Introduction à l'Optimisation Classique

- Majeure Science des Données -

Notes de cours
Novembre 2019

Eric Touboul

Généralités

1.1 Quelques exemples de situations relevant de problèmes d'optimisation

- Optimisation de procédés à l'aide de logiciels de simulation
  - Exemple : position optimum de puits de production pétrolière
- Calage de modèles, problèmes inverses
  - Exemple : détermination de la capacité calorifique d'un matériau
- Identification de paramètres de lois de probabilité
  - Exemple : détermination de (μ,σ) d'une loi par max de vraisemblance
- Mesures indirectes
  - Exemple : mesure de la taille de particules par diffraction de la lumière
- Résolution de systèmes d'équations
  - $F(x)=0$ sss $\|F(x)\|^2$ minimum
- …etc. …

1.2 Grandes classes de problèmes d'optimisation

- Optimisation discrète :

L'ensemble des possibilités est fini, ou dénombrable (recherche opérationnelle)
Exemple : recherche d'un optimum parmi les feuilles d'un graphe.
Programmation linéaire : bien qu'ayant une formulation continue, (minimum d'une fonction affine sur un polyèdre), l'ensemble des solutions se cherche en parcourant les sommets du polyèdre qui sont en nombre fini

- Optimisation continue

L'ensemble des possibilités est un continuum. Par exemple R^n (optimisation en dimension finie) , un espace de Hilbert – espace fonctionnel (optimisation en dimension infinie)

- Optimisation contrainte

La variable n'appartient pas à l'espace tout entier, mais à une partie de cet espace définie par des « contraintes », ie des conditions que la variable doit respecter.

- Optimisation non linéaire (contraire : linéaire)

La fonction à optimiser n'est pas linéaire en la variable. C'est un cas très courant.

- Optimisation convexe

La fonction à optimiser, et l'ensemble des contraintes, sont des convexes. Dans ce cas de figure, on peut démontrer des résultats d'existence et d'unicité de l'optimum.

- Optimisation locale (contraire : globale)

On cherche un minimum local de la fonction, sans garantie qu'il soit global. Le point de départ a une grande influence sur le résultat.

- Optimisation multi – critère (contraire : mono-critère)

Compromis entre plusieurs critères, c'est-à-dire plusieurs fonctions à « optimiser » en même temps. Voir « front de Pareto »

- Méthodes stochastiques (contraire : déterministes)

Chaque pas d'évolution vers la solution cherchée fait appel à un processus aléatoire. Ces méthodes sont principalement utilisées pour l'optimisation globale, et évitent le recours au calcul de dérivées (idéal pour fonctions objectif bruitées) . Elles permettent notamment des phases « d'exploration » aléatoires. Exemple : recuit simulé, algorithmes évolutionnaires.

1.3 Cadre de ce cours

Optimisation continue (en général non linéaire) , déterministe, mono-critère, recherche d'optimums locaux. On étudiera cependant à la fin, sur la base d'un TP, une méthode stochastique (CMA-ES pour covariance matrix adaptation – evolution strategy)

J : fonction de $R^n \to R$ dans un premier temps
(Puis fonctionnelle de $E \to R$, E espace de Hilbert, e.g. espace de fonctions)

$Min(J(x)) \qquad avec \quad x \in C \subset R^n$

C : espace des contraintes
$C=R^n$: optimisation non contrainte

J est supposé le plus souvent C^2. J est appelé fonction coût ou critère.
La matrice jacobienne d'une fonction F sera noté J_F ou Jac(F), à ne pas confondre avec J.

Les points où J atteint son minimum, s'ils existent, sont appelés "minimiseurs" dans la littérature anglo-saxone. (Terme qui permet de distinguer le minimum de J et le point ou il est atteint). Ils sont quelquefois nommés "arguments" dans la littérature française (x^*=argmin(J(x)))

Minimum local: x est un "minimiseur" local de J sur C s'il existe une boule B de centre x telle que J atteint son minimum en x sur $B \cap C$, ie : $\forall x \in B \cap C, J(x) \geq J(x^*)$

Ouvrages utilisés : « Introduction à l'Optimisation » Jean-Christophe Culioli, Ellipses
« Programmation mathématique », Michel Minoux, Dunod

On pourra aussi consulter en ligne les ouvrages suivants :

Cours de l'Ecole des Mines de Nancy, X. Antoine et al. : http://math.unice.fr/~dreyfuss/P13.pdf
Cours de l'Ecole des Mines de Paris, N. Petit : http://cas.ensmp.fr/~petit/ESoptimisation/poly.pdf
http://public.telecom-bretagne.eu/~chonavel/poly_optim_00/poly_optim_00.htm

2 Outils et notions de base

2.1 Dérivée suivant un vecteur, différentielle Gâteaux

Si J est une fonction de $R^n \to R$, u un vecteur unitaire de R^n : $\frac{\partial J}{\partial u}(x) = \lim_{\varepsilon \to 0} \frac{J(x+\varepsilon u) - J(x)}{\varepsilon}$

J est différentiable en x si : $\exists v \in R^n / \forall u, \|u\| = 1, \frac{\partial J}{\partial u}(x) = {}^T v.u$

Attention : J dérivable en x dans toutes les directions n'implique pas J différentiable en x.

Si J est une fonctionnelle $E \to R$, v un élément de E (eg, une fonction)

$\partial \mathrm{J}_{\mathrm{v}}(x) = \lim_{\varepsilon \to \to 0} \frac{J(x+\varepsilon v) - J(x)}{\varepsilon}$ (différentielle Gâteaux de J suivant v)

Exercice : u est une fonction de $\Omega \subset R^n \to R$, soit la fonctionnelle (de $C^1(\Omega) \to R$)

$J(u) = \frac{1}{2}\int_{\Omega} (\nabla u)^2 d\Omega + \int_{\Omega} (fu) d\Omega$, calculer $\partial \mathrm{J}_{\mathrm{v}}(u)$, calculer $\partial \mathrm{J}_{\mathrm{v}}(u)$

On trouve : $\partial \mathrm{J}_{\mathrm{v}}(u) = \int_{\Omega} (\nabla u \nabla v) d\Omega + \int_{\Omega} (fv) d\Omega$ (formulation variationnelle d'une edp de Laplace)

2.2 Gradient

Définition : le gradient de *J* au point $x \in R^n$, noté $\nabla J(x)$, ou $J'(x)$, ou encore g (anglo-saxon) est le vecteur de R^n dont la ième composante est:

$(\nabla J(x))_i = \frac{\partial J}{\partial x_i}(x)$

Dérivée dans une direction: *exercice*: montrer que, pour $u \in R^n, \|u\| = 1$ $\frac{\partial J}{\partial u}(x) = {}^T u.(\nabla J(x))$

On notera indifféremment le produit scalaire ${}^T u.v$ ou $\langle u, v \rangle$ ou meme $u.v$

Formule de Taylor à l'ordre 1: $J(x+d) = J(x) + < \nabla J(x) , d > + O(\|d\|^2)$

Courbes de niveau et gradient:

Courbe de niveau $\Gamma_a = \{x \in R^n / J(x) = a\}$

Exercice : montrer que le champ de gradient de J est orthogonal aux courbes de niveaux de J.

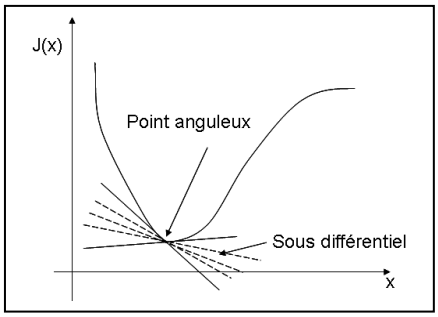
exercice: $J(x) = {}^T x A x + b^T x$, A matrice nxn . calculer $\nabla J(x)$

Le gradient est une propriété géométrique de la courbe. Il ne dépend pas du repère choisi. Il dépend par contre de la métrique choisie.

Calcul numérique du gradient : dans la plupart des cas, le gradient n'est pas connu : il faut alors le calculer numériquement par formules aux différences.

$(\nabla J(x))_i \approx \frac{J(x+\varepsilon e_i) - J(x)}{\varepsilon}$, pour « ε petit »

Exercice : le programmer



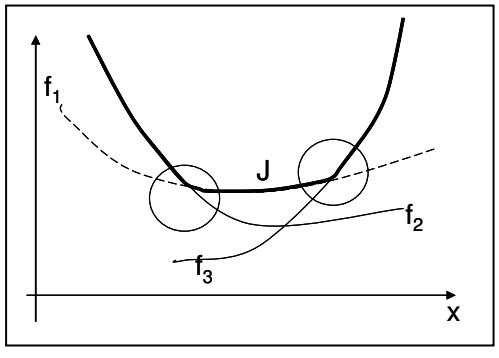
2.3 Sous Gradient

Si J n'est pas différentiable en x, on peut définir un sous-gradient :
Pour une fonction convexe, la tangente est sous la courbe :

$J(x) \geq J(x_0) + {}^T \nabla J(x_0).(x - x_0)$

g est sous gradient de J en x_0, si, dans un voisinage de x_0 :

$J(x) \geq J(x_0) + {}^T g.(x - x_0)$



Sous-différentiel : ensemble des sous gradients
Intérêt : dans les où J n'est pas partout différentiable,
les algorithmes de gradient classiques marchent en utilisant les sous-gradients (un sous-gradient quelconque dans le sous-différentiel).

Exemple : si $J(x) = \max(f_i(x))_{i=1..n}$, la recherche de $\min(J(x))$ nécessite l'utilisation du sous-gradient (J n'est pas partout différentiable)

2.4 Matrice Hessienne

Définition:

$$H_J(x) = \begin{bmatrix} \frac{\partial^2 J}{\partial x_1^2}(x) & \dots & \frac{\partial^2 J}{\partial x_1 \partial x_n}(x) \\ \vdots & \vdots & \vdots \\ \frac{\partial^2 J}{\partial x_d \partial x_1}(x) & \dots & \frac{\partial^2 J}{\partial x_n^2}(x) \end{bmatrix} = \left(\frac{\partial^2 J}{\partial x_i \partial x_j}(x) \right).$$

Cette matrice (symétrique) est aussi notée J"(x) , ou G(x) (anglo-saxon)
Elle est très lourde à calculer, aussi s'est-on ingénié à trouver des méthodes qui peuvent soit l'éviter, soit la remplacer par une matrice approchée (cf : methodes de quasi-Newton : construction d'une suite de matrices simples et non coûteuses à calculer convergeant vers la matrice Hessienne)

Développement de Taylor d'une fonction de plusieurs variables :

$$J(x+d) = J(x) + \langle \nabla J(x), d \rangle + \frac{1}{2} \langle H(x)d, d \rangle + O(\|d\|^3)$$

exercice: montrez le pour n=2.

Matrice Hessienne et courbure:

Si en x la matrice hessienne est définie positive (toutes ses valeurs propres >0) , alors J est localement strictement convexe. Dans une base de directions propres, les valeurs propres sont les inverses des rayons de courbure dans les directions propres. (Si H n'est pas positive, les valeurs propres négatives correspondent à des directions où J est concave.)

Conditionnement: Localement, les surfaces d'isovaleurs "ressembleront" d'autant plus à des sphères que les valeurs propres de H seront voisines (H bien conditionnée: rayons de courbures dans des directions différentes pas trop différents les uns des autres - cas idéal: la sphère). Le bon conditionnement de H sera important pour le bon fonctionnement des méthodes d'optimisation en général.

2.5 Matrice Jacobienne

Définition : F de R^n dans R^p

$Jac(F(x)) = J_F(x) = \begin{bmatrix} \frac{\partial F_1}{\partial x_1} & \frac{\partial F_1}{\partial x_2} & \dots & \frac{\partial F_1}{\partial x_n} \\ \frac{\partial F_2}{\partial x_1} & \frac{\partial F_2}{\partial x_2} & & \frac{\partial F_2}{\partial x_n} \\ \dots & \dots & \dots & \dots \\ \frac{\partial F_p}{\partial x_1} & \frac{\partial F_p}{\partial x_2} & & \frac{\partial F_p}{\partial x_n} \end{bmatrix}$, c'est la matrice dont les lignes sont les gradients des F_i. On pourra, si le contexte est clair, la noter F'.

