Optimisation : Méthodes déterministes et algorithmes numériques

Plan

Partie 1

- Introduction
- Cas de problèmes sans contraintes
- Cas non différentiable

Partie 2

- Cas de problèmes avec contraintes
- Calcul du gradient
- Méthode de diffusion

Introduction.

Les problèmes considérés s'écrivent sous la forme

$$(P)\begin{cases} min\ f(x) & x \in \mathbb{R}^n \\ c_j(x) \leq 0 & j \in I \\ c_j(x) = 0 & j \in E \end{cases}$$

I désigne les contraintes d'inégalité et E désigne les contraintes d'égalité.

On suppose qu'on dispose d'un sous-programme calculant $f(x)$, $\nabla f(x)$ et éventuellement $\nabla^2 f(x)$.

Classification des problèmes.

- Problèmes sans contraintes ($I = E = \emptyset$)
  - problème quadratique ($f(x) = \frac{1}{2}xAx - bx$ avec A symétrique)
  - problème non linéaire (f ni linéaire ni quadratique)
  - problème non différentiable
- Problèmes avec contraintes linéaires
  - uniquement des contraintes d'égalité ($I = \emptyset$)
    - f quadratique
    - f non quadratique
  - $I \neq \emptyset$
    - f linéaire (programmation linéaire)
    - f quadratique (programmation quadratique)
    - f ni linéaire ni quadratique
- Problèmes avec contraintes non linéaires (programmation non linéaire)
  - $I = \emptyset$
  - $I \neq \emptyset$

Problèmes sans contraintes.

Conditions d'optimalité

- **CN 1** : $\nabla f(x^*) = 0$
- **CN 1** : $\nabla^2 f(x^*)$ semi-définie positive
- **CS** : $\nabla f(x^*) = 0$ et $\nabla^2 f(x^*)$ définie positive

les algorithmes cherchent $\overline{x}$ tel que $\nabla f(\overline{x}) = 0$ en construisant une suite minimisante $\{x_k\}$ tel que $liminf \nabla f(x_k) = 0$

schéma général :

- `Etape 0` : choisir x_0
- `Etape 1` : test d'arrêt : $\nabla f(x_k) \leq \epsilon$
- `Etape 2` : calcul d'une direction de descente on cherche d_k minimisant un problème approché du type (P_k) :
  $$\min_{\{d, \|d\| \leq a\}} f(x_k) + (g(x_k), d) \left[+ \frac{1}{2}(M_k^{-1} d, d) \right]$$
- `Etape 3` : calcul d'un pas (Recherche linéaire)
  trouver un pas t_k qui fait décroître f suffisamment en cherchant à minimiser $q(t) = f(x_k + t d_k)$
- `Etape 4` : $x_{k+1} = x_k + t_k d_k$
  $k = k + 1$
  aller en `Etape 1`

Recherche linéaire. (1)

Remarques :

- La recherche linéaire est une boucle exécutée à chaque itération, il est donc essentiel qu'elle soit très rapide. Les recherches linéaires "exactes" sont à éviter.
- la fonction $q(t) = f(x_k + td_k)$ n'est connue que par ses valeurs numériques
- $q'(t) = (\nabla f(x_k + td_k), d_k)$
  en particulier $q'(0) < 0$
- on va chercher $t \in [t_g, t_d]$

Schéma

- Etape 0 : t initial > 0 (initialisation par la boucle extérieure)
  $t_g = 0$ et $t_d = \infty$
- Etape 1 : test de t
  - (a) t *acceptable*. Stop
  - (b) t trop grand, faire $t_d = t$ et aller en Etape 2
  - (c) t trop petit, faire $t_g = t$ et aller en Etape 2
- Etape 2 : choix d'un nouveau t
  - si aucun $t_d \neq \infty$ n'a été trouvé : prendre un t plus grand (*extrapolation*)
  - si $t_d \neq \infty$: prendre $t \in]t_g, t_d[$ (*interpolation*)
- Etape 3 : aller en Etape 1

Recherche linéaire. (2)

- Pour extrapoler on peut prendre $t = a * t$ avec $a > 1$ et pour interpoler on peut prendre $t = \frac{t_g + t_d}{2}$. On peut aussi accélérer en testant la valeur minimisant l'interpolation cubique de $q(t)$ connaissant $q(t)$, $q'(t)$, $q(t_-)$ et $q'(t_-)$.
- Pour tester si t est acceptable ou non on peut utiliser la règle de Wolfe ou la règle de Goldstein et Price (ou la règle d'Armijo dans certains cas).

