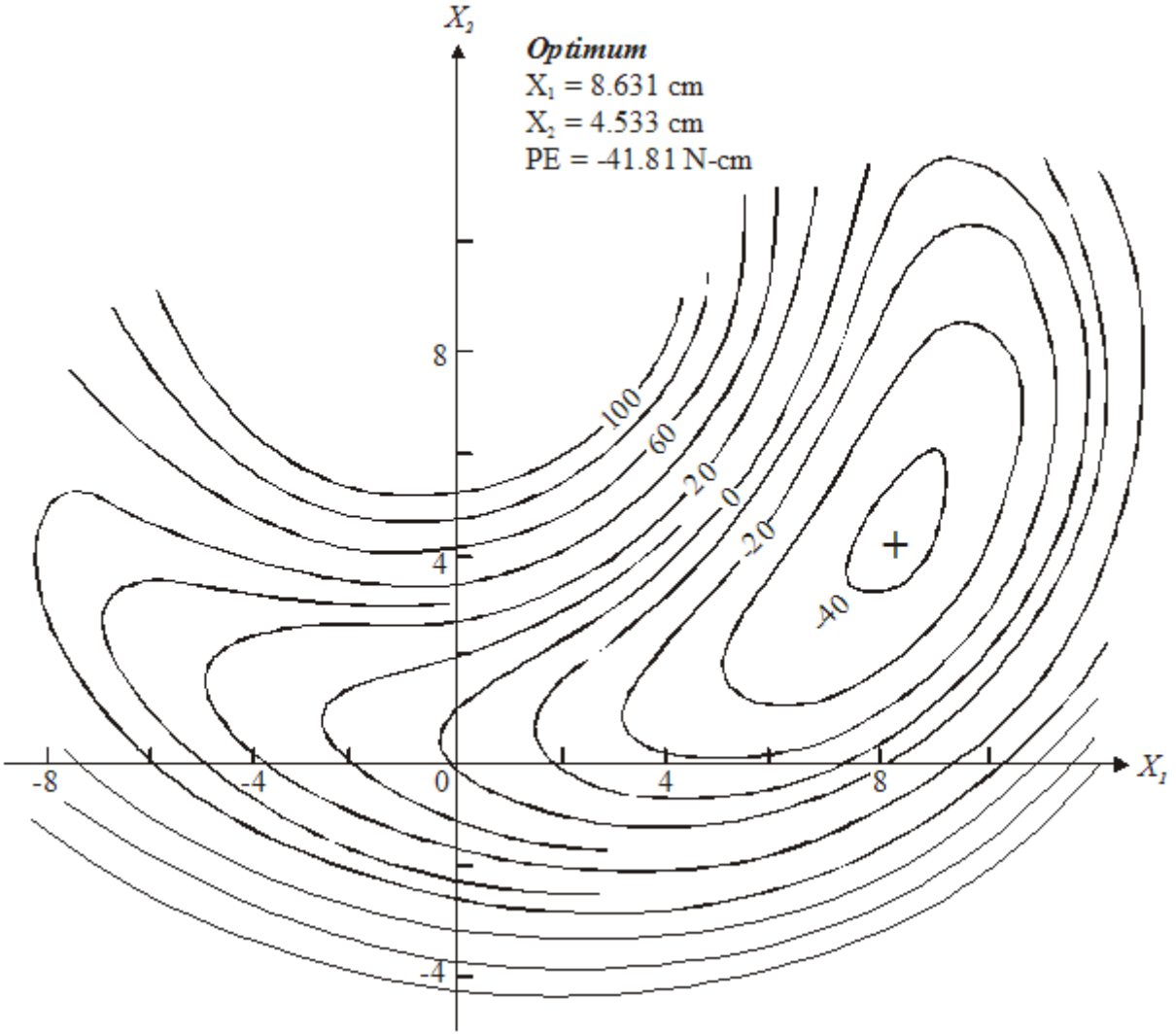
$$\Delta L_2 = \left[(10 + X_2)^2 + X_1^2\right]^{\frac{1}{2}} - 10$$



Exemple illustratif extrait d'une présentation de G. Vanderplaats

Exemple

Espace des paramètres



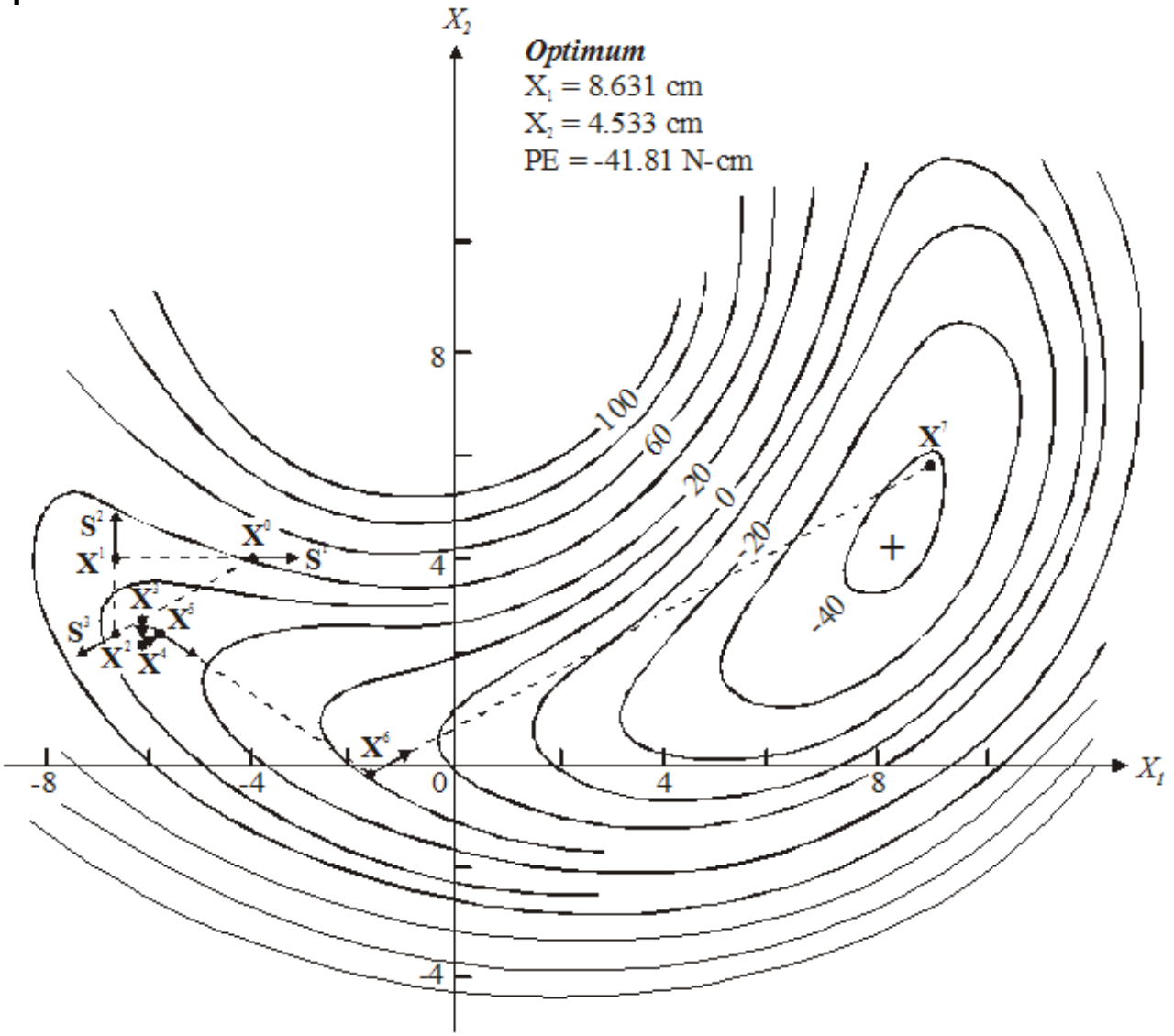
La Méthode de Powell

1. Chercher suivant chacune des directions de recherche.
   - α peut être positif ou négatif
   - Mémoriser chaque $\alpha_i^{opt} . S_i , \quad i = 1, \ldots, n$
2. Chercher suivant la direction conjugée $S_{n+1} = \sum_{i=1}^{n} \alpha_i^{opt} \cdot S_i$
3. Mettre tous les $\alpha_i^{opt} \cdot S_i$ en colonnes dans la matrice H, i = 1,n+1
4. Supprimer la colonne 1 et décaler les autres colonnes sur la gauche
5. Chercher suivant la direction donnée par les colonnes 1,n
6. Répéter les étapes à partir de 2 jusqu'à convergence

La matrice H contient la base canonique au départ

La Méthode de Powell

Graphiquement



La Méthode de Powell

Conclusion

- Utiliser uniquement les valeurs des fonctions objectif
- Efficace pour des problèmes de petite taille
- Résoud les problèmes quadratiques en n directions conjuguées ou moins