Développement de Taylor d'une fonction de R^d dans R^p

$F(x+d) = F(x) + Jac(F(x))d + O(\|d\|^2)$

On a: $Jac(\nabla J(x)) = H_{J(x)}$

2.6 Dérivée de fonction composée

exercice: montrer que:

$\frac{d}{d\alpha}(J(u(\alpha))) = \langle \nabla J(u(\alpha)), u'(\alpha) \rangle$ où α est un réel et u' est le vecteur des dérivées des u_i

exercice: calculer les dérivées première et seconde en α de $J(x+\alpha d)$, x et d dans R^d

exercice : retrouver le développement de Taylor avec reste intégral :

$J(x+h) = J(x) + \int_0^1 \langle \nabla J(x+th), h \rangle dt$ (produit scalaire dans l'intégrale)

exercice: montrer que:

$\frac{d^2}{d\alpha^2}(J(u(\alpha))) = (u''(\alpha)^T) \nabla J(u(\alpha)) + (u'(\alpha)^T) \left[H_{J(u(\alpha))} \right] u'(\alpha)$ et en déduire, en prenant $u=x+\alpha h$, le lien entre H def > 0 et J convexe

2.7 Espace des contraintes

Les contraintes sont classiquement définies à partir d'une famille de fonctions $h_i(x)$ $R^n \to R$ (notation pour les contraintes égalité) ou $g_i(x)$ (notation pour les contraintes inégalité) de la façon suivante

$C = \{ x \in R^n / h_i(x) = 0 \forall i = 1...p \}$ p<n ou $C = \{ x \in R^n / g_i(x) \le 0 \forall i = 1...p \}$

Remarque : une contrainte égalité peut être remplacée par deux contraintes inégalités.

Une contrainte inégalité $g_i(x) \leq 0$ est dite « active » si, x^* étant solution du problème d'optimisation, $g_i(x^*) = 0$. (en général, si on relaxe (ie supprime) une contrainte active, la solution du problème n'est plus la même)

Direction admissible

Si x est un point de C, on dit que d, vecteur de R^n, est une direction admissible en x si
$\exists \alpha > 0, [x, x + \alpha d] \subset C$

2.8 Quelques exemples

- Minimisation du risque d'un portefeuille boursier à rendement fixé (Markowitz) (optimisation en dimension finie) : voir TD
- Formulation énergétique d'un problème de thermique et forme variationnelle(optimisation en dimension infinie)

3 Critères d'optimalité

3.1 Théorème de Weierstrass :

C'est un théorème de base que nous citons sans démonstration :

- Une fonction J continue sur un compact (pour R^n : un fermé-borné) atteint ses bornes.
- Corollaire : si J est continue sur R^n et vérifie $J(x) \to +\infty$ quand $\|x\| \to +\infty$ (propriété qui joue un rôle similaire à la convexité) alors $\exists x^* = arg\ min\ J(x)$.

3.2 Condition nécessaire du premier ordre:

Si J est différentiable, et si x* est un minimum local, alors :
- Pour toute direction admissible en x^*, d, ${}^T d.\nabla J(x^*) \geq 0$ (inégalité d'Euler)

- Si $\nabla J(x^*) = 0$ alors $H_J(x^*)$ est semi-défini positif sur l'ensemble des directions admissibles (ie $d^T H_J(x^*) d \geq 0 \quad \forall d$ direction admissible)

Exercice : démonstration (utiliser un DL à l'ordre 1)
Exercice : si C est un espace affine : $\nabla J(x^*)$ est orthogonal à C

3.3 Condition suffisante :

Si J est deux fois différentiable, et si, en x^* *point intérieur* de l'ensemble des contraintes
- $\nabla J(x^*) = 0$ et
- $H_J(x^*)$ est défini positif (ie $u^T H_J(x^*) u > 0 \quad \forall u \neq 0$)

Alors x* est un minimum local strict (unique dans un voisinage de x^*) de J

Exercice : preuve.

3.4 Cas convexe :

Sans contraintes :

Si J est convexe sur R^n, alors tout minimum local est global
Si J est strictement convexe, elle admet un minimum global unique

Avec contraintes :

Problème convexe : J **et** C sont convexes
Alors : tout minimum local est global

4 Principes des différentes méthodes numériques d'optimisation non contrainte

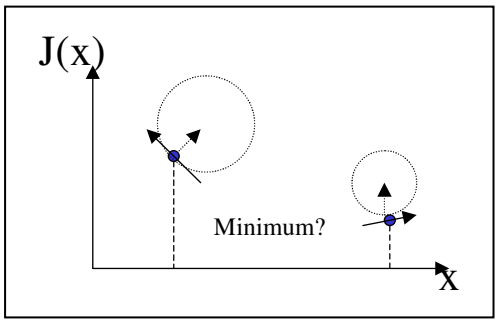
Les méthodes que nous étudions ici sont basées sur la construction d'une suite x^k de R^n convergeant vers un minimum local x^*. Ses suites ont la propriété d'être des suites (idéalement…) « descendantes » pour J , c'est-à-dire que l'on veut que : $J(x^{k+1}) < J(x^k) \quad \forall k$

Un résultat très général, s'appliquant à des classes d'algorithmes, assure la convergence des algorithmes utilisés. Il s'agit du théorème de convergence global que nous énoncerons après avoir présenté une ou deux méthodes afin de le rendre plus concret dans l'esprit du lecteur.

Avant de rentrer dans les détails, nous donnons une description générale des différentes méthodes :

4.1 Généralités

Difficulté: On n'a qu'une vision locale de la fonction à minimiser. On peut en général évaluer en un point:
- sa valeur - sa pente - sa courbure (coûteuse à estimer)



Ces évaluations coûtent généralement cher en calculs.
Une méthode d'optimisation sera performante si:

- le nombre d'évaluations à effectuer est minimum
- chaque évaluation est peu coûteuse

Comme pour la méthode de la sécante, il est souvent efficace de remplacer un calcul exact (dérivée) par un calcul approché (pente de la corde).

4.2 Critères d'optimalité, Optimisation convexe - cas non contraint -

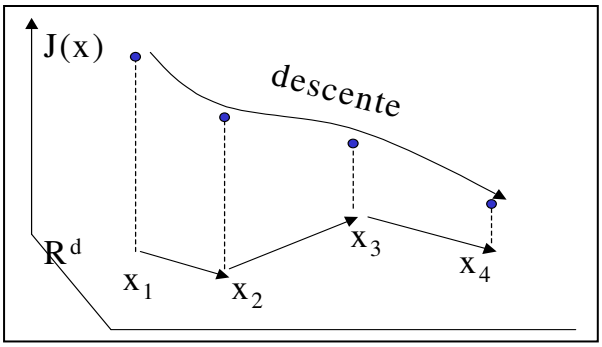
Si J est C^1 et si x est un "minimiseur" de J, alors grad(J(x))==0 (condition nécessaire d'optimalité)

Si J est C^2 et si x est un minimiseur de J, alors grad(J(x)==0, et le Hessien de J en x est semi-défini positif (pour tout s non nul, $s^THs>=0$) . Ce qui est équivalent à : J localement convexe.

Si J est C^2 et si en x grad(J(x)==0, et le Hessien de J en x est défini positif alors x est un minimiseur local de J. (J localement strictement convexe)

Si J est strictement convexe sur R^d, le minimiseur existe et est unique.

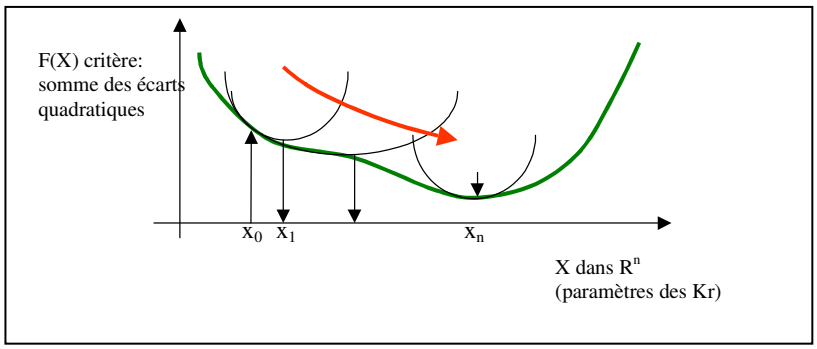
4.3 Méthodes d'ordre 1



Direction admissible, descente (suivant une direction faisant un angle obtus avec le gradient).
Avantages: amélioration à chaque étape
Inconvénient: vitesse de convergence
lente dans les cas mal conditionnés (d'où certaines amélioration des méthodes de base)

4.4 Méthodes d'ordre 2

Interpolation locale par une forme quadratique, obtenue par exemple par troncature de la formule de Taylor. Méthodes de type "Newton" .Avantages: rapidité de la convergence (convergence quadratique au voisinage de la solution) , inconvénient: calcul de la matrice Hessienne lourd, d'où des approximations utilisées pour l'éviter.



Les vecteurs $d^k = x^{k+1} - x^k$ sont appelés directions de Newton. Ce ne sont en général pas systématiquement des directions de descente comme dans les méthodes de descente.

4.5 Cas particulier des moindres carrés non linéaires

$J(x) = \frac{1}{2}\sum_{i=1}^{p}\left(f_i(x)\right)^2$. La structure particulière de J permet, dans certains cas, de négliger les termes contenant le Hessien.

4.6 Approximation des dérivées par différences finies

Dans beaucoup de cas pratiques, on ne connaît pas de forme analytique pour le critère J. Les gradients et Hessien peuvent alors être approchés par des formules aux différences classiques. Ceci peut être cependant très coûteux, puisqu'il faut au minimum n+1 évaluations de la fonction J pour calculer le gradient en un point J (différences finies décentrées)

4.7 Problèmes de grande taille

Si q est très grand (cela arrive couramment, lorsque le problème d'optimisation provient de la discrétisation d'un problème d'EDP par exemple), deux difficultés peuvent se rencontrer:
- Le Hessien est très lourd à calculer
- Le problème est mal conditionné

Des méthodes spécifiques sont employées, comme le gradient conjugué pré-conditionné ou les méthodes de type "région de confiance".

4.8 Régions de confiance (trust region)

On contraint explicitement, dans les méthodes de type Newton (approximation quadratique), la recherche autour du point courant à se faire dans une boule centrée sur ce point (région dans laquelle l'approximation est valide). Cela conduit à des méthodes de type Levenberg-Marquard.

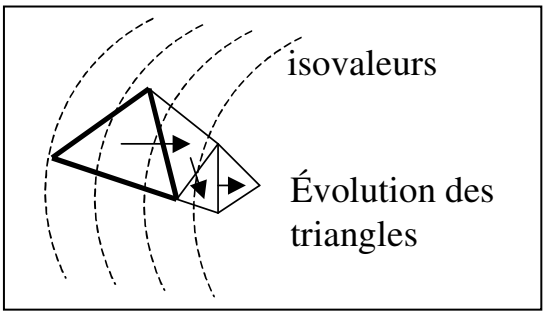
De façon générale:
- on remplace localement la fonction par un modèle (eg: modèle quadratique)
- on cherche le minimum du modèle sur une région autour du point courrant
- si le minimum du modèle ne correspond pas à un progrès significatif (descente), on diminue la taille de la région de confiance, ou même on change le modèle lui même, et on recommence. Sinon, on prend le point trouvé comme nouveau point.

On pratique donc à chaque étape une optimisation contrainte;

4.9 Méthodes sans calcul de dérivée

Méthode de Nelder et Mead (dite du simplexe, à ne pas confondre avec la méthode de programmation linéaire du même nom) .

Elle consiste à évaluer J aux 3 sommets d'un "hyper-triangle", de chercher le sommet sur lequel J est maximal, puis de transformer le triangle en remplaçant ce sommet par son symétrique par rapport à la base opposée, ou encore par rapport au centre du triangle. La méthode prévoit également une diminution de la taille des triangles quand on se rapproche de la solution.