Règle de Wolfe

a)	$q(t) \leq q(0) + m_1 t q'(0)$	et	$q'(t) \geq m_2 q'(0)$	t acceptable
b)	$q(t) > q(0) + m_1 t q'(0)$			$t_d = t$
c)	$q(t) \leq q(0) + m_1 t q'(0)$	et	$q'(t) < m_2 q'(0)$	$t_g = t$

Règle de Goldstein et Price

a) $m_2 q'(0) \leq \frac{q(t)-q(0)}{t} \leq m_1 q'(0)$

b) $m_1 q'(0) < \frac{q(t)-q(0)}{t}$

c) $\frac{q(t)-q(0)}{t} < m_2 q'(0)$

Règle d'Armijo

a) $\frac{q(t)-q(0)}{t} \leq m_1 q'(0)$

b) $m_1 q'(0) < \frac{q(t)-q(0)}{t}$

c) jamais !

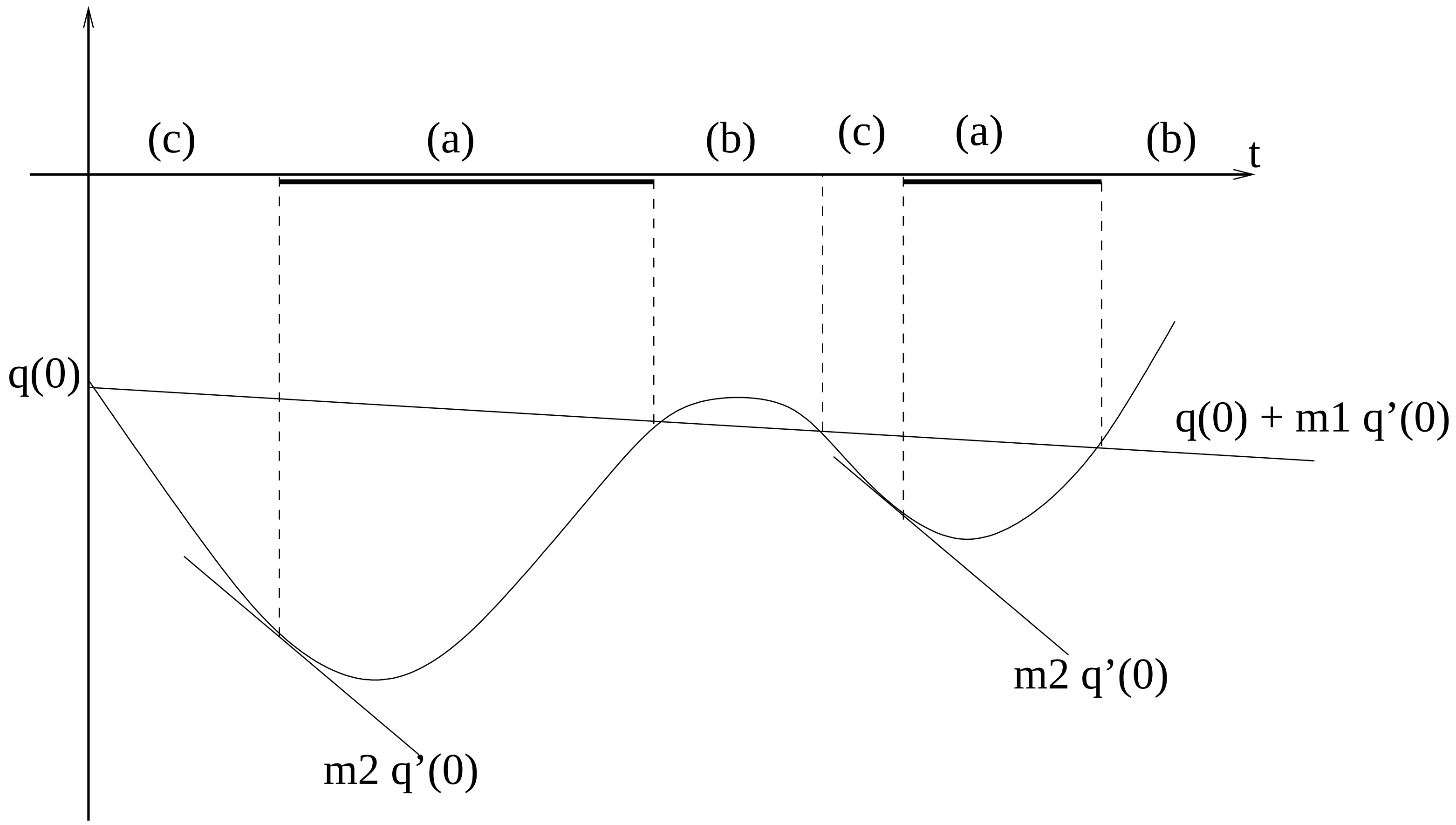


Schéma de la règle de Wolfe

Direction de descente (1)