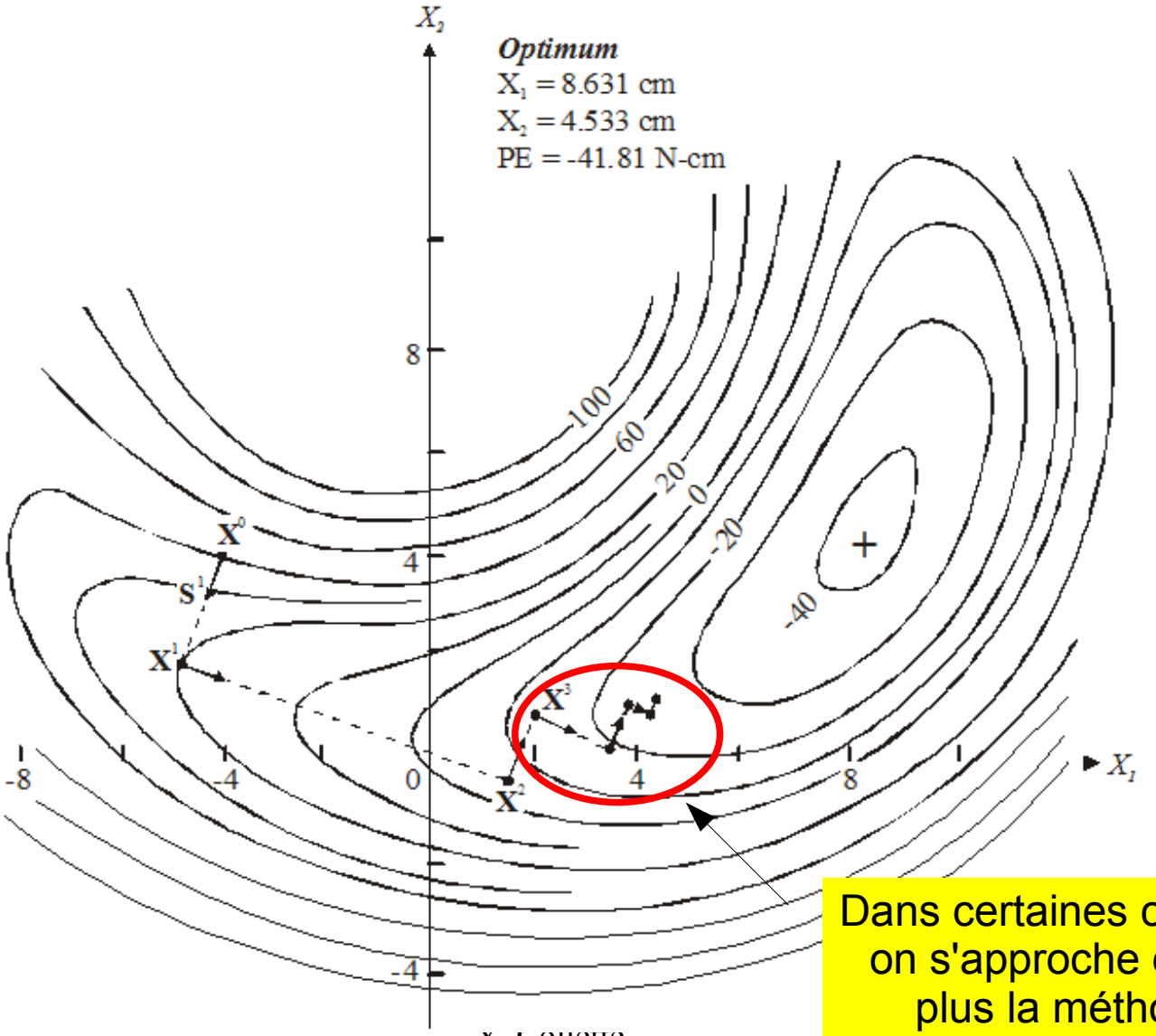
La méthode du Gradient

(Méthode « Steepest Descent »)

1. Calculer le gradient ∇f
2. Poser $S = -\nabla f \rightarrow Steepest\ Descent$
3. Chercher suivant la direction S
4. Répéter les étapes à partir de l'étape 1 jusqu'à convergence

La méthode du Gradient

Graphiquement



Dans certaines conditions, plus on s'approche de l'optimum, plus la méthode ralentit

La méthode du Gradient

conclusions

- Utilise les valeurs des fonctions objectif et les valeurs de gradient obtenues:
- Facile à programmer
- Consomme peu de mémoire
- Inefficace au possible

- analytiquement (différenciation autmatique)
- différences finies (forward, centrale)

Méthode à ne surtout pas utiliser

Demo

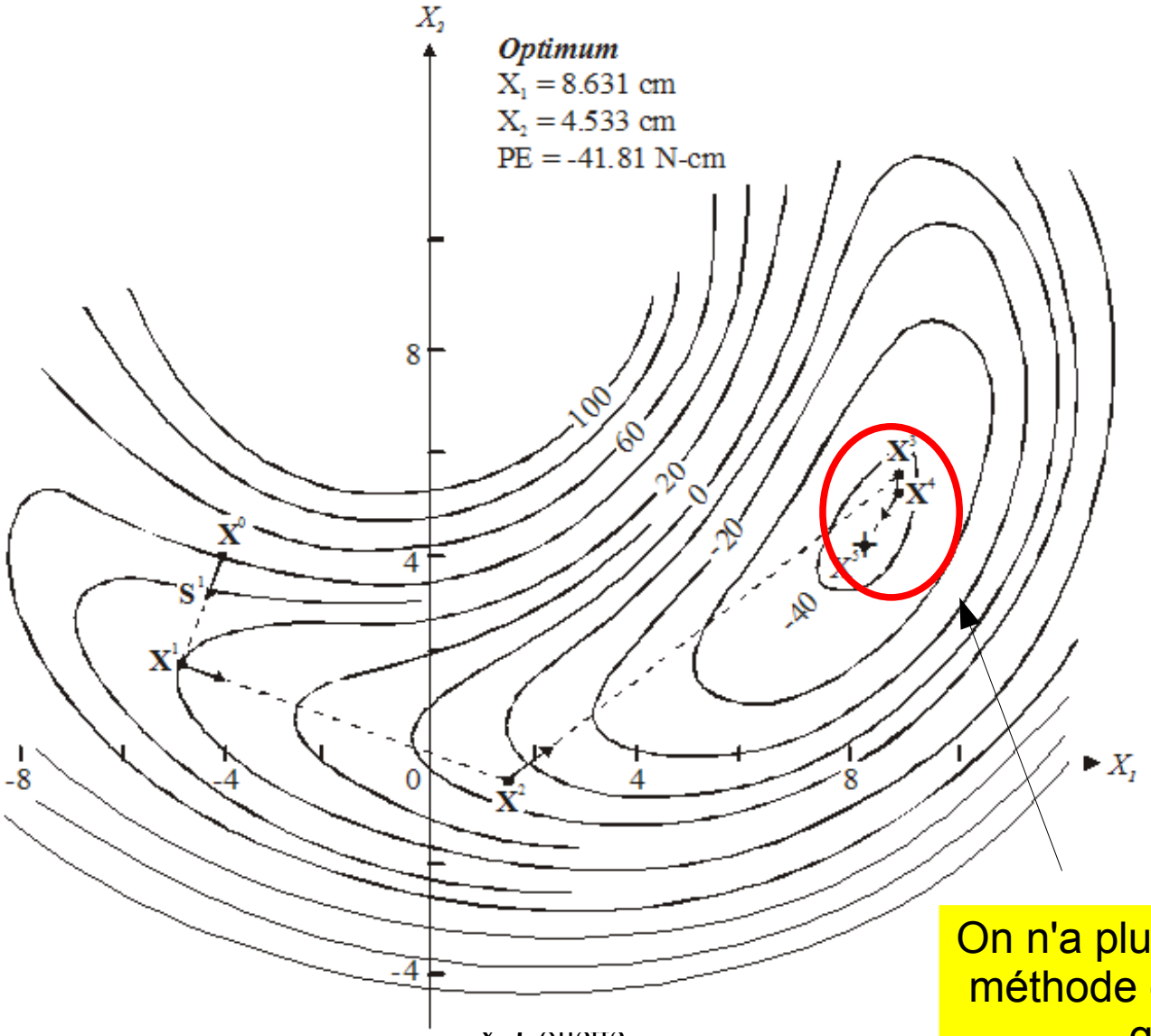
Le gradient conjugué

Variante de Fletcher-Reeves

1. Initialisation du compteur d'itérations n=0
2. On pose n=n+1 et on calcule le gradient ∇f
3. Si n=1 alors $S_n = -\nabla f(X_{n-1})$
4. Sinon $\beta = \dfrac{\left|\nabla f(X_n)\right|^2}{\left|\nabla f(X_{n-1})\right|^2}$ et $S_q = -\nabla f(X_{n-1}) + \beta \cdot S_{n-1}$
5. Chercher dans la direction S_n
6. Répéter à partir de l'étape 2 jusqu'à convergence

La Méthode de Fletcher-Reeves

Graphiquement



On n'a plus le défaut de la méthode de descente du gradient

La Méthode de Fletcher-Reeves

Conclusions

- Utilise les valeurs de fonction objectif et du gradient
- Facile à programmer

Demo

- Peu gourmand en mémoire
- Converge en **n** itérations ou moins pour les problèmes quadratiques
- Réinitialiser avec la méthode Steepest Descent quand les progrès ralentissent (après **n+1** itérations)