5 Méthodes numériques d'optimisation non contrainte

5.1 Recherche mono-dimensionnelle

Les méthodes de descente, à chaque itération, nécessitent une recherche de minimum dans la direction de descente: $x_{k+1}=x_k+\alpha_k d_k$. Au début de leur existence (années 50), les créateurs de ces méthodes préconisaient une recherche exacte du minimum dans les directions de descente. Puis on s'est rendu compte que cette stratégie était pénalisante pour la performance (perte de temps). Les méthodes modernes (Armijo) préconisent d'emblée une recherche très approchée, mais bien sur qui devient de plus en plus précise quand on se rapproche de l'optimum.

La recherche du minimum dans une direction peut se traiter comme la recherche du zéro de la dérivée sur la droite de descente (ce qui ne garantit un minimum que si J est convexe sur cette droite). On peut alors utiliser les méthodes classiques de recherche de zéro (Newton, sécante, dichotomie...).

Cette recherche conduit donc souvent elle même à un processus itératif, pour approcher le minimum de $f(\alpha)=J(x+\alpha d)$.

5.1.1 Interpolation quadratique

Il s'agit d'interpoler f par un polynôme de degré 2 en se servant des valeurs J ou de sa dérivée (3 valeurs nécessaires) pour calculer a, b, et c:

$$f(\alpha)=a\alpha^2+b\alpha+c$$

Puis, l'optimum (minimum seulement si a>0 $\alpha=\frac{-b}{2a}$. On peu vérifier que, si on a les valeurs de J en 3 points sur la droite de recherche:

$$x^{k+1}=\frac{\beta_{23}J(x_1)+\beta_{31}J(x_2)+\beta_{12}J(x_3)}{\gamma_{23}J(x_1)+\gamma_{31}J(x_2)+\gamma_{12}J(x_3)}$$

avec
$$\beta_{ij}=x_i^2-x_j^2$$
$$\gamma_{ij}=x_i-x_j$$

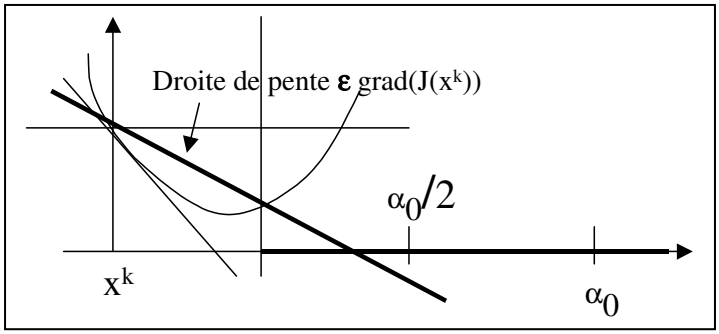
5.1.2 Interpolation cubique

$$f(\alpha)=a\alpha^3+b\alpha^2+c\alpha+d$$

5.1.3 Méthode d'Armijo

Sur la droite de recherche, on veut trouver un point $x^{k+1}=x^k+\alpha d$ qui réalise une descente, sans pour autant réaliser le minimum (trop coûteux).
L'algorithme est le suivant:

On note u le vecteur directeur de la droite, unitaire
Et dirigé dans le sens de la recherche (descente)



On note α l'abscisse sur cette droite, avec α=0 pour le point x^k.

La pente (négative) en x^k dans la direction u est Donnée par le produit scalaire

$$p = \nabla J(x^k).u$$

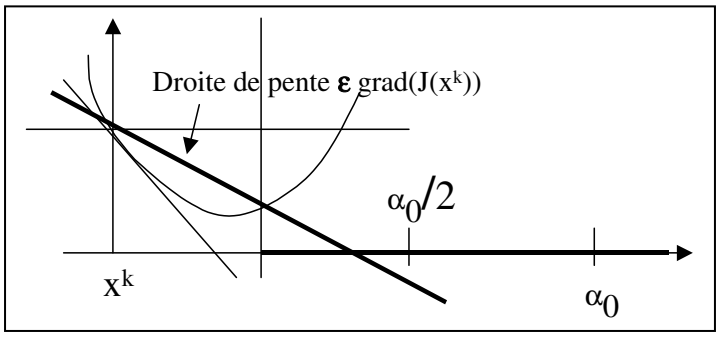
On considère la droite (en gras) de pente εp, ε dans [0,1]. (donc comprise entre la pente de la tangente en x^k et la pente de la droite horizontale). E est un paramètre de « réglage de la méthode » et assure une descente « minimale »

L'équation de cette droite est : $m(\alpha) = J(x^k) + \alpha\varepsilon p$

On cherche le α le plus grand tel que $J(x^k+\alpha d)$ soit sous la droite, ce qui assure que l'on réalise une descente.

Pour ce faire, on choisit α_0, puis on divise successivement sa valeur par 2, jusqu'a obtenir la condition désirée:

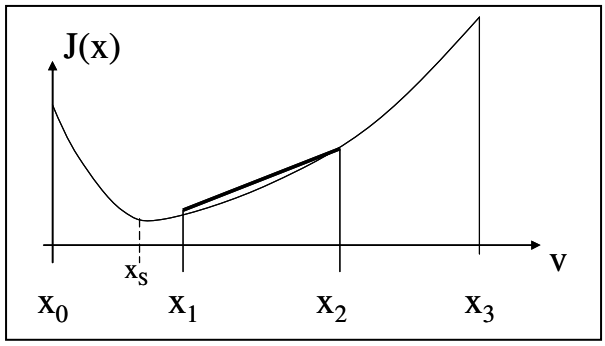
On prend le plus petit entier i tel que:
$J(x^k + 2^{-i}\alpha_0 u) < J(x^k) + \alpha\varepsilon p$ et donc:
$x^{k+1} = x^k + 2^{-i}\alpha_0 u$



5.1.4 Méthode de trichotomie (voir aussi section dorée)

On dispose d'une fonction J de R^n dans R, d'un point de départ x_0 et d'une direction v qui définit une demi-droite D à partir de x_0. On suppose J convexe sur D. On cherche $x_s = x_0 + \alpha v$ réalisant $\underset{\alpha>0}{Min} J(x_0 + \alpha v)$

- On détermine un point x_3 de D, tel que $x_s \in [x_0, x_3]$
- On divise le segment $[x_0, x_3]$ en 3, en introduisant deux points : x_1, x_2 (voir dessin)
- Suivant le signe de $p = J(x_2) - J(x_1)$ (signe de la pente de la corde), on élimine le segment $[x_0,x_1]$ ou le segment $[x_2,x_3]$. (justification : croissance de la dérivée d'une fonction convexe):
  - $p \geq 0$, on élimine $[x_2,x_3]$.
  - $p \leq 0$, on élimine $[x_0,x_1]$.
- On recommence (critère d'arrêt)



Voir aussi : section dorée

5.2 Méthode de relaxation

Rappel
Avant d'aborder des méthodes spécifiques, on donne quelques idées générales à travers deux exemples.

Exemple 1. $J(x_1, \ldots, x_n) = J_1(x_1) + \ldots + J_n(x_n)$.

Dans un tel cas, les variables sont dites "entièrement découplées". L'optimum est fournit en d recherches linéaires en minimisant successivement par rapport aux axes. Bien sûr, rien n'empêche d'utiliser cette méthode lorsque J est quelconque, c'est-à-dire en minimisant successivement par rapport aux axes et en répétant cette opération un certain nombre de fois. Mais il faudra s'attendre à une convergence assez lente, convergence qui va dépendre de la nature des termes de couplage.

Exemple 2. $J(x_1, \ldots, x_n) = J_1(x_1, \ldots, x_n) + J_2(x_{p+1}, \ldots, x_n)$.
Le problème se ramène à deux sous-problèmes indépendants et de taille moindre.

5.3 Méthodes de gradient

5.3.1 Méthode de la plus profonde descente (à pas optimal)

Rappel : On descend dans la direction opposée au gradient, sur une droite, en minimisant la fonction J sur la droite de descente:

$x^{k+1} := x^k - \alpha_k \nabla J(x^k)$

$$f_{d^k}'(\alpha_k) = < \nabla J(x^k + \alpha_k d^k) , d^k > = 0, \text{ où } d^k := - \nabla J(x^k).$$

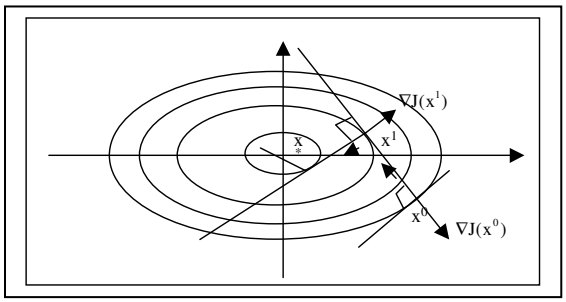
Or

$$\nabla J(x^k + \alpha_k d^k) = \nabla J(x^{k+1}) = - d^{k+1},$$

donc

$$< d^k , d^{k+1} > = 0$$

Deux directions de recherche consécutives sont orthogonales, ce que l'on illustre de la manière suivante.



Propriétés de convergence. Si l'on est dans un "bon" voisinage d'un minimum local x^* satisfaisant les conditions du théorème 2 (localement, le graphe est celui d'un paraboloïde elliptique), la suite $(x^k)_k$ converge vers x^*, convergence d'ordre 1[1], plus ou moins bonne selon que le conditionnement du hessien en x^*, $\frac{\max(\lambda_i(x^*))}{\min(\lambda_i(x^*))}$, est grand devant 1.

Il est bon de repenser au dessin ci-dessus en imaginant une ellipse aplatie. Dans ce cas, J s'écrit dans le repère défini par l'ellipse (forme réduite)

$$J(x) = 1/2(\lambda_1 x_1^2 + \lambda_2 x_2^2) \text{ où } \lambda_1 << \lambda_2.$$

Dire que l'ellipse est aplatie, c'est dire que le rapport des courbures $\frac{\lambda_2}{\lambda_1}$ est grand devant 1, c'est-à-dire que le la matrice hessienne est mal conditionnée. Dans le cas général, on peut simplement garantir que l'on a diminué la fonction *J* à chaque étape.

Exercice: Si J est une forme quadratique, *J(x) := 1/2 < Hx, x > + < b , x > + b* (*H* sym définie positive), montrer que $\alpha_k = - \frac{< \nabla J(x^k), d^k >}{|| d^k ||_H^2}$ où $|| d^k ||_H^2 = <Hd^k , d^k>$.

Remarque : dans ce cas $x^* = \arg\min_{x \in R^n}(J(x))$ est aussi l'unique solution du système $Hx^* = -b$

5.3.2 Méthode du gradient conjugué

Remarque préalable : si J est une forme quadratique, $J(x) = \frac{1}{2}{}^T xHx + b$, avec H symétrique définie positive (ie H définit un produit scalaire sur R^n) - J est alors strictement convexe - , et si $\{d^0, ...d^{n-1}\}$ est une base de R^n formée de vecteurs 2 à 2 orthogonaux pour le produit scalaire H (deux à deux conjugués, ie : $i \neq j \Rightarrow \langle d^i, Hd^j \rangle = 0 (= \langle Hd^i, d^j \rangle = \langle d^i, d^j \rangle_H)$

Alors on a explicitement une solution du système $Hx = -b$, ou encore on a explicitement le minimiseur de J, x^*

En effet : x^* peut s'écrire dans la base $\{d^0, ...d^{n-1}\}$: $x^* = \sum_0^n \alpha_i d^i$ et en multipliant à droite par ${}^T d^k H$ il vient : $\langle Hd, x^* \rangle = \langle Hx^*, d \rangle = \langle -b, d \rangle_H = \alpha_k \|d^k\|_H$ d'où les coordonnées : $\alpha_k = \frac{-\langle b, d \rangle_H}{\|d^k\|_H}$.

La méthode du gradient conjugué conduit à construire cette base en prenant des directions de descente deux à deux conjuguées.

Théorème : si J est une forme quadratique et si on pratique une méthode de descente avec des directions mutuellement conjugués, avec à chaque pas une recherche monodimensionnelle exacte, alors :

x^k réalise le minimum de J sur $V^k = \{d^0, ...d^{k-1}\}$, et en particulier, x^n réalise le minimum de J sur R^n.