- $d_k = -\nabla f(x_k)$ (plus forte pente, à éviter !)
- $d_k = -M_k \nabla f(x_k)$
  Newton $M_k = (\nabla^2)^{-1} f(x_k)$
- Newton restauré :

$$d_k = \begin{cases} -(\nabla^2)^{-1} f(x_k) \nabla f(x_k) & \text{si } \nabla^2 f(x_k) \text{ est définie positive} \\ -\nabla f(x_k) & \text{sinon} \end{cases}$$

- Quasi-Newton : approcher le Hessien ou plutôt son inverse en construisant une suite de matrices M^k symétriques, de préférence définie positives et vérifiant $y_k = M^k s_k$.
  $y_k = g_{k+1} - g_k$ et $s_k = x_{k+1} - x_k$.
  On a la relation de moyenne satisfaite par le Hessien : $y_k = \left(\int_0^1 \nabla^2 f(x_k + t s_k) dt \right) s_k$

Direction de descente (2) : gradient conjugué

- Principe : chercher d_{k+1} dans $\{d = \sum_1^{k+1} \alpha_i g_i + \sum_1^k \beta_i d_i\}$
- Dans le cas ou f est quadratique

$$\begin{cases} d_1 = -g_1 \\ d_{k+1} = -g_{k+1} + c_k d_k \text{ avec } c_k = \frac{|g_{k+1}|^2}{|g_k|^2} \end{cases}$$

le pas exact est $t_k = -\frac{(g_k, d_k)}{(Ad_k, d_k)}$
converge en au plus n itérations

- gradient conjugué non linéaire

$$\begin{cases} d_1 = -g_1 \\ d_k = -g_k + c_{k-1} d_{k-1} \text{ et si } (d_k, g_k) \geq 0 \text{ prendre } d_k = g_k \end{cases}$$

$c_{k-1} = \frac{|g_k|^2}{|g_{k-1}|^2}$: Fletcher-Reeves (même choix que pour le cas quadratique)
$c_{k-1} = \frac{(g_k - g_{k-1}, g_k)}{|g_k|^2}$: Polak-Ribière.
Remarque : pour FR on peut montrer la convergence globale de la méthode (avec une recherche linéaire) et il existe un contre exemple pour lequel PR ne converge pas. En pratique PR est beaucoup mieux que FR qui est presque jamais utilisée.

Direction de descente (3) : BFGS

Motivations des méthodes de quasi-Newton

- pour la méthode de Newton il faut calculer $\nabla^2 f(x)$
- f peut ne pas être C^2
- la direction n'est pas nécessairement une direction de descente

Les méthodes de quasi-Newton permettent de :

- ne calculer que le gradient
- donner une direction de descente pour f

Algorithme de BFGS

$$\begin{cases} M_0 = Id \\ M_{k+1} = M_k + \frac{y_k y_k^T}{y_k^T s_k} - \frac{M_k s_k s_k^T M_k}{s_k^T M_k s_k} \end{cases}$$

avec recherche linéaire de Wolfe

Résultats de convergence

Remarque : on parle de convergence de l'algorithme lorsque celui-ci converge vers un minimum local. On dira que la convergence est locale lorsque l'on a convergence pour x_0 assez proche d'un optimum et on dira que la convergence est globale lorsqu'elle se produit $\forall x_0$ (à ne pas confondre avec la recheche d'un minimum global d'une fonction).

- **Méthode de plus forte descente** : si le gradient g est lipschitzien, et si t_k n'est "pas trop petit" alors on a convergence.

- **Gradient Conjugué Non Linéaire** : on a convergence globale pour Fletcher-Reeves et on a des cas de non convergence de Polak-Ribière.

- **Newton** : convergence locale très rapide (quadratique) quand x_0 est proche de l'optimum. Si $\nabla^2 f_k$ est définie positive pour tout k, on peut globaliser la méthode de Newton par une recherche linéaire ou la méthode de région de confiance.