La Méthode du gradient conjugué

Divers

Comment on obtient la direction de recherche de Fletcher-Reeves

Direction conjuguée:
Trouver S^{k+1} tel que S^{k+1} H S^k
On modélise cette nouvelle direction comme une somme entre la direction donnée par la direction inverse du gradient et la direction suivie lors de l'itération précédente.

$$s^1 = -\nabla f(x^1) + \omega s^0$$
$$s^{0t} H s^1 = 0$$

Un calcul donne :

$$\omega = \frac{\nabla^t f(x^1) \nabla f(x^1)}{\nabla^t f(x^0) \nabla f(x^0)}$$

☆ ☆ Hestenes-Stiefel:

$$\omega = \frac{\nabla^t f(x^1)(\nabla f(x^1) - \nabla f(x^0))}{s^0(\nabla^t f(x^1) - \nabla f(x^0))}$$

☆ Polak-Ribière:

$$\omega = \frac{\nabla^t f(x^1)(\nabla f(x^1) - \nabla f(x^0))}{\nabla^t f(x^0) \nabla f(x^0)}$$

La Méthode de Newton

Introduction

- Considérons un développement Taylor à l'ordre 2

$$\tilde{f} = f + \nabla f^t \cdot \delta X + \frac{1}{2} \delta X^t \cdot H \cdot \delta X$$

$$\delta X = X - X_0$$

- On obtient un minimum en $\nabla \tilde{f} = 0$ (H étant définie-positive)

$$\nabla \tilde{f} = \nabla f_0 + H \cdot \delta X = 0 \text{ d'où } X = X_0 - H^{-1} \cdot \nabla f_0$$

- Comme le problème général n'est pas totalement quadratique, on utilise δX comme direction de recherche S puis on applique une recherche unidimensionnelle

Plein de difficultés apparaissent à cet endroit

La Méthode de Newton

Exemples et algorithme

Demo 1D
Demo 2D

1. Calculer $\nabla f(X_{n-1})$ et $H(X_{n-1})$
2. Poser $S_n = -H(X_{n-1})^{-1} \cdot \nabla f(X_{n-1})$
3. Chercher dans la direction S_n, $\alpha = 1$ étant un bon départ
4. Répéter les étapes à partir de l'étape 1 jusqu'à convergence

Pas utilisé comme tel car
Hessien difficile à obtenir

Les Méthodes à Métrique Variable

Introduction

- Direction de recherche $S_n = -H \cdot \nabla f(X_{n-1})$
- Après la recherche unidimensionnelle, mettre à jour l'estimation de H:

$$H_{n+1} = H_n + D_n$$

$$D_n = \frac{\sigma + \theta \cdot \tau}{\sigma^2} p \cdot p^t + \frac{\theta - 1}{\tau} (H_n \cdot y)(H_n \cdot y)^t - \frac{\theta}{\sigma} \left[(H_n \cdot y \cdot p^t) + p(H_n \cdot y)^t \right]$$

- Où

$$p = X_n - X_{n-1}, \quad y = \nabla f(X_n) - \nabla f(X_{n-1}), \quad \sigma = p^t y, \quad \tau = y^t \cdot H_n \cdot y$$