Les méthodes de gradient conjugué utilisent cette propriété
Les directions de descente sont prises deux à deux conjuguées. On peut montrer qu'elles sont alors mutuellement conjugués

$$d^1 = - \nabla J(x^1) + \beta d^0 \text{ (nécessairement direction de descente)}$$

Cas d'une forme quadratique

Dans le cas d'une fonction quadratique, le calcul du gradient est immédiat.
Exercice: calculer le coefficient β. en utilisant la condition la condition $< d^1 , d^0 >_H = 0$.
On trouve :

$$\beta = \frac{< d^0 , \nabla J(x^1) >_H}{\| d^0 \|_H^2}$$

Algorithme de gradient conjugué pour une fonction quadratique définie positive

- Choisir un point initial x^0
  Le premier pas est un pas de gradient à pas optimal simple, en prenant la plus grande pente :
- Calculer la direction d plus grande pente : $d^0 = -\nabla J(x^0)$
- Calculer le pas optimal $\alpha_0 = -\frac{\left(\nabla J(x^0), d^0\right)}{\left\|d^0\right\|_H^2} = \frac{\left\|d^0\right\|^2}{\left\|d^0\right\|_H^2}$
- Calculer le premier itéré $x^1 = x^0 + \alpha_0 d^0$
  Ensuite, la direction n'est plus celle de la plus grande pente, mais utilise la conjugaison (H orthogonalité)
- (*) Etape k : si $d^{k-1} \neq 0$ (sinon, on a atteint le minimum)
- On calcule la direction conjuguée : $d^k = -\nabla J(x^k) + \beta_k d^{k-1}$ avec $\beta_k = \frac{\left\langle d^{k-1}, \nabla J(x^k)\right\rangle}{\left\|d^k\right\|_H^2}$
- Puis on calcule le pas optimal $\alpha_k = -\frac{\left(\nabla J(x^k), d^k\right)}{\left\|d^k\right\|_H^2}$
- On calcule le nouvel itéré : $x^{k+1} = x^k + \alpha_k d^k$ et on retourne à (*)

Résultat. Les directions de recherche $d^0, \ldots, d^n$ construites par cet algorithme sont ***H-conjuguées*** (i.e. forment une famille orthogonale pour $<.,.>_H$) et libre . Les vecteurs gradient $\nabla J(x^0), \ldots, \nabla J(x^n)$ une **famille orthogonale** pour le produit scalaire usuel $<.,.>$.

Conséquence. Cet algorithme converge de manière exacte (en théorie) vers le minimum absolu de *J* en **au plus** n itérations (= nombre de valeurs propres distinctes de H).

Exercice : montrer que $\nabla J(x^{k+1})$ est orthogonal à $\left\{\nabla J(x^k)\right\}_{k=0\ldots p}$.

Généralisation

Se pose maintenant le problème de la généralisation. A l'étape k+1, faut-il calculer $H = J''(x^k)$ *? En fait, non:*
Reprenons l'expression de β et tâchons de faire disparaître *H*. On a

$$\beta = \frac{< d^0, \nabla J(x^1) >_H}{\| d^0 \|_H^2}$$

où $x^1 := x^0 + \alpha_0 d^0$; $d^0 := - \nabla J(x^0)$.
Par ailleurs,

$$\nabla J(x) = Hx + a.$$

Donc

$$\nabla J(x^1) - \nabla J(x^0) = \alpha_0 H d^0$$

et

$$\alpha_0 < d^0 , \nabla J(x^1) >_H = < \nabla J(x^1) - \nabla J(x^0) , \nabla J(x^1) > = \| \nabla J(x^1) \|^2$$

Pour le premier pas, on a bien :

$$\alpha_0 < d^0 , d^0 >_H = < \nabla J(x^1) - \nabla J(x^0) , - \nabla J(x^0) > = || \nabla J(x^0) ||^2$$

en utilisant le fait que $\nabla J(x^1)$ et $\nabla J(x^0)$ sont des directions orthogonales.
Pour les suivants

$$d^{k-1} = -\nabla J(x^{k-1}) + \beta d^{k-2} \text{ et } \alpha_{k-1} H d^{k-1} = \nabla J(x^k) - \nabla J(x^{k-1})$$

D'où, en utilisant l'orthogonalité : $\nabla J(x^k)$ orthogonal à $\{d^0, ...d^{k-1}\}$

$$\alpha_{k-1} \left\langle d^{k-1}, H d^{k-1} \right\rangle = \left\langle \nabla J(x^k) - \nabla J(x^{k-1}), -\nabla J(x^{k-1}) + \beta d^{k-2} \right\rangle = \left\langle \nabla J(x^{k-1}), \nabla J(x^{k-1}) \right\rangle$$

donc on a autant

$$\beta = \frac{|| \nabla J(x^1) ||^2}{|| \nabla J(x^0) ||^2}$$

que $\beta = \dfrac{< \nabla J(x^1) - \nabla J(x^0) , \nabla J(x^1) >}{|| \nabla J(x^0) ||^2}$.

ce qui conduit aux deux généralisations suivantes.

Algorithme de Fletcher-Reeves (1964)

Etape 0. *On détermine un point d'initialisation x^0, puis on choisit $d^0 := - \nabla J(x^0)$.*

Etape k. *On se déplace au point $x^{k+1} := x^k - \alpha_k d^k$ où le pas α_k est déterminé par une recherche linéaire.*

On calcule ensuite la nouvelle direction de recherche d^{k+1} selon

$$d^{k+1} := - \nabla J(x^{k+1}) + \beta_k d^k \text{ où } \beta_k = \frac{|| \nabla J(x^{k+1}) ||^2}{|| \nabla J(x^k) ||^2}$$

Algorithme de Polak-Ribière (1971)

Etape 0. *On détermine un point d'initialisation x^0, puis on choisit $d^0 := - \nabla J(x^0)$.*

Etape k. *On se déplace au point $x^{k+1} := x^k - \alpha_k d^k$ où le pas α_k est déterminé par une recherche linéaire.*

On calcule ensuite la nouvelle direction de recherche d^{k+1} selon

$$d^{k+1} := - \nabla J(x^{k+1}) + \beta_k d^k \text{ où } \beta_k = \frac{< \nabla J(x^{k+1}) - \nabla J(x^k), \nabla J(x^{k+1}) >}{|| \nabla J(x^k) ||^2}).$$

5.3.3 Gradient Conjugué pré-conditionné

Dans le cas quadratique : $J(x) = \frac{1}{2} x' Hx + bx + c$, H symétrique définie positive

la méthode du gradient conjugué "converge" en un nombre d'itérations égal aux nombre de valeurs propres distinctes de H. Le cas idéal serait donc que toutes les valeurs propres de H soient égales....

Le préconditionnement consiste à faire un changement de métrique ramenant l'ellipsoïde à une surface la plus voisine possible d'une sphère.

L'idée est de faire un changement de variable $x=W^{-1/2} z$ ramenant l'ellipsoïde à une surface la plus proche possible d'une sphère:.

$$J(z)=\frac{1}{2}z'W^{-\frac{1}{2}}HW^{-\frac{1}{2}}z+b'W^{-\frac{1}{2}}z+c$$

$$W^{-1}Hu=\lambda u \Leftrightarrow W^{-\frac{1}{2}}Hu=\lambda W^{+\frac{1}{2}}u$$

La nouvelle matrice a les mêmes valeurs propres que $W^{-1}H$. $v=W^{-\frac{1}{2}}u$

$$W^{-\frac{1}{2}}HW^{+\frac{1}{2}}v=\lambda v$$

Il faut donc que W soit le plus proche possible de H .
Bien sûr, on pourrait choisir W=H, et le problème reviendrait à inverser H, ce qui n'est pas très intéressant au niveau coût de calcul....

On choisit dans la pratique des matrices "proches" de H:

- $W_{ij}=H_{ii}\, d_{ij}$ Matrice de la diagonale de H
- $W_{ij}=H_{ii}$ si $|i-j|<3$ (3 diagonales de H)

-W = factorisée de Cholesky incomplète de H (pré-conditionnement ssor)

$$W=LL^T$$

$$L=\sqrt{\frac{\omega}{2-\omega}}(\frac{D}{\omega}-E)D^{-1/2}$$

$$H=D-E-E^T$$

D étant la partie diagonale de H, E sa partie triangulaire inférieure stricte.

5.4 Théorème de convergence

En voici une description rapide, il s'agit d'un théorème assez général :

- Un algorithme générant une suite x^k peut être vu comme une application A de R^n dans R^n, avec $x^{k+1}= A(x^k)$.
- on note Ω l'ensemble des points de R^n vérifiant une certaine condition nécessaire d'optimalité (par exemple, ensemble des points stationnaires, ie tels que $\nabla J(x)=0$)
- une fonction de descente pour l'algorithme A est une fonction ***continue*** z telle que

$$\begin{cases} z(x^{k+1}) < z(x^k) \text{ si } x \notin \Omega \\ z(x^{k+1}) <\leq z(x^k) \text{ si } x \notin \Omega \end{cases}$$ par exemple : J est en général une fonction de descente pour A

Théorème :

- Si les points x^k sont dans un compact (dans R^n : ie un fermé borné) : c'est le cas par exemple si $J(x) \to +\infty$ quand $\|x\| \to +\infty$, ou si J est strictement convexe.

- Si il existe une fonction de descente pour A
- Si A est continue sur R^n - Ω et $\forall x \notin \Omega, A(x) \neq \varnothing$

Alors si x est limite d'une sous-suite convergente de x^k (il en existe toujours à cause de la compacité), $x \in \Omega$ (ie : x vérifie la condition d'optimalité)

Remarque : On peut énoncer ce théorème pour la convergence de classes d'algorithmes. Si on veut étudier d'un seul coup une famille d'algorithmes dépendant de paramètres (par exemple : gradient avec recherche d'Armijo, paramétrée par la pente), on a besoin de la notion d'applications multivoques : une famille d'algorithmes associe à un point x^k tout un ensemble de points x^{k+1} suivant les paramètres choisis pour l'algorithme.
-Applications *multivoques* R^n : c'est un application qui à x dans R^n associe un sous ensemble A(x) de R^n.
- Application multivoque fermée : c'est le pendant de ce qu'est la continuité pour les applications ordinaires :

5.5 Méthodes de Newton

5.5.1 Méthode de Newton de base

Nous abordons à présent les méthodes dites quadratiques. La méthode de newton consiste à chaque itération à chercher le minimum de la forme quadratique osculatrice à la courbe J. C'est une application de la méthode de Newton-Raphson pour la recherche du zéro de ∇J

$$Q_k(x) := J(x^k) + < \nabla J(x^k) , x-x^k > + \frac{1}{2} < H_k (x-x_k) , x-x^k >$$

$$\nabla Q_k(x) = H_k (x-x^k) + \nabla J(x^k).$$

Et donc, la méthode de Newton appliquée à la recherche des zéros de $\nabla Q(x)$ donne:

$$x^{k+1} = x^k - Jac(\nabla J(x^k))^{-1} \nabla J(x^k) = x^k - \left(H_J(x^k)\right)^{-1} \nabla J(x^k)$$

Ce qui conduit à la résolution d'un système linéaire: $H(x^{k+1} - x^k) = -\nabla J(x^k)$
on pose généralement $\delta^k = (x^{k+1} - x^k)$.

Attention, si H n'est pas définie positive, la direction δ^k peut ne pas être une direction de descente . Ce ci fait que la méthode de Newton peut ne pas converger, notamment si on part trop loin de la solution (voir méthode de Levenberg et Marquard)

Exercice: monter que si A est une matrice définie positive, alors $d^k = -A^{-1}\nabla J(x^k)$ est une direction de descente.

D'autre part, cette méthode nécessite à chaque étape la résolution d'un système linéaire, couteux pour les problèmes de grande taille (la factorisation est moins coûteuse si la matrice est définie positive - Choleski)

5.5.2 Méthodes de "Quasi-Newton"

Cette famille de méthodes s'apparente à la méthode de la sécante, alternative performante à la méthode de Newton-Raphson, et qui présente l'avantage d'éviter le calcul de la dérivé de la fonction (donc ici de la dérivée du gradient, c'est à dire de la matrice Hessienne).

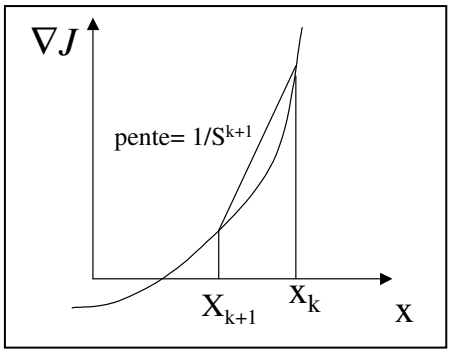
On remplace à chaque itération la matrice H^{-1} par une matrice S . La direction de descente est prise égale à $-S^k \nabla J(x^k)$, et on fait (peut faire) une recherche de minimum mono dimensionnelle dans cette direction , d'ou le paramètre α_k: $x^{k+1} := x^k - \alpha_k S^k \nabla J(x^k)$

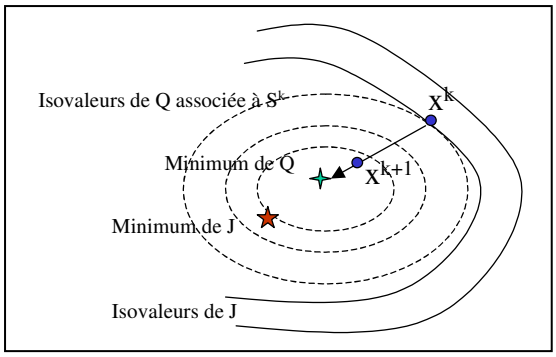
Les matrices successives S^k doivent être simples à calculer. La remise à jour se fait par une formule additive: $S^{k+1} = S^k + C^k$.

C^k sera choisie de façon à ce que $S^{k+1}\left(\nabla J(x^{k+1}) - \nabla J(x^k)\right) = x^{k+1} - x^k$

Que l'on note: $S^{k+1}\gamma^k = \delta^k$

Cette condition appelée condition de quasi-Newton n'est autre que l'expression multidimensionnelle de la formule de la sécante:





L'objectif est de trouver une suite de matrice S^k facile à construire (n'utilisant que le gradient) et qui converge vite vers l'inverse du Hessien, tout en satisfaisant la condition de Quasi-Newton.

Correction de rang 1

Il est naturel de vouloir conserver la symétrie, on peut par exemple chercher S^{k+1} sous la forme:

$S^{k+1} = S^k + v^k (v^k)^T$

La condition de quasi-Newton nous donne v^k : $S^k\gamma^k + v^k\left[(v^k)^T\gamma^k\right] = \delta^k$

$$v^k = \frac{\left(\delta^k - S^k\gamma^k\right)}{\left[\left(v^k\right)^T\gamma^k\right]}$$ d'où en prenant le carré

$$v^k\left(v^k\right)^T = \frac{\left(\delta^k - S^k\gamma^k\right)\left(\delta^k - S^k\gamma^k\right)^T}{\left[\left(v^k\right)^T\gamma^k\right]^2}$$

or : $$\left[\left(v^k\right)^T\gamma^k\right]^2 = \left(\left(v^k\right)^T\gamma^k\right)^T\left(\left(v^k\right)^T\gamma^k\right) = \left(\gamma^k\right)^T v^k\left[\left(v^k\right)^T\gamma^k\right] = \left(\delta^k - S^k\gamma^k\right)^T\gamma^k$$

$$S^{k+1} = S^k + \frac{\left(\delta^k - S^k\gamma^k\right)\left(\delta^k - S^k\gamma^k\right)^T}{\left(\delta^k - S^k\gamma^k\right)^T\gamma^k}$$

Algorithme

- On se donne x^0 et on pose S^0=Id
- à l'itération k, on calcule: $x^{k+1} := x^k - \alpha_k S^k \nabla J(x^k)$ où α_k est obtenu par recherche linéaire
- On met S^k à jour (voir ci-dessus)

D'autres constructions sont plus performantes, notamment celle de Davidon-Fechter Powell qui combine les idées précédentes à celles du gradient conjugué (directions conjuguées)

Algorithme DFP (1971)
Etape 0. *On détermine une matrice d'initialisation S^0 symétrique définie positive et x^0*
Etape k. *La recherche unidimensionnelle se fait selon*

$$d^k := - S^k \nabla J(x^k),$$

et l'on choisit $x^{k+1} := x^k + \alpha_k d^k$.
On remet alors à jour la matrice S^k selon la formule

$$S^{k+1} := S^k + \frac{\delta_k\,\delta_k^T}{<\delta_k, g_k>} - \frac{S^k g^k (g^k)^T S^k}{<S^k g^k, g^k>}$$

$$\text{où } \delta^k := x^{k+1} - x^k =: \textit{vecteur déplacement}$$

$$g^k := \nabla J(x^{k+1}) - \nabla J(x^k) =: \textit{variation du gradient}$$

Commentaires.
1. Retenir que l'on ne cherche pas à "inverser" $H(x^k)$ et que S^k est une "bonne" approximation de l'inverse du hessien $H(x^k)$
2. Dans le cas quadratique défini positif, si $S^0 := I_d$, on a $S^d = H =$ *hessien constant de J*

Et celle de Broyden-Flechter-Goldfarb-Shanno:

Algorithme BFGS (1970)

Etape 0. *On détermine une matrice d'initialisation $\tilde{S}^0$ symétrique défini positive et x^0*
Etape k. *La recherche unidimensionnelle se fait selon*

$$d^k := - \tilde{S}^k \nabla J(x^k),$$

et l'on choisit $x^{k+1} := x^k + \alpha_k d^k$.

On remet alors à jour la matrice $\tilde{S}^k$ selon la formule

$$S^{k+1} := S^k, \tilde{} + [1+\frac{<S^k g_k, \ g_k>}{<\delta_k, g_k>}] \ \frac{dkdkT}{<dk, gk>} \ - \ \frac{\delta^k(g^k)^T S^k + S^k g^k(\delta^k)^T}{<\delta^k, \ g^k>}$$

où $\delta^k := x^{k+1} - x^k =:$ *vecteur déplacement*

$g^k := \nabla J(x^{k+1}) - \nabla J(x^k) =:$ *variation du gradient*

Commentaires.

1. S^k joue le rôle de $H(x^k)^{-1}$ à nouveau...

2. Cette formule est considérée aujourd'hui comme la "meilleure" (en général) et c'est celle qui est "programmée" par défaut sur "*Matlab*" (commande "*fminu*", la commande "*fmins*" utilisant la méthode exploratoire de *Nelder et Mead*).

5.6 Cas des moindres carrés non linéaires

5.6.1 Gauss Newton

Dans le cas fréquemment rencontré où le critère est du type moindres carrés non linéaires:

$J(x)=\frac{1}{2}\sum_{i=1}^{p}\left(f_i(x)\right)^2$, Le gradient et le Hessien prennent des formes particulières de part la structure de J:

$\frac{\partial J}{\partial x_j}(x)=\sum_{i=1}^{p} f_i(x)\frac{\partial f_i}{\partial x_j}$ La j^{ième} composante du gradient de J est le produit scalaire du vecteur f par la j^{ième} colonne de la matrice jacobienne de f, notée f', d'où:

$$\nabla J(x)=\left(J_f(x)\right)^T f(x)$$

Calcul de la matrice Hessienne:

$$\frac{\partial^2 J}{\partial x_j \partial x_k}(x)=\frac{\partial}{\partial x_k}\left(\sum_{i=1}^{n} f_i(x)\frac{\partial f_i}{\partial x_j}\right)=\sum_{i=1}^{n}\frac{\partial f_i}{\partial x_k}\frac{\partial f_i}{\partial x_j}+\sum_{i=1}^{n} f_i(x)\frac{\partial^2 f_i}{\partial x_j \partial x_k}$$

Le premier terme est le produit de la k^{ième} colonne de f' par la j^{ième} colonne de f'.

Le deuxième terme est la somme des produits des f_i par le terme (j,k) de la matrice hessienne de f_i

$$H=\left(J_f(x)\right)^T J_f(x)+Q(x) \quad \text{avec} \qquad Q(x)=\sum_{1}^{p} f_i(x)H_i(x)$$

où H_i est la matrice Hessienne de f_i

La méthode de Gauss Newton consiste à négliger ce deuxième terme.

Si en la solution, J est très petit (nul), alors chaque f_i est très petit et le terme Q(x) s'évanouit.

La méthode de Newton devient alors méthode de Gauss-Newton et s'écrit:

$$\left[J_f{}^T(x^k)J_f(x^k)\right]\delta^k=-\left(J_f(x^k)\right)^T f(x_k)$$

δ_k est dite : "direction de Newton". Noter que ici, la matrice ^{T}JJ étant définie positive, la direction δ est toujours une direction de descente

La méthode de Gauss Newton évite le calcul du Hessien. C'est une méthode d'ordre 1, dont on peu monter que la convergence est quadratique, d'où son succès (simplicité et performance) . De plus, la matrice restante , J^TJ est, sauf cas dégénérés, définie positive. Donc la direction de Gauss-Newton est toujours une direction de descente (mais peut aller "trop loin" dans le sens de la descente, et donc donner un résultat qui ne réalise pas une descente).

5.7 Région de confiance, méthode de Levenberg Marquard

Le fait que la direction de la méthode de Newton ne soit pas systématiquement une direction de descente est un inconvénient, que la méthode de Levenberg-Marquard vise à éliminer, en combinant une direction de gradient et une direction de Newton.
on note Jac(f) : J_f

$$\left[H_J(x^k) + \lambda_k I \right] d^k = -\nabla J(x^k)$$

La direction δ^k peut être vue comme un compromis entre la direction de Newton et celle d'une méthode de "plus profonde descente" . Plus λ_k est grand, plus on se rapproche d'une méthode de gradient. On choisira dans tous les cas λ_k suffisamment grand pour rendre la matrice du premier membre définie positive, de façon à assurer une direction de descente.

En jouant sur le terme λ_k, on peut forcer à la descente même lorsque le terme quadratique négligé dans Gauss-Newton n'est plus négligeable. Il faudra donc diminuer λ_k lorsqu'on se rapproche de la solution (près de la solution, J est convexe)

Cette méthode peut être vue aussi comme une méthode de type "région de confiance". En effet, si on considère le problème contraint:

$\underset{\|\delta\|^2 <= \Delta^2}{Min} \left(\frac{1}{2} \delta' H_J \delta + \delta' \nabla J \right)$ avec Δ, rayon de la région de confiance sur lequel on remplace J par son approximation quadratique

L'annulation du gradient du Lagrangien s'écrit: $(H_J + \lambda I)\delta = -\nabla J$ et $\|\delta\|^2 = \Delta^2$

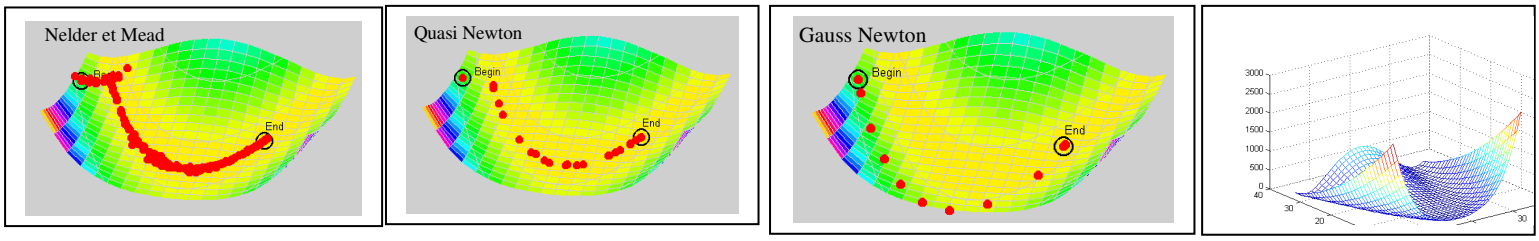
La méthode générale est de résoudre le problème contraint en déterminant λ_k et δ^k, puis de décider si la solution trouvée ^{conduit} à un progrès suffisant; Si non, on recommence le pas en diminuant la taille de la région de confiance.