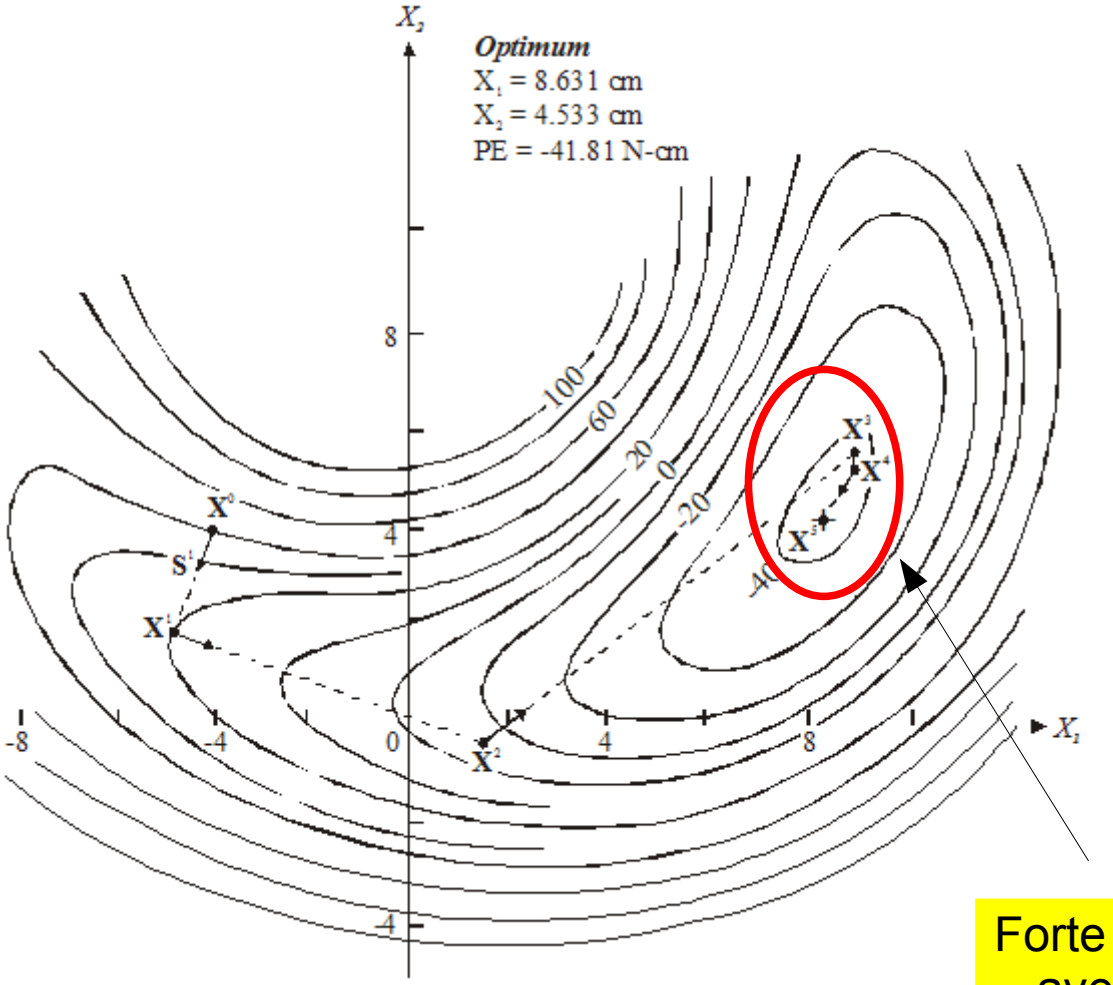
$\theta = 0$ pour la méthode DFP

$\theta = 1$ pour la méthode BFGS

Autre méthode: SR1

Les Méthodes à Metrique Variable

Graphiquement



Forte ressemblence avec le gradient conjugué

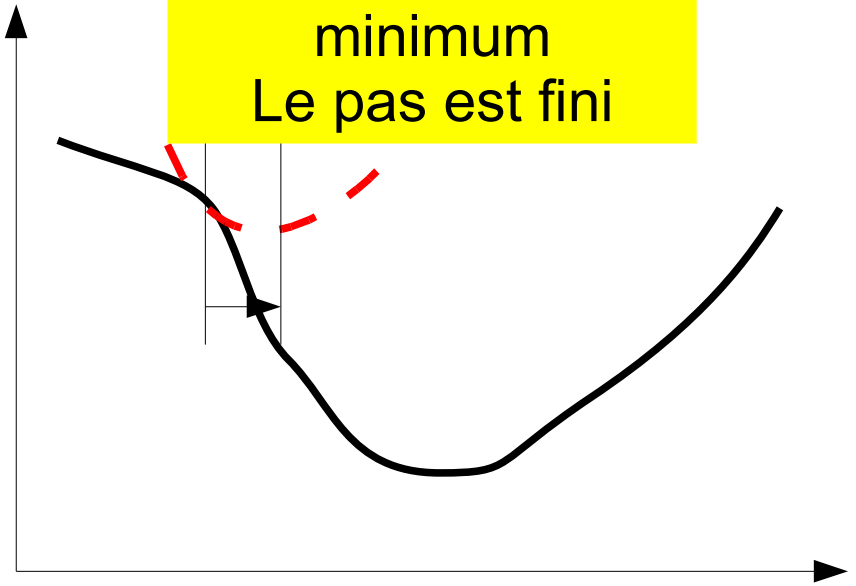
Les méthodes à Métrique Variable

Interêt matrice définie positive

$$y = a + b\,x + c\,x^2$$

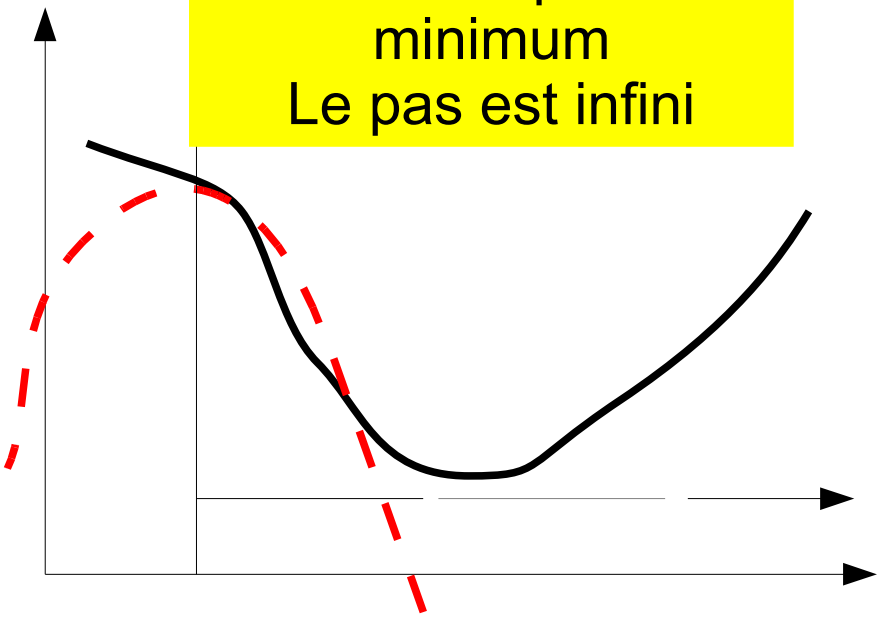
$c > 0$

Il existe un minimum
Le pas est fini



$c < 0$

Il n'existe pas de minimum
Le pas est infini



Les Méthodes à Métrique Variable

Conclusions

- Utilise les valeurs de fonction objectif et de gradient
- Plus difficile à programmer Demo
- Grosse consommation de mémoire (matrice H – pas tout le temps creuse)
- Converge en **n** itérations ou moins pour les problèmes quadratiques
- Réinitialisation avec la méthode Steepest Descent lorsque la performance ralentie (après **n+1** itérations)
- Pour la consommation mémoire, il y a L-BFGS

Comparaison de Méthodes

1. Méthode de Powell
2. Méthode Steepest Descent
3. Méthode Fletcher-Reeves
4. Méthode DFP
5. Méthode BFGS
6. Méthode de Newton

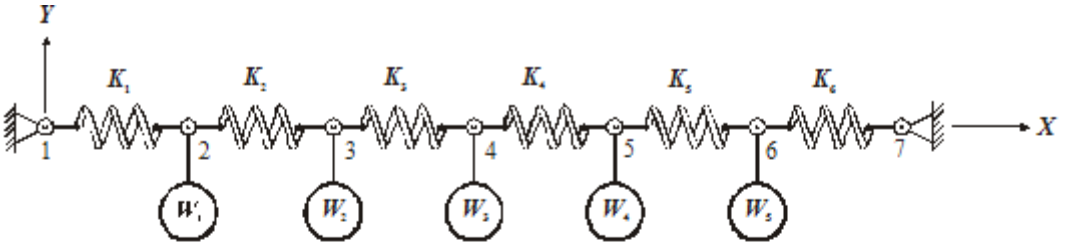
La recherche en ligne est faite par interpolation polynômiale

Comparaison de méthodes

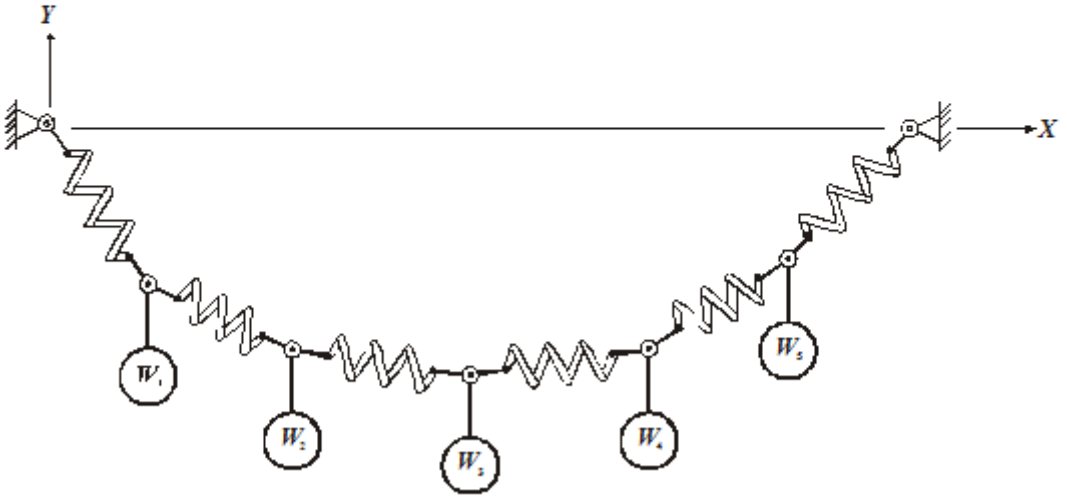
Le problème test

Minimisation de l'énergie potentielle totale :

$$PE = \sum_{i=1}^{N+1} \frac{1}{2} K_i \cdot \Delta L_i + \sum_{j=1}^{N} W_j \cdot Y_{j+1}$$



(*a*) Undeformed position



(*b*) Deformed position

Exemple illustratif extrait d'une présentation de G. Vanderplaats

Les Méthodes d'optimisation

Algorithmes Typiques

Nom de la méthode	Ordre de la méthode	No
Méthode de Powell	Ordre 0	1
Méthode du gradient	Ordre 1	2
Fletcher-Reeves	Ordre 1	3
Davidon-Fletcher-Powell	Ordre 1	4
Broydon-Fletcher-Goldfarb-Shanno	Ordre 1	5
Méthode de Newton	Ordre 2	6

Comparaison de méthodes

Résultats

Variable de Décision	Valeur Initiale	Méthode					
		1	2	3	4	5	6
X_2	10	10.3	10.3	10.4	10.2	10.2	10.4
X_3	20	21.1	20.7	21.1	20.8	20.8	21.1
X_4	30	31.7	31.0	31.6	31.4	31.4	31.7
X_5	40	42.1	41.3	42.0	41.8	41.7	42.1
X_6	50	51.8	51.1	51.6	51.4	51.4	51.8
Y_2	0	-4.28	-2.65	-3.96	-4.64	-4.64	-4.28
Y_3	0	-7.90	-5.25	-7.77	-8.19	-8.19	-7.90
Y_4	0	-9.86	-7.35	-10.2	-10.0	-10.0	-9.86
Y_5	0	-9.40	-7.63	-9.52	-9.18	-9.19	-9.40
Y_6	0	-6.01	-4.97	-5.79	-5.42	-5.43	-6.01

1 – Powel
2 – Gradient
3 – FR
4 – DFP
5 – BFGS
6 – Newton

Comparaison de méthodes

Résultats

Itération	Méthode					
	1	2	3	4	5	6
0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	-60	-60	-60	-60	-1256
2	0.0	-126	-292	-292	-292	-1618
3	0.0	-151	-661	-666	-666	-1987
4	0.0	-194	-1148	-941	-941	-2175
5	0.0	-223	-1559	-1430	-1432	-2330
10	-1722	-1958	-3540	-2573	-2572	-3539
20	-2623	-2621	-4393	-4340	-4340	-3740
Finale	-4416	-3964	-4393	-4378	-4378	-4416
Itérations	178	40	22	26	26	19
Analyses	465	578	331	383	383	378

Nombre d'appels à la fonction objectif

1 – Powel
2 – Gradient
3 – FR
4 – DFP
5 – BFGS
6 – Newton

La méthode LFOP

Introduction

LFOP = Leap-Frog OPtimizer

On résoud l'équation du mouvement « modifiée » :

$$M \cdot \frac{\partial^2 x}{\partial t^2} = -\nabla f$$

Pour favoriser la convergence, on ajoute de l'amortissement :

$$M \cdot \frac{\partial^2 x}{\partial t^2} = -\nabla f - \alpha \cdot \frac{\partial x}{\partial t}$$

Paramètres de la méthode:
Δt: la taille du pas temporel pour la résolution du système dynamique
δ: la taille maximale du pas de recherche
m: si $a_{k+1}{}^t.a_k$ est non positif pendant m itérations, on réduit Δt et on repart d'un autre point
δ_1: taux d'accroissement du coefficient de dilatation de Δt
pstart: valeur de départ du coefficient de dilatation (1.01 est une bonne valeur)

La méthode LFOP

Algorithme

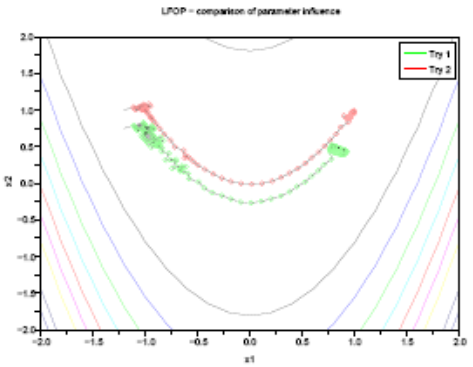
Algorithme 18 La méthode LFOP1(b).

1 : Paramètres utilisateur : $x_0, \Delta t, \delta, m, \delta_1$.
$i = 0, j = 2, k = -1, s = 0, p = p_{start}$
2 : Calculer $a_0 = -\nabla f(x_0)$ et $v_0 = 1/2 \cdot a_0 \cdot \Delta t$
3 : $k = k + 1$ et calculer $\|\Delta x_k\| = \|v_k\| \cdot \Delta t$
4 : si $\|\Delta x_k\| < \delta$ alors aller à l'étape 5a
sinon $v_k = \delta \cdot v_k / (\Delta t \cdot \|v_k\|)$ et aller à l'étape 5b
5a : $p = p + \delta_1, \Delta t = p \cdot \Delta t$
5b : si $s < m$ alors aller à l'étape 5
sinon $\Delta t = \Delta t/2, x_k = (x_k + x_{k-1})/2$
$v_k = (v_k + v_{k-1})/4, s =$
0 et aller à l'étape 5
5 : $x_{k+1} = x_k + v_k \cdot \Delta t$
6 : Calculer $a_{k+1} = -\nabla f(x_{k+1})$ et $v_{k+1} = v_k + a_{k+1} \cdot \Delta t$
7a : si $a_{k+1}^t \cdot a_k > 0$ alors $s = 0$ et aller à l'étape 7
sinon $s = s + 1, p = 1$ et aller à l'étap 7
7 : si $\|a_{k+1}\| \leq \epsilon$ alors STOP
sinon aller à l'étape 8
8 : si $\|v_{k+1}\| > \|v_k\|$ alors $i = 0$ et aller à l'étape 3
sinon $x_{k+2} = (x_{k+1} + x_k)/2$, $i =$
$i + 1$ et aller à l'étape 9
9 : si $i \leq j$ alors $v_{k+1} = (v_{k+1} + v_k)/4$ et $k =$
$k + 1$ et aller à l'étape 6
sinon $v_{k+1} = 0$, $j = 1$, $k = k + 1$ et aller à l'étape 6

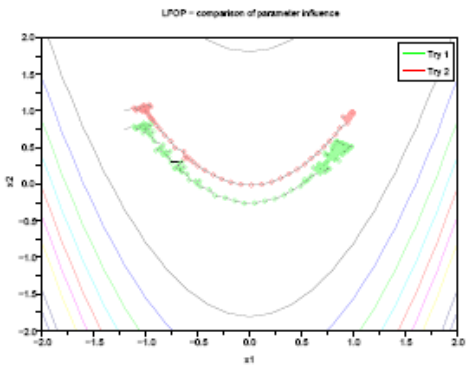
Utilisé dans l'outil LS-Dyna (via la toolbox LS-Opt)
De nombreuses applications industrielles:

- Crash;
- Optimisation de structure;
- ...

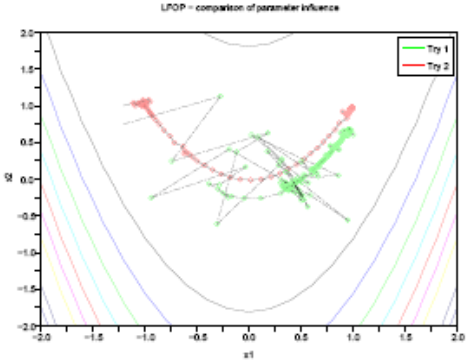
La méthode LFOP
Essais de paramétrage



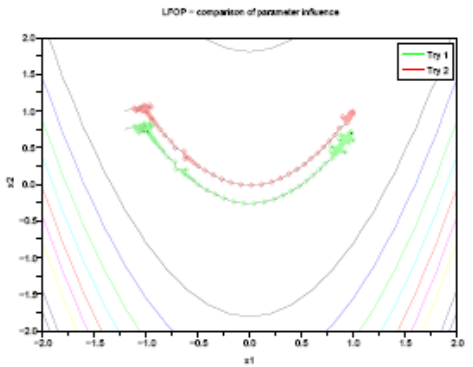
Influence du paramètre **m**



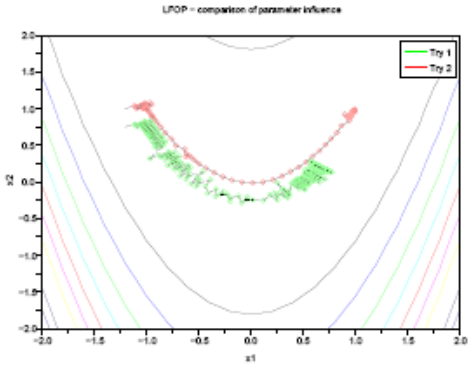
Influence du paramètre $\boldsymbol{\delta_1}$



Influence du paramètre **δ**



Influence du paramètre **Δt**

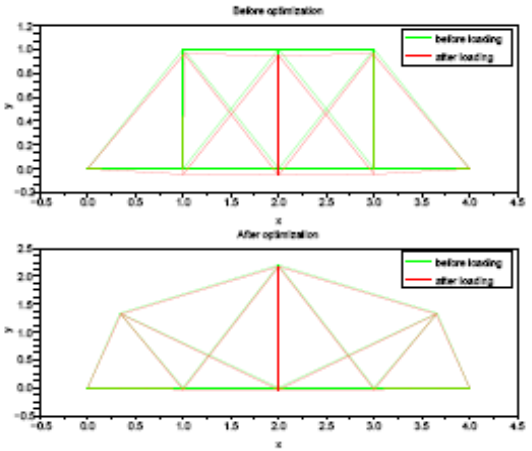


Influence du paramètre $\boldsymbol{p_{start}}$

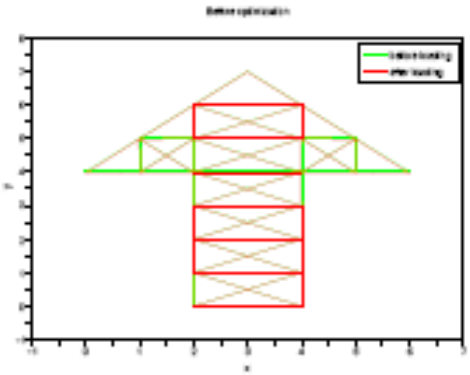
	Paramètres nominaux	Paramètres modifiés
m	3	10
δ_1	0.01	0.1
δ	0.1	1
Δt	0.5	5
p_{start}	1.1	2.1

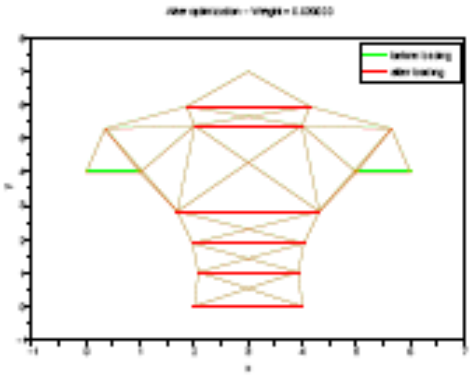
Quelques applications

Pont et gradient conjugué

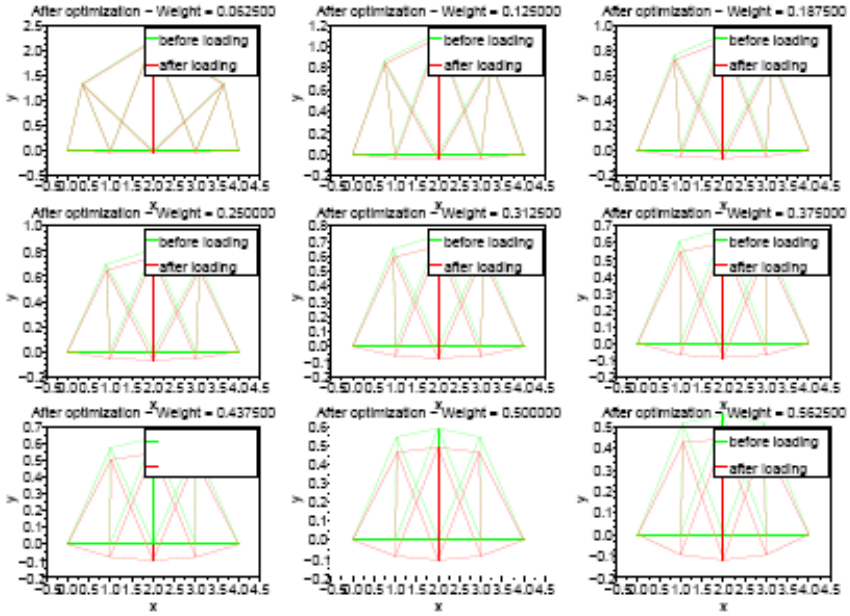


Minimiser la déformation





Pylône et la méthode LFOP



Minimiser la déformation **et** la longueur totale des barres

Convergence

Quelques critères d'arrêt

- Nombre maximum d'itérations
- Conditions KKT satisfaites de façon raisonable

$$\left|\nabla f(X)\right| \leq \epsilon_g$$

- Chute de performance
  - Atteinte depuis un ou plusieurs itérations
  - Absolue $\left|f(X_q) - f(X_{q-1})\right| \leq \epsilon_A$
  - Relative $\left|\frac{f(X_q) - f(X_{q-1})}{f(X_{q-1})}\right| \leq \epsilon_R$