On peut appliquer la méthode de Levenberg-Marquard a l'idée de Gauss-Newton:

$$\left[\left(J_f(x^k) \right)^T J_f(x^k) + \lambda_k I \right] d^k = -\left(J_f(x^k) \right)^T f_k$$

5.8 Comparaison de différentes méthodes

Sur la fonction "banane" : $z=10(y-x^2)+(x-1)^2$: c'est l'une des démos de la toolbox "optim" de Matlab



Exercice TP : tester sur un cas simple les différents outils de Matlab pour l'optimisation non contrainte

Ci-dessous quelques exemples de petits problèmes pouvant faire l'objet de TDs ou TPs :

6 Optimisation contrainte et multiplicateurs de Lagrange

$Min(J(x)) \quad avec \quad x \in C \subset R^n$

6.1 Contraintes égalités : muliplicateurs de Lagrange

6.1.1 Théorème des multiplicateurs de Lagrange

$C = \{x \in R^n / h_i(x) = 0, i = 1...p\}$ p<n : C est l'intersection de p hyper-surfaces de dimension n-1.

On note h(x) le vecteur $(h_i(x))_{i=1,p}$, h' la matrice jacobienne de h, matrice des vecteurs gradients « en lignes » : $h' = {}^T(\nabla h_i(x))_{i=1,p}$

Point régulier : x est dit « point régulier » de C si h'(x) est de rang maximal, c'est-à-dire si les vecteurs $\{\nabla h_i(x)\}_{i=1,p}$ forment une famille libre.

Si x est régulier, Ker(h') est l'espace tangent à C en x.

Si tous ses points sont réguliers, C est dite variété régulière.

Théorème : Si x^* est solution du problème d'optimisation, et si x^* est un point régulier de C, alors

$\exists(\lambda_i)_{i=1,p} \in R^p$ tels que $\quad \nabla J(x^*) + \sum_1^p \lambda_i \nabla h_i(x^*) = \nabla J(x^*) + {}^T h'.\lambda = 0$

(h'= matrice jacobienne de h)
Les coefficients λ sont appelés multiplicateurs de Lagrange, ou variables duales.

Lagrangien : $L(x,\lambda) = J(x^*) + \sum_1^p \lambda_i h_i(x) = J(x) + {}^T\lambda.h$, $\quad (x,\lambda) \in R^n \times R^p$

On peut remarquer que $\nabla_x L(x,\lambda) = 0$ traduit la condition nécessaire d'optimalité du théorème précédent, et que $\nabla_\lambda L(x,\lambda) = h(x) = 0$ traduit la condition de vérification des contraintes.
En la solution , (x^*,λ^*), le gradient du Lagrangien dans R^n x R^p s'annule. Attention, ce point ne correspond pas à un minimum du Lagrangien, mais à un point selle (voir la suite)

6.1.2 Interprétation

Les multiplicateurs de Lagrange s'interprètent comme le taux de variation du coût $J(x^*)$ (dérivée) si on fait varier les contraintes (perturbation).

Fonction de perturbation

$\Phi(y) = \min J(x), h(x) = y$, y « petit », est la fonction de perturbation, de R^p dans R.

Théorème : Φ est dérivable sur R^m et sa dérivée (son gradient) en y=0 vaut $-\lambda^*$ (solution duale du problème non perturbé)

Notons $(x^*(y), \lambda^*(y))$ la solution pour y donné, et (x^*, λ^*) la solution pour y=0
au premier ordre on a :

$$\Phi(y) - \Phi(0) = J(x^*(y)) - J(x^*) \approx \nabla J(x^*)^T \left(x^*(y) - x^*\right) \approx -\lambda^{*T} h'(x^*)\left(x^*(y) - x^*\right)$$
$$\approx \lambda^{*T}\left(h\left(x^*(y)\right) - h\left(x^*\right)\right) \approx -\lambda^{*T} y = -\Phi'(y)^T y$$

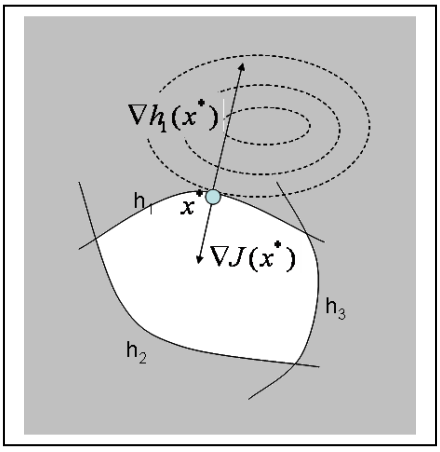
6.2 Contraintes inégalités : conditions de Kuhn et Tucker

$C = \left\{x \in R^n / h_i(x) \le 0, i = 1....p\right\}$ (la contrainte inégalité est souvent notée g_i plutôt que h)

Théorème : si x^* est solution, il existe p réels **positifs** λ (on notera : λ>0) tels que

$$\begin{cases} \nabla J(x^*) + \sum_1^p \lambda_i \nabla h_i(x^*) = 0 \\ \lambda_i h_i(x^*) = 0, \forall i = 1...p \end{cases}$$

La deuxième égalité traduit que soit la contrainte est active, soit le multiplicateur l est nul .
On définit le lagrangien L(x,λ) de la même façon que précédemment.



Sur le dessin ci-contre :

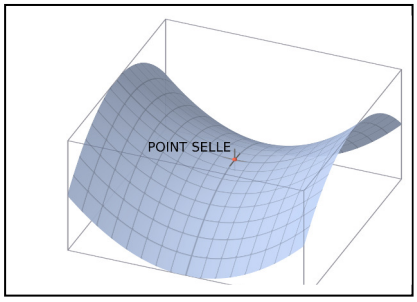
- $h \le 0$ correspond à la zone blanche
- si h_i est active en x^*, ∇h est nécessairement dirigé vers l'extérieur de C (dessin), alors que ∇J est nécessairement dirigé vers l'intérieur. D'où le signe des multiplicateurs de Lagrange dans le cas ce contraintes inégalités (à cause de la convention $h \le 0$)

6.3 Lagrangien et point selle, formulation duale

6.3.1 Point selle

Si L est une fonction de $(x,\lambda) \in R^n \times R^p \to R$, (x^*,λ^*) est un point selle de L si

- x^* est un minimum pour la fonction partielle $x \to L(x,\lambda^*)$
- λ^* est un maximum pour la fonction partielle $\lambda \to L(x^*,\lambda)$



$$\underset{\lambda \in R^p}{Sup}\, L(x^*,\lambda) = L(x^*,\lambda^*) = \underset{x \in R^n}{Inf}\, L(x,\lambda^*)$$

Théorème du point selle :

Si (x^*,λ^*) est un point selle de L, alors

$$\underset{\lambda \in R^p}{Sup}\ \underset{x \in R^n}{Inf}\, L(x,\lambda) = L(x^*,\lambda^*) = \underset{x \in R^n}{Inf}\ \underset{\lambda \in R^p}{Sup}\, L(x,\lambda)$$

Note : si L n'admet pas de point selle, $\underset{x \in R^n}{Inf}\ \underset{\lambda \in R^p}{Sup}\, L(x,\lambda) - \underset{\lambda \in R^p}{Sup}\ \underset{x \in R^n}{Inf}\, L(x,\lambda) > 0$ (« saut de dualité »)

6.3.2 Minimum contraint et point selle

Théorème
Si (x^*,λ^*) est un point selle de $L(x,\lambda) = J(x) + {}^T\lambda.h$ sur $R^n \times (R^+)^p$,
alors
1) x^* est dans C
2) x^* est solution de $\underset{x \in C}{Min}\, J(x)$

Remarque : le Lagrangien est affine en λ. (ie : ne pas se représenter J comme une selle de cheval, qui elle est strictement concave dans une direction)
Remarque : La recherche du point selle se fait pour x^* dans R^n tout entier

Soit C défini par $x \in C \Leftrightarrow h(x) \le 0$ et (x^*,λ^*) un point selle de L sur $R^n \times (R^+)^m$,

Pour montrer le théorème, on utilise tout à tour les deux inégalités du point selle :

$\forall \lambda \geq 0, L(x^*, \lambda) \leq L(x^*, \lambda^*)$ c'est-à-dire : $\lambda^T h(x^*) \leq \lambda^{*T} h(x^*)$

Ce qui s'écrit aussi $\forall \lambda \geq 0, \sum_1^n {}^T (\lambda_i - \lambda_i^*) h_i(x^*) \leq 0$

En prenant $\forall k \neq i, \lambda_k = \lambda_k^*$ et $\lambda_i = 2\lambda_i^*$ on obtient, compte tenu de la positivité de λ :
$\forall i, h_i(x^*) \leq 0$ Ce qui établit bien que x^* est dans C

En prenant $\lambda == 2\lambda^*$, il vient $\lambda^{*T} h(x^*) \leq 0$: En prenant $\lambda=0$, il vient : $\lambda^{*T} h(x^*) \geq 0$ d'où :
$\lambda^{*T} h(x^*) = 0$, résultat utilisé plus bas.
Au passage : $(x^*, \lambda^*) \in C \times (R^+)^p$ λ>=0 et $h(x^*) \leq 0$, une somme de terme négatifs ou nuls ne pouvant être nulle que si chaque terme est nul, on a de plus : $\forall i, \lambda_i^* h_i(x^*) = 0$ (si la contrainte n'est pas active, le multiplicateur correspondant est nul)

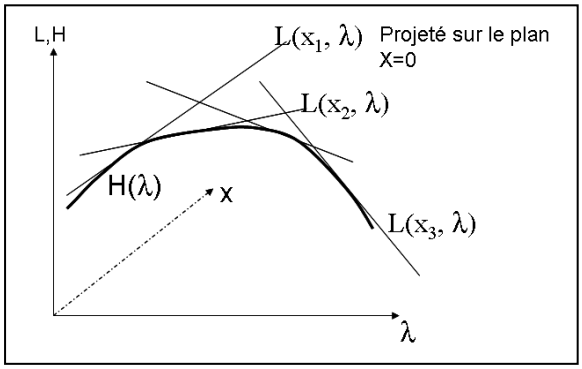
L'autre inégalité s'écrit : $\forall x, L(x, \lambda^*) \geq L(x^*, \lambda^*)$ ie $\forall x, J(x) + \lambda^{*T} h(x) \geq J(x^*) + \lambda^{*T} h(x^*)$
Comme $\lambda^{*T} h(x^*) = 0$ (vu plus haut)
On a bien : $\forall x \in C, J(x) \geq J(x^*)$ car comme $x \in C \Rightarrow \lambda^{*T} h(x) \leq 0$ et donc $J(x) \geq J(x^*)$, ce qui établit que x^* est bien le minimum contraint.

6.3.1 Fonction duale

On peut définir deux fonctions
$G(x) = \underset{\lambda}{Sup}\, L(x, \lambda)$ et $H(\lambda) = \underset{x}{Inf}\, L(x, \lambda)$
H est appelée fonction duale et joue un rôle important dans les algorithmes de recherche de point selle
H est l'enveloppe inférieure d'une famille de fonctions affines en λ : elle est concave.

Dans le cas où $L(x, \lambda) = J(x) + {}^T \lambda.h$
L(x,λ) est une surface « réglée », ie : à x donné, c'est une fonction affine en λ (droite sur le dessin).

On a projeté chaque fonction affine L(x,λ) sur le plan (x=0 ici) , pour faire apparaître H(λ) comme l'enveloppe concave inférieure du faisceau de droites



Exercice sur les contraintes égalités :

On considère l'ellipsoïde E de R^n d'équation x'Ax=1 (A symétrique, définie positive, x vecteur de R^n), et un plan P d'équation: b'x=c, c>0, ne coupant pas l'ellipsoïde. On veut trouver le point de l'ellipsoïde

le plus proche du plan (Le problème de minimisation de forme quadratique avec contraintes linéaires, qui semble purement géométrique à priori, peut se rencontrer dans des situations bien concrètes) .

1- Montrer que le problème revient à minimiser -b'x avec la contrainte x'Ax=1 (dessin en dimension 2)
2- Quel est le lagrangien du problème et quel est son gradient ?
3- Quel est le point x^* de R^3 solution du problème, en fonction de A, b et c ?
4- Et si on cherchait le point du plan le plus proche de l'ellipsoïde?