La recherche en ligne imparfaite

Introduction

- Chercher à localiser avec précision la position d'un optimum dans une recherche en ligne coûte très cher en itérations !
- Cette précision n'est pas nécessaire au bon fonctionnement des méthodes d'optimisation
- Recherche en ligne imparfaite: « Backtracking »
- Repose sur la définition de la définition de la qualité suffisante d'un α:
  - Condition de Armijo
  - Condition de Goldstein
  - Condition de Wolfe
  - Condition de Wolfe Forte

La Recherche en ligne

Pas initial et Backtraking

Choix du pas initial

Pour les méthodes de Newton et quasi-Newton: $\alpha_0 = 1$

Pour les autres méthodes: $\alpha_0 \cdot \nabla f(x_k)^t \cdot p_k = \alpha_{k-1} \cdot \nabla f(x_{k-1})^t \cdot p_{k-1}$

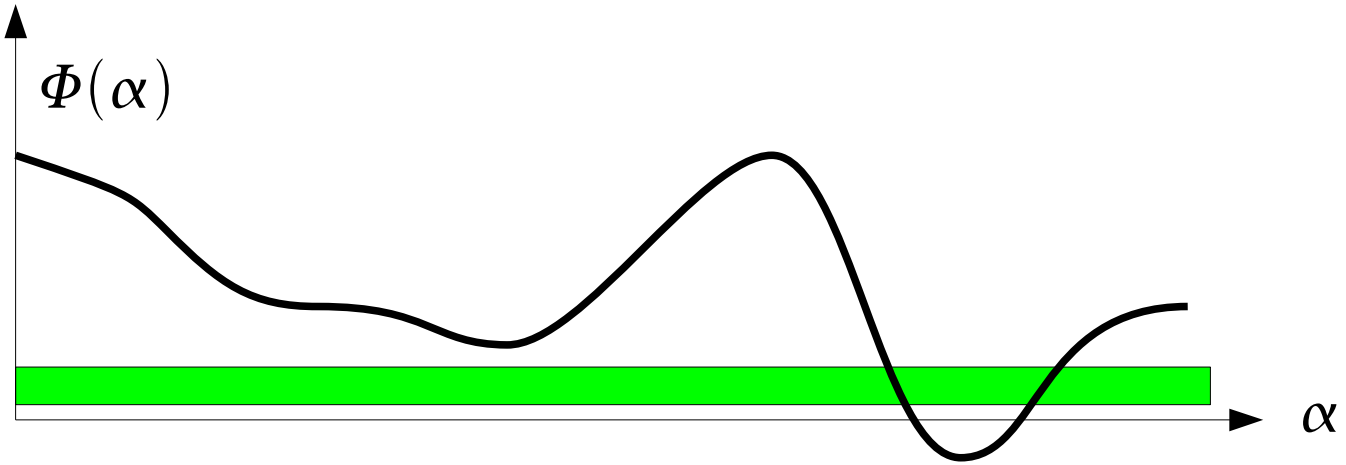
Backtracking → opt++ + limite sur le nombre max d'itérations

Choisir $\bar{\alpha} > 0, \rho, c_1 \in]0,1[$
$\alpha = \bar{\alpha}$
repeat
$\alpha = \rho \cdot \alpha$
until $f(x_k + \alpha \cdot p_k) \leq f(x_k) + c_1 \cdot \alpha \cdot \nabla f(x_k)^t \cdot p_k$

La Recherche en ligne

Condition de base

On pose $\Phi(\alpha) = f(x_k + \alpha \cdot p_k)\ \alpha > 0$



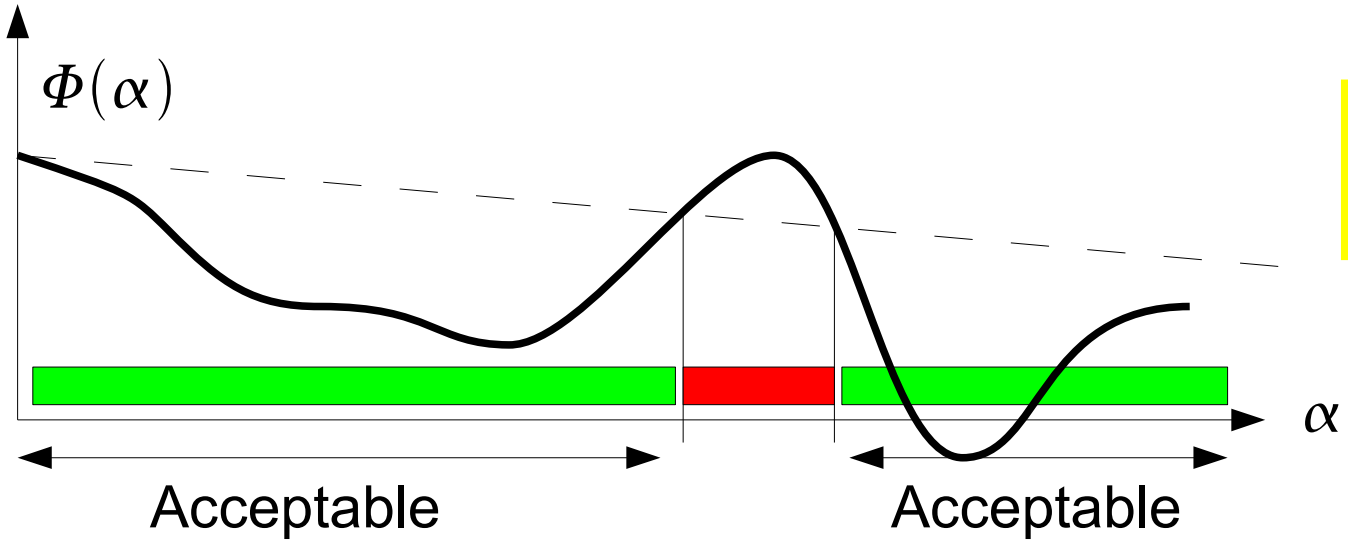
On impose $f(x_k + \alpha \cdot p_k) < f(x_k)$

→ Pas suffisant

La Recherche en ligne

Condition d'Armijo

On pose $\Phi(\alpha) = f(x_k + \alpha \cdot p_k)\ \alpha > 0$



Condition utilisé dans la méthode de Backtracking

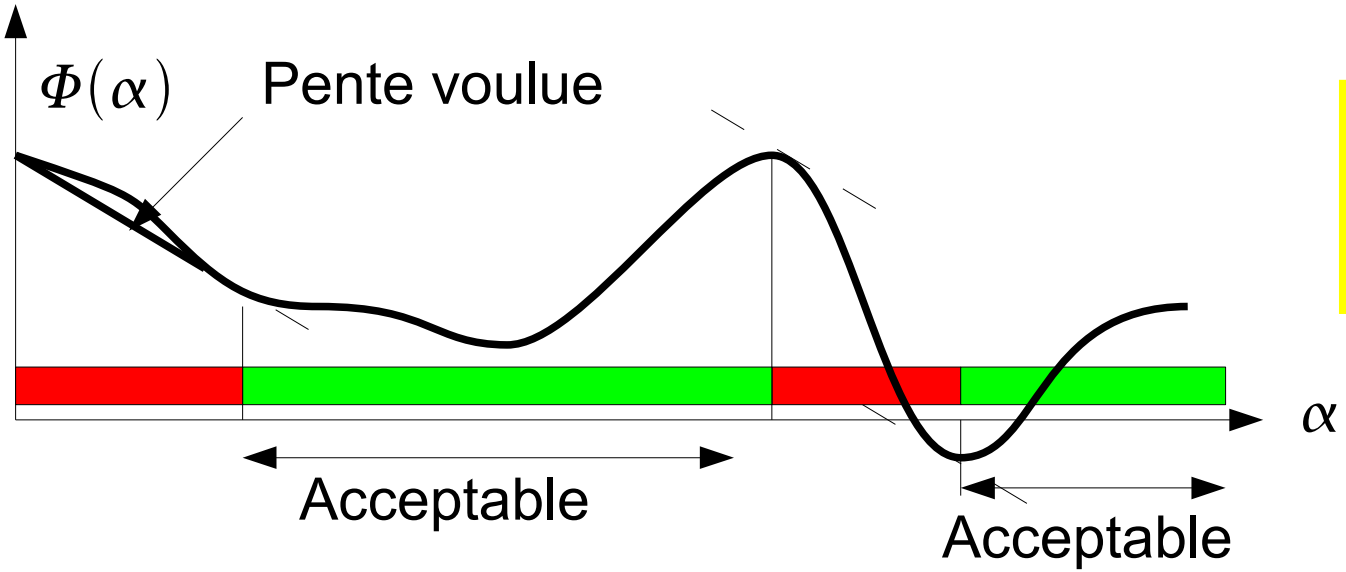
On impose $f(x_k + \alpha \cdot p_k) \leq f(x_k) + c_1 \cdot \alpha \cdot \nabla f_k^t(x_k) \cdot p_k = l(\alpha)$