Corrigé :

1- La distance a P d'un point *du même côté que O* s'écrit: $d(x) = \left(c - \langle b, x \rangle\right) / \|b\|$ (évident par projection sur b normé, faire un dessin). Minimiser d(x) revient donc à minimiser -b'x
On peut utiliser les multiplicateurs de Lagrange :

2- $\boxed{L(x,\lambda) = -b'x + \lambda(x'Ax - 1)}$ $\boxed{\nabla L(x,\lambda) = \begin{cases} 2\lambda Ax - b \\ x'Ax - 1 \end{cases}}$

3- Le minimum contraint annule le gradient du lagrangien

$Ax = b/2\lambda$
$x'Ax = 1$

(Les « courbes de niveau » de f(x)=bx sont des hyperplans parallèles. Les extremums de x'Ax sont atteints aux 2 points de tangence de ces hyperplans et de l'ellipsoïde. C'est à dire aux points de l'ellipsoïde où la normale est colinéaire à b. Il faut faire le dessin dans R^2)
la normale extérieure étant Ax, on retrouve bien ce qui est plus haut)

On résout en x (A est inversible):
$x = A^{-1}b / 2\lambda$
et en reportant dans x'Ax=1 :
(A est symétrique définie positive, A^{-1} aussi - représenter A dans une base propre)
$bA^{-1}b = 4\lambda^2$

D'où 2 valeurs pour λ, l'une correspondant au maximum (l<0 donne le point de l'ellipsoïde le plus loin du plan, diamétralement opposé à la solution cherchée) et l'autre au minimum (>0)

$$\lambda = \frac{\sqrt{bA^{-1}b}}{2} \qquad \text{d'où} \quad x = \frac{A^{-1}b}{\sqrt{bA^{-1}b}}$$

7 Méthodes numériques d'optimisation contrainte

7.1 Méthodes admissibles

Elles consistent, d'un pas à l'autre, à forcer les itérés à rester dans C.
Elles sont surtout efficaces dans le cas où C est affine, où au moins à frontières affines, car il est alors possible, à partir d'un itéré x^k, de trouver un segment issu de x^k et inclus dans C (impossible si la frontière est courbe – sauf à l'approximer par le plan tangent : voir méthodes SQP). La plus connue et pratiquée est celle du gradient projeté.

Pour mémoire :

- Méthodes des directions admissibles : $x^{k+1} = x^k + \alpha^k d^k$, d^k direction admissible, et α^k minimisant $J(x^k + \alpha^k d^k)$
- Méthodes des contraintes actives : on cherche à chaque étape quelles sont les contraintes actives (celles pour lesquelles $h_i(x^k)=0$) et on fait une minimisation avec contraintes égalités sur ces contraintes.

7.1.1 Méthode du gradient projeté

Cette méthode est souvent utilisée dans le cas de contraintes à bord affines. En particulier, pour la méthode d'Uzawa (voir plus bas), quad $C = (R^+)^n$

Principe : si x^k est un itéré admissible, on calcule $-\nabla J(x^k)$, que l'on projette sur l'espace des contraintes actives, ce qui donne une direction de descente d^k. Puis on fait une recherche monodimensionnelle dans la direction d^k, en vérifiant qu'on ne sort pas de la contrainte active en x^k (sinon, on s'arrête « au bord »). Cette dernière étape est souvent difficile à mettre en œuvre (tests).

La projection du gradient sur C est en elle-même un problème d'optimisation, que l'on peut traiter de façon analytique si C est « simple »

Exercice : C est un espace affine, calculer le projecteur en utilisant les multiplicateurs de Lagrange.

7.2 Pénalisations

Elles consistent à remplacer J par une fonction dont l'optimum est proche, voire le même, que celui de J, mais dont les valeurs augmentent de façon très forte quand les contraintes sont violées, et qui est le « plus nulle » possible sur l'espace C. Puis on minimise cette fonction sans contraintes.

7.2.1 Pénalisation extérieure

Soit $\psi(x), x \in R^n$ telle que $\begin{cases} \psi(x) = 0 \text{ si } x \in C \\ \psi(x) \geq 0 \text{ si } x \notin C \\ \psi(x) \to +\infty \text{ quand } distance(x, C) \to +\infty \end{cases}$

On résout le problème sans contraintes : $Min\ J(x) + a\psi(x)$, a>0

Pour des raisons principalement numériques, on résout en pratique une suite de problèmes, en prenant une suite a^k croissante (pénalisation de plus en plus sévère)

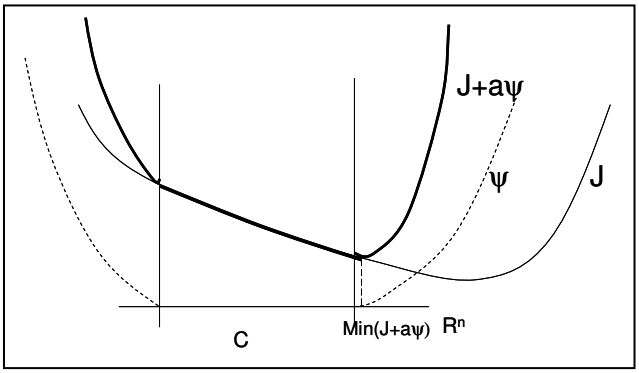
Exemple : si les contraintes s'explicitent sous la forme $h(x) = 0, h \in R^p$, on résout la suite de problèmes :

$Min\ \left(J(x) + a^k \|h(x)\|^2 \right)$

Dans le cas de contraintes inégalités :

$$Min \left(J(x) + a^k \sum_{i=1,p} max(h_i(x),0)^2 \right)$$

7.2.2 Pénalisations intérieure



On pénalise les solutions admissibles qui s'approchent de la frontière de C (envisageable dans le cas de contraintes inégalités)

$$Min \left(J(x) - a^k \sum_{i=1,p} \frac{1}{h_i(x)} \right)$$

7.3 Méthodes primales-duales

Ces méthodes utilisent la dualité et cherchent le point selle du Lagrangien.

7.3.1 Méthode de Newton appliquée au Lagrangien

On applique la méthode de recherche de zéro de Newton en dimension n+p pour le gradient du Lagrangien : système de n+p équations à priori non linéaire
Exercice: formaliser cette méthode.

7.3.2 Algorithme d'Uzawa

C'est un algorithme de gradient à pas constant appliqué à la fonction duale $H(\lambda) = \underset{x}{Inf}\ L(x,\lambda)$ avec la contrainte sur λ : $\lambda >= 0$.

On résout alors une suite de problèmes non contraints en x, et contraints de façon simple sur λ.

Le gradient de la fonction H est facile à calculer à partir de la connaissance des contraintes:
On note $x(\lambda) = arg\ \underset{x}{Inf}\ L(x,\lambda)$

Théorème : sous dés conditions de régularité suffisantes, $\nabla H(\lambda) = h(x(\lambda))$

Exercice : montrez le dans le cas de contraintes égalités où h(x)=Dx-c, D matrice (p,n), c dans R^p

Déduisez le également dans le cas général de la représentation de H comme enveloppe concave de la famille d'hyperplan (fig du paragraphe 6.3.2) ($\nabla H(\lambda) = \nabla_\lambda L(x(\lambda),\lambda)$ …à x(λ) fixé …)

A λ donné, il faut donc résoudre un problème d'optimisation sans contraintes en x pour avoir le gradient, et donc une direction de descente.

Algorithme d'Uzawa : à chaque pas, minimisation en x, puis un pas de gradient projeté (pas constant) sur λ.

- On choisit un paramètre a>0, (profondeur de « montée » pour le pas constant)
- On initialise λ^0, première valeur de λ
- On calcule $H(\lambda^0)$ (pb d'optimisation non contraint en x)
- On prend $\lambda^{k+1}=\lambda^k+a\nabla H(\lambda^k)$
- Si les contraintes sont inégalités, **On projette λ^{k+1} sur $(R^+)^p$**

Remarque : la projection sur R^{+p} est particulièrement simple à effectuer : il suffit d'annuler les coordonnées de λ négatives (*exercice*)

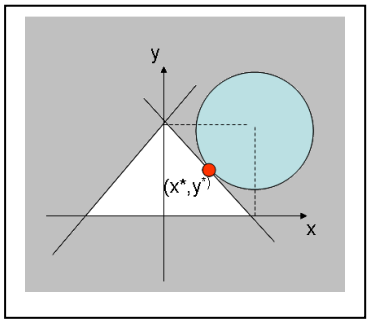
Variante : Arrow-Hurwicz ne met en œuvre qu'un pas dans la minimisation en x

- On choisit deux paramètres a>0 et b>0 pour les pas de gradient (gradient à pas constant)
- On part de deux vecteurs initiaux x^0 et λ^0
- On calcule $x^{k+1}=x^k- b\nabla_x L(x^k,\lambda^k)$ $\left(\nabla_x L(x^k,\lambda^k) = \nabla J(x^k) + {}^T h'(x^k)\lambda^k\right)$: descente à pas constant
- On calcule $\lambda^{k+1} = \lambda^k + bh(x^{k+1})$: montée à pas constant.
- Si les contraintes sont inégalités, On projette λ^{k+1} sur $(R^+)^p$

Exemple sur un cas simple

Voyons comment l'algorithme d'Uzawa/ Arrow-Hurwicz fonctionne, et en particulier comment il « détermine automatiquement » quelles sont les contraintes actives et inactives :

$$J(x,y) = \frac{1}{2}\left((x-c_1)^2 + (y-c_2)^2\right)$$



Les isovaleurs sont donc des cercles de centre C =(c_1,c_2).

Contraintes : intérieur d'un triangle isocèle :

$$\begin{cases} h_1(x,y) = -y \le 0 \\ h_2(x,y) = x + y - 1 \le 0 \\ h_3(x,y) = -x + y - 1 \le 0 \end{cases}$$

$$L(x,y,\lambda_1,\lambda_2,\lambda_3) = J(x,y) - \lambda_1 y + \lambda_2 (x+y-1) + \lambda_3 (-x+y-1)$$

On peut aussi écrire globalement :

$u = (x,y)$

$J(u) = \frac{1}{2}\,{}^T(u-C)\,A\,(u-C)$ avec A = Id (on pourrait prendre une autre matrice sym def>0)

et les contraintes : $h(u) = Du + E \le 0$ avec $D = \begin{pmatrix} 0 & -1 \\ 1 & 1 \\ -1 & 1 \end{pmatrix}$ et $E = \begin{pmatrix} 0 \\ -1 \\ -1 \end{pmatrix}$ alors la matrice jacobienne de h est $h' = D$ et on a $\nabla_x \left({}^T\lambda h\right) = {}^T D\lambda$

$L(u,\lambda) = J(u) + {}^T\lambda h$

$\nabla_u L(u,\lambda) = \nabla J(u) + {}^T h'\lambda = A(u-C) + {}^T D\lambda = 0$

$u(\lambda) = \left(C - A^{-1}\,{}^T D\lambda\right)$

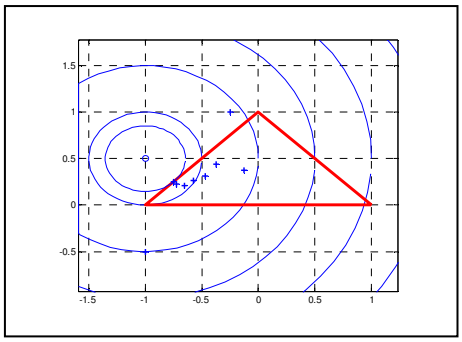
$H(\lambda) = \underset{u}{Min} L(u,\lambda) = L(u(\lambda),\lambda)$ et $\nabla_\lambda H(\lambda) = h(u(\lambda)) = Du(\lambda) + E$

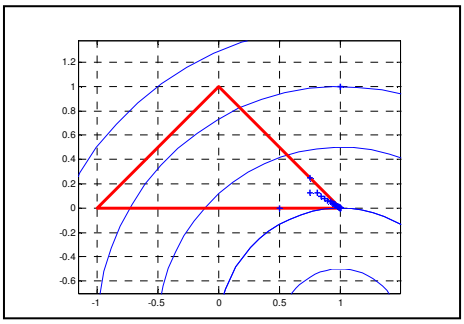
Programme en Matlab réalisant l'algo d'A.H. :

```
% critère: 1/2(x-c)A(x-c)
C=[-1;0.5];
A=eye(2);

%contraintes: h(x)=Bx+C<=0
D=[1 1 ; -1 1; 0 -1];
E=[-1;-1;0];
eps=1e-4;

rho=0.5 % gradient à pas cst
arret=0;
Lam0=[1 1 1]'; % variable duale de départ
while arret==0
    IA=inv(A);
    xl=-IA*D'*Lam0+C;
    gradH=D*xl+E;
    Lam=Lam0+rho*gradH; % pas de montée sur H(λ), gradient à pas constant
    Lam=max(Lam,zeros(3,1)) % projection sur (R+)^3
    Conv=norm(Lam-Lam0);
    if Conv<eps             % critère de convergence
        arret=1;
    else
        Lam0=Lam;
    end
    plot(xl(1),xl(2),'+','linewidth',2)
    hold on
    pause(0.1);
end
```

fig1 : convergence vers (-0.75, 0.25)

fig 2 : convergence vers (1,0)


1.5
1
0.5
0
-0.5
-1.5
-1
-0.5
0
0.5
1


1.2
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-1
-0.5
0
0.5
1

TD/TP : Les Machines à Vecteurs Supports (SVM) pour la classification et la régression.