Habituellement, $c_1 \simeq 10^{-4}$

La Recherche en ligne

Condition de pente

On pose $\Phi(\alpha) = f(x_k + \alpha \cdot p_k)\ \alpha > 0$



Condition utilisé dans d'autres méthodes de recherche imparfaite

Nouvelle condition: $\nabla f(x_k + \alpha \cdot p_k)^t \cdot p_k \geq c_2 \cdot \nabla f(x_k)^t \cdot p_k$

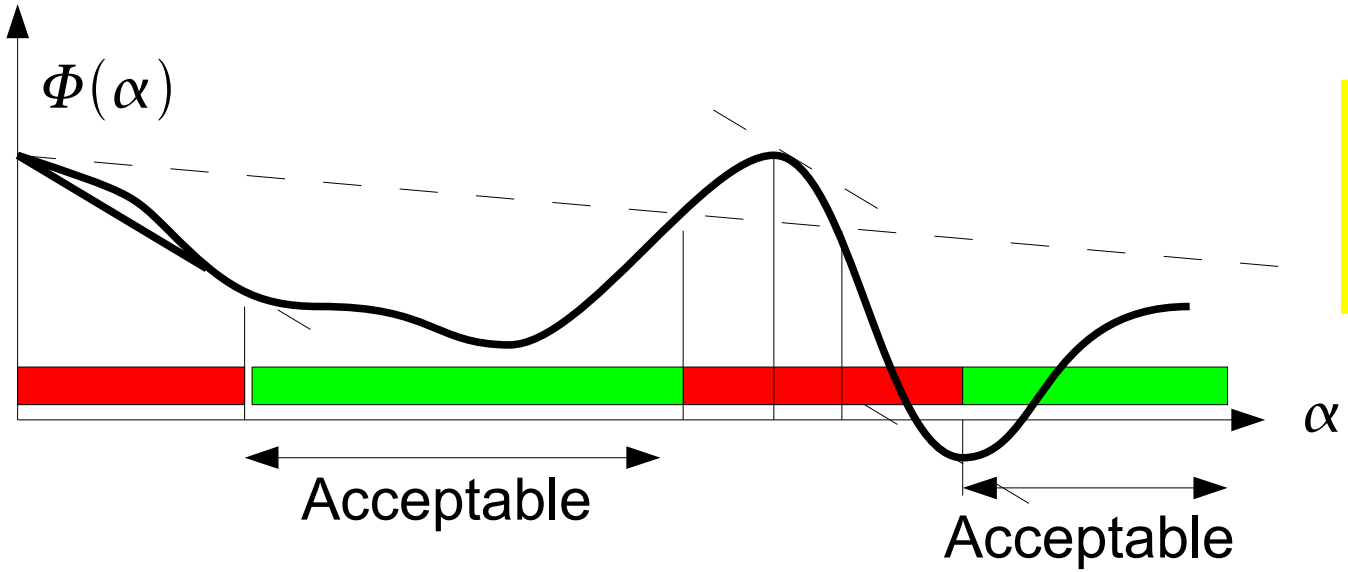
$c_2 \simeq 0.9$ si p_k est choisi par une méthode de newton ou quasi-newton

$c_2 \simeq 0.1$ si p_k est choisi par une méthode de gradient conjugué

La Recherche en ligne

Condition de Wolfe

On pose $\Phi(\alpha) = f(x_k + \alpha \cdot p_k) \quad \alpha > 0$



Condition utilisé dans d'autres méthodes de recherche imparfaite

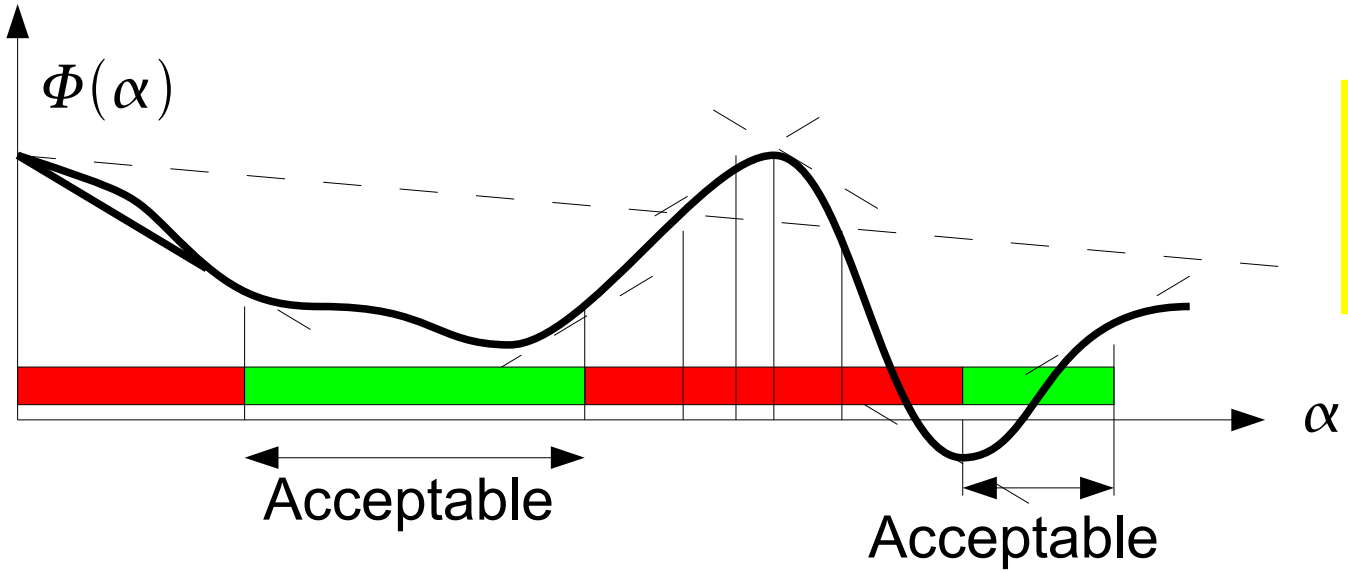
Conditions: $f(x_k + \alpha \cdot p_k) \leq f(x_k) + c_1 \cdot \alpha \cdot \nabla f_k(x_x)^t$

$\nabla f(x_k + \alpha \cdot p_k)^t \cdot p_k \geq c_2 \cdot \nabla f(x_k)^t \cdot p_k$

La Recherche en ligne

Condition forte de Wolfe

On pose $\Phi(\alpha) = f(x_k + \alpha \cdot p_k) \quad \alpha > 0$



Condition utilisé dans d'autres méthodes de recherche imparfaite

Conditions: $f(x_k + \alpha \cdot p_k) \leq f(x_k) + c_1 \cdot \alpha \cdot \nabla f_k(x_x)^t$

$$|\nabla f(x_k + \alpha \cdot p_k)^t \cdot p_k| \leq c_2 \cdot |\nabla f(x_k)^t \cdot p_k|$$

La Recherche en ligne

Strong Wolfe line search (1/2)

Algorithme Principal

Choisir $\alpha_1 > 0$ et $\alpha_{\max}$
$\alpha_0 = 1$
$i = 1$
repeat
  évaluer $\Phi(\alpha_i)$
  si $(\Phi(\alpha_i) > \Phi(0) + c_1 \alpha_i \Phi'(0))$ et
    $(\Phi(\alpha_i) \geq \Phi(\alpha_{i-1})$ et $i > 1)$
    $\alpha^* = \text{Zoom}(\alpha_{i-1}, \alpha_i)$ et Stop
  évaluer $\Phi'(\alpha_i)$
  si $(|\Phi'(\alpha_i)| \leq -c_2 \Phi'(0))$
    $\alpha^* = \alpha_i$ et Stop
  si $(\Phi'(\alpha_i) \geq 0)$
    $\alpha^* = \text{Zoom}(\alpha_i, \alpha_{i-1})$ et Stop
  choisir $\alpha_{i+1} \in]\alpha_i, \alpha_{\max}[$
  $i = i + 1$
end repeat

La séquence des α_i est monotoniquement croissante.
Cette procédure utilise l'information que l'interval (α_{i-1}, α_i) contient des pas qui vérifient la condition forte de Wolfe si l'une des conditions suivantes est vérifiée:

- α_i viole la condition de décroissance suffisante;
- $\Phi(\alpha_i) \geq \Phi(\alpha_{i-1})$;
- $\Phi'(\alpha_i) \geq 0$.

Condition de Wolfe non vérifiée

Condition de Wolfe Forte vérifiée

Pas une direction de descente

$\Phi(\alpha) = f(x_k + \alpha p_k)$

La Recherche en ligne

Strong Wolfe line search (2/2)

Fonction Zoom(α_{lo}, α_{hi})

repeat
 interpoler de façon à trouver un premier α_j
  entre α_{lo} et α_{hi}
 évaluer $\Phi(\alpha_j)$
 si $(\Phi(\alpha_j) > \Phi(0) + c_1 \alpha_j \Phi'(0))$ ou
  $(\Phi(\alpha_j) \geq \Phi(\alpha_{lo}))$
  $\alpha_{hi} = \alpha_j$
 sinon
  évaluer $\Phi'(\alpha_j)$
  si $(|\Phi'(\alpha_j)| \leq -c_2 \Phi'(0))$
   $\alpha^* = \alpha_j$ et Stop
  si $(\Phi'(\alpha_j)(\alpha_{hi} - \alpha_{lo}) \geq 0)$
   $\alpha_{hi} = \alpha_{lo}$
  $\alpha_{lo} = \alpha_j$
 end sinon
end repeat

Si la fonction Zoom trouve un α_j qui satisfait la condition forte de Wolfe, on retourne cette valeur.
Autrement, on cherche un α_j qui vérifient l'une des conditions suivante:
1. l'interval α_{lo} α_{hi} contient des pas qui vérifient la condition de Wolfe
2. α_{lo} est, parmis les pas générés jusque là, le pas qui donne la plus petite valeur de fonction objectif
3. α_{hi} est choisi de façon à vérifier: $\Phi'(\alpha_{lo}).(\alpha_{hi} - \alpha_{lo}) < 0$

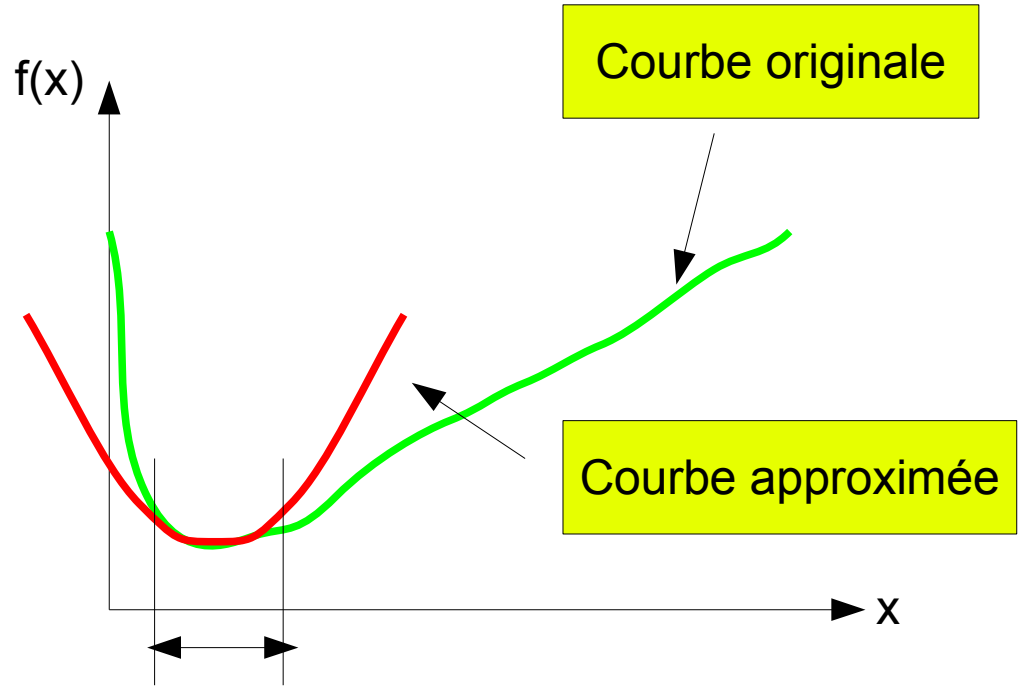
Condition de base non vérifiée

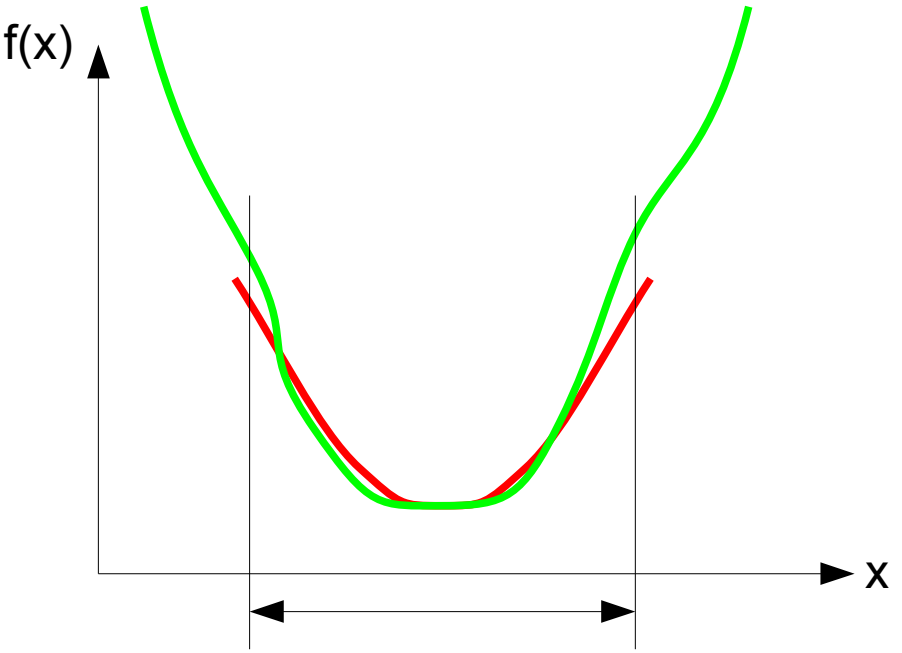
Condition de Wolfe Forte vérifiée

$\Phi(\alpha) = f(x_k + \alpha p_k)$

La Méthode de Région de Confiance

Introduction





La Méthode de Région de Confiance

Notations

On a un modèle :

$$m_k(p) = f(x_k) + \nabla f(x_k)^t \cdot p_k + \frac{1}{2} p_k^t \cdot B_k \cdot p_k$$

Et le développement au second ordre:

$$f(x_k + t \cdot p_k) = f(x_k) + \nabla f(x_k)^t \cdot p_k + \frac{1}{2} p_k^t \cdot \nabla^2 f(x_k + t \cdot p_k) \cdot p_k$$

Si $B_k = \nabla^2 f(x_k + t \cdot p_k) \rightarrow$ le modèle est en accord avec la fonction

On cherche $\min_p m_k(p)$ avec $\|p\| \leq \Delta_k$ → Défini la taille de la région de confiance

Calcul du rayon de confiance:

$$\rho_k = \frac{f(x_k) - f(x_k + p_k)}{m_k(0) - m_k(p_k)}$$

→ Réduction actuelle

→ Réduction prédite

si $\rho_k < 0$ on rejette l'itération

si $\rho_k \simeq 1$ modèle en accord avec la réalité

La Méthode de Région de Confiance

Algorithme

On choisit $\Delta_{\max}, \Delta_0 \in]0, \Delta_{\max}]$ et $\eta \in [0, 1/4[$

for $k = 0$ to N

  calcul de p_k par résolution de $\min_p m_k(p)$

  évaluer ρ_k

  si $(\rho_k < 1/4)$

    $\Delta_{k+1} = 1/4 \|p_k\|$

  sinon

    si $(\rho_k > 3/4)$ et $(\|p_k\| = \Delta_k)$

      $\Delta_{k+1} = min(2\Delta_k, \Delta_{\max})$

    sinon

      $\Delta_{k+1} = \Delta_k$

    end sinon

    si $(\rho_k > \eta)$

      $x_{k+1} = x_k + p_k$

    sinon

      $x_{k+1} = x_k$

    end sinon

  end sinon

end for

Si la qualité l'adéquation du modèle avec la réalité n'est pas suffisante, on réduite le domaine de validité du modèle

Sinon, on double le domaine de validité du modèle, dans la limite définie par Δ_{max}.

Sinon, le domaine de validité du modèle no change pas.

Dés que la qualité a atteint un niveau suffisant, on applique le vecteur de déplacement p_k à la variable x_k

La Méthode de Région de Confiance

Résolution approché du modèle

Résolution d'une approximation linéaire du problème original sous contraintes

Résolution approché de $m_k(p)$

- Calcul du point de Cauchy

$$p_k = arg\min_p f(x_k) + \nabla f(x_k)^t p_k \text{ avec } \|p_k\| \le \Delta_k$$

Domaine de confiance du modèle

- Méthode DogLeg → opt++
  Approximation de la courbe par deux droites
- Minimisation dans le sous-espace bi-dimensionnelle
  Méthode DogLeg sophistiquée
- Méthode de Steihaug
  Méthode basée sur la méthode du gradient conjuguée
- Méthode de résolution presque exacte

On a un modèle quadratique de dimension n, il sera résolu en n itérations

Conclusions

- Beaucoup de méthodes
- Toutes ont des défenseurs et des critiques (sauf la descente du gradient qui n'a que des critiques)
- Beaucoup de récents développements
- Quelques références:

J. Nocedal, S. J. Wright, « Numerical Optimization », Springer-Verlag, 2006
D. M. Himmelblau, « Applied Nonlinear Programming », McGraw-Hill, 1972
G. Vanderplaats, « Numerical Optimization Techniques: with Applications », McGraw-Hill, 1984
R. T. Haftka, Z. Gürdal, « Elements of Structural Optimization », Kluwer Academic Publisher, 1992