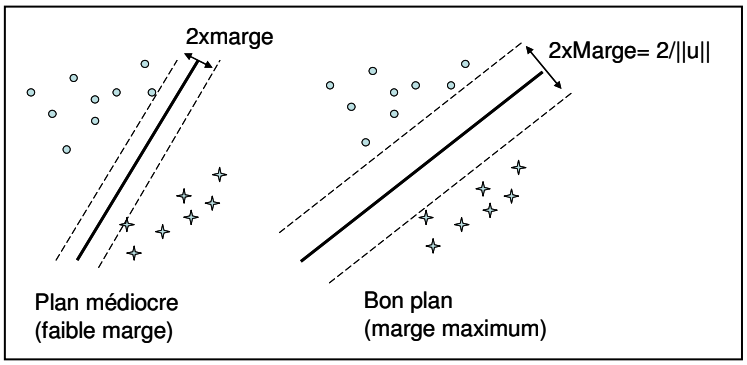
Un document plus précis expose le principe des SVM, et décrit le travail à réaliser. Nous donnons ici les grandes lignes :

Classification par SVM :

Il s'agit du problème de base des « machines à vecteur support», ou « séparateurs à vaste marge » (*Support Vector Machine*, SVM), utilisés notamment en intelligence artificielle.
On considère un ensemble de p points (x_k) dans de R^n . Ces points se divisent en deux familles, leur appartenance à l'une où l'autre des familles est marquée par un label l_k attaché à chaque point, qui vaut 1 pour l'une des familles et -1 pour l'autre.

On cherche à séparer au mieux les deux familles par un hyperplan $^T wx+b=0,\ \ w\in R^N, b\in R$.
L'hyperplan optimum devra maximiser la *marge*, c'est-à-dire la demi-largeur de la bande symétrique de largeur maximale autour de l'hyperplan (voir dessin) qui ne contient aucun point

Cette séparation « linéaire » peut se généraliser grâce à la technique dite du *kernel*-trick pour séparer par des surfaces courbes des familles non séparables linéairement.



L'objectif du TP est de développer un code de classification par SVM et de le tester sur des jeux de données, d'évaluer l'influence de différents paramètres de la méthode.

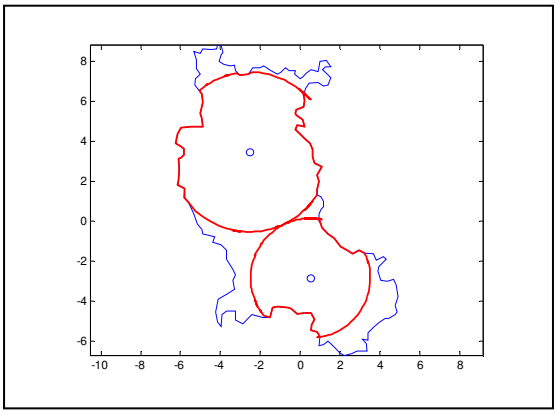
Régression par SVM : les SVR

Le même outil, un peu modifié, peut servir à réaliser une régression sur un ensemble de couples (x_i,y_i), de R^nxR, pour déterminer la fonction y(x) qui *approche au mieux* les points.

Dans le cas « linéaire », on cherche l'hyperplan de R^nxR le « plus proche des points » au sens ou on veut qu'un maximum de points soit à l'intérieur d'une marge données centrée sur l'hyperplan (on voit la similitude avec le problème précédent)

Il s'agira de développer un code de SVR et de le tester, au départ pour p=1 ou 2. Puis de l'utiliser sur le problème des antennes de la Loire décrit ci-dessous.

Application : Positionnement d'antennes d'une radio locale dans la Loire



Une radio locale, qui émet sur un rayon donné autour de son centre d'émission, se demande où placer ses p antennes pour couvrir une surface maximale du département de la Loire.

Le pourtour du département a été numérisé et est donné par une suite de points consécutifs (x_i,y_i), stockée dans un fichier (loire.mat) . Puis il a été réalisé un programme qui, en fonction de la position des centre C_i et des rayons d'émission R_k , calcule la surface (approchée) de l'intersection entre les disques et le département.

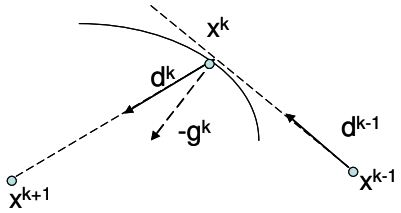
Il s'agit d'un problème d'optimisation dans un espace de dimension 2p (position des centres). Le logiciel étant lourd et couteux en temps de calcul quand la dimension augmente, on se propose de mettre en œuvre des méthodes d'optimisation par surface de remplacement (surrogate optimisation).

La surface de remplacement est obtenue par SVR sur la base d'un ensemble de points d'apprentissage (un nombre restreint de runs du simulateur pour certaines position des centres des antennes). On commencera par une seule antenne mobile (il pourrait y en avoir d'autres fixées, mais une seule sera variable dans un prien temps, pour que le problème soit à 2 dimensions (p=2) et que l'on puisse facilement visualiser les résultats.

Vous pourrez mettre en œuvre soit une fonction matlab, soit une fonction que vous aurez développée (gradient à pas constant …) pour réaliser cette optimisation.

Annexe

Compléments sur le gradient conjugué



J est une forme quadratique : $J(x) = \frac{1}{2}\,{}^{T}xHx + {}^{T}bx + c$,
avec A définie positive, symétrique (donc A définit un produit scalaire)

Les lignes de niveau de J sont des hyper-ellipsoïdes
$J(x) \le a$, intérieur d'un ellipsoïde, est un convexe borné

x^k est la suite de R^n définie par la méthode du gradient conjugué.

On note : $g^k = \nabla J(x^k) = Hx^k + b$,
$x^{k+1} = x^k + \alpha_k d^k$, avec $\alpha_k = \arg\min_{\alpha}(J(x^k + \alpha d^k))$
$d^{k+1} = -\nabla J(x^k) + \beta_k d^k$, avec β_k tel que $\left\langle d^{k+1}, Ad^k \right\rangle = 0$

Proposition 1 :

Si les directions $\left\{d^i\right\}_{i=0..k}$ sont des directions deux à deux conjuguées, alors x^{k+1} est le minimiseur de J sur le sous espace $x^0 + \left\{d^i\right\}_{i=0..k}$

Preuve : pour cela, il suffit de montrer que $\nabla J(x^{k+1})$ est orthogonal à $\left\{d^i\right\}_{i=0..k}$

$x^{k+1} = x^0 + \sum_{1}^{k} \alpha_i d^i = x^p + \sum_{p}^{k} \alpha_i d^i$ (on peut partir de n'importe quel x^p)

donc , si on prend

$\nabla J(x^{k+1}) = Hx^{k+1} + b = Hx^p + b + \sum_{p}^{k} \alpha_i Hd^i = \nabla J(x^p) + \sum_{p}^{k} \alpha_i Hd^i$ avec comme on l'a vu :

$\alpha_i = -\frac{\left\langle g^i, d^i \right\rangle}{\left\| d^i \right\|_H^2}$. les d^i etant deux à deux conjugués, seul le terme de rang p est non nul dans le développement du produit scalaire avec la somme :

$\left\langle \nabla J(x^{k+1}), d^p \right\rangle = \left\langle g^p, d^p \right\rangle - \left\langle g^p, d^p \right\rangle = 0$

Proposition2 :

1 : $\alpha_k = \frac{\left\| g^k \right\|^2}{\left\langle d^k, Hd^k \right\rangle}$

2 : $\beta_k = \dfrac{\left\langle g^{k+1}, g^{k+1} - g^k \right\rangle}{\left\| g^k \right\|^2} = \dfrac{\left\langle g^{k+1}, g^{k+1} \right\rangle}{\left\| g^k \right\|^2}$

Ces deux expressions de β ne font pas intervenir la matrice Hessienne de J, ce qui sera un gros progrès lors de la généralisation au cas de fonctions coût non quadratiques

3 : les directions $\left\{ d^i \right\}_{i=0..n-1}$ sont des directions deux à deux conjuguées.

Preuve :

1) $\alpha_k = -\dfrac{\left\langle g^k, d^k \right\rangle}{\left\langle d^k, Hd^k \right\rangle} = \dfrac{\left\langle g^k, -g^k + \beta_k d^{k-1} \right\rangle}{\left\langle d^k, Hd^k \right\rangle} = \dfrac{\left\| g^k \right\|^2}{\left\langle d^k, Hd^k \right\rangle}$

2) on a vu : $\beta_k = \dfrac{\left\langle g^{k+1}, Hd^k \right\rangle}{\left\| d^k \right\|_H^2}$ or $g^{k+1} - g^k = H\left(x^{k+1} - x^k \right) = \alpha_k Hd^k$ d'où

$\beta_k = \dfrac{\left\langle g^{k+1}, g^{k+1} - g^k \right\rangle}{\alpha_k \left\| d^k \right\|_H^2} = \dfrac{\left\langle g^{k+1}, g^{k+1} - g^k \right\rangle}{\left\langle g^k, g^k \right\rangle}$

3)

Récurence : supposons les d^i deux à deux conjugués pour i =0 …k. Alors, comme vu précédemment, $\nabla J(x^{k+1}) = g^{k+1}$ est orthogonal à $\left\{ d^i \right\}_{i=0..k}$.

d^{k+1} est conjuguée avec d^k par construction, donc c'est vrai pour i=k. Pour les autres valeurs de i:

$\left\langle d^{k+1}, Hd^i \right\rangle = \left\langle -g^{k+1} + \beta_k d^k, Hd^i \right\rangle = \left\langle -g^{k+1}, Hd^i \right\rangle$ car d^k et d^i sont conjugués. Reste à montrer l'orthogonalité de g^{k+1} et Hd^i

Si 0<i<k : $Hd^i = \dfrac{1}{\alpha_i} H\left(x^{i+1} - Hx^i \right) = \dfrac{1}{\alpha_i}\left(g^{i+1} - g^i \right) = \dfrac{1}{\alpha_i}\left(-d^{i+1} + \beta_i d^i + d^i - \beta_{i-1} d^{i-1} \right)$

Donc $Hd^i \in vect\left\{ d^{i-1}, d^i, d^{i+1} \right\}$ qui est orthogonal à g^{k+1}

Si i=0, $Hd^0 = \dfrac{1}{\alpha_i} H\left(x^1 - x^0 \right) = \dfrac{1}{\alpha_i}\left(g^1 - g^0 \right) = \dfrac{1}{\alpha_i}\left(-d^1 + \beta_i d^0 + d^0 \right)$

$Hd^i \in vect\left\{ d^0, d^1 \right\}$ qui est orthogonal à g^{k+1}