- **BFGS** : si f convexe et g lipschitzien et si M_1 est définie positive alors avec une recherche linéaire de Wolfe on a convergence globale de la méthode.

Méthode de région de confiance (1)

- La méthode permet de globaliser (rendre la convergence globale) les méthodes de descente.
- Elle est utile quand la fonction est assez bruitée.
- Elle est plus difficile à mettre en œuvre mais possède des meilleurs résultats de convergence sous des hypothèses plus faibles.
- La recherche d'une direction de descente et le calcul d'un pas se font en même temps.
- Le principe de la méthode : on approche les variations de f pour un incrément s de x_k par un modèle quadratique de la forme $\psi_k(s) = g_k^T s + \frac{1}{2} s^T M_k s$ où g_k est le gradient et où M_k peut être le Hessien et peut ne pas être définie positive.
  Quand $M_k = \nabla^2 f(x_k)$ on a $f(x_k + s) \simeq f(x_k) + \psi_k(s)$.
- Le sous problème qu'on résout est

$$(RC_k) \begin{cases} \min \psi_k(s) \\ |s| \leq \Delta_k \end{cases}$$

Δ_k est appelé le rayon de confiance et l'ensemble $\{x \in \mathbb{R}^n \,,\, |x| \leq \Delta_k\}$ est appelé région de confiance. La norme $| \cdot |$ est la norme euclidienne.

Méthode de région de confiance (2)

On se donne les constantes suivantes :
- des seuils de succès : $0 < \omega_1 < \omega_2 < 1$,
- des facteurs de mise à jour de Δ_k : $0 < \sigma_1 < \sigma_2 < 1 < \sigma_3$.

Schéma de l'algorithme

- `Etape 1` : Calcul du déplacement s_k
  - `(a)` calcul d'une solution approchée de (RC_k) telle que $\psi_k(s_k) < 0$
    si impossible, stop.
  - `(b)` calcul du ratio $\rho_k = \frac{R_k}{P_k}$
    entre la décroissance réelle de f :
    $R_k = f(x_k) - f(x_k + s_k)$
    et la décroissance prédite par le modèle : $P_k = -\psi_k(s_k)$.
  - `(c)` Si $\rho_k \leq \omega_1$, la concordance entre le modèle ψ_k et f est considérée comme non satisfaisante. Réduire Δ_k en le prenant dans $[\sigma_1\Delta_k, \sigma_2\Delta_k]$ et revenir à `Etape 1 (a)`.
- `Etape 2` : $x_{k+1} = x_k + s_k$
- `Etape 3` : Mise à jour de Δ_k :

$$\Delta_{k+1} \in \begin{cases} [\sigma_2\Delta_k, \Delta_k] \text{ si } \rho_k \leq \omega_2 \\ [\Delta_k, \sigma_3\Delta_k] \text{ sinon} \end{cases}$$

- `Etape 4` : Mise à jour du modèle : calcul de g_{k+1} et M_{k+1}.

Quelques remarques

- Que choisir ?
  - comme c'est le cas en général, *il n'y a pas de règle générale* !
  - le choix dépend du problème à résoudre et surtout de ce que l'on a à notre disposition (∇f, $\nabla^2 f$).
  - on peut dire que l'on utilise la méthode de Newton quand cela est possible, sinon on utilise la méthode BFGS et sinon on utilise la méthode du gradient conjugué non linéaire.
- Dans le cas des gros systèmes on rencontre des problèmes dus à la taille du système.
  - les méthodes de Newton ou quasi-Newton peuvent être très ralenties à cause des opérations matricielles (inversion et/ou multiplication par un vecteur).
  - la méthode de *Newton tronqué* inverse le Hessien d'une façon incomplète.
  - la méthode *BFGS à mémoire limitée* ne stocke pas la matrice M_k mais fait les calculs à partir des vecteurs s_i et y_i pour $i = k, k-1, \cdots, k-m$. On peut avoir des résultats acceptables pour m compris entre 5 et 10, sur des problèmes de grandes taille.

Optimisation non différentiable.

Introduction :

- Pour une fonction f convexe à valeurs finies, on définit la dérivée directionnelle par :

$$f'(x,d) = \lim_{t \mapsto 0^+} \frac{f(x+td) - f(x)}{t},$$

- On définit le sous-différentiel de f en x par :

$$\begin{aligned} \partial f(x) &= \{s \in \mathbb{R}^n, f(y) \geq f(x) + < s, y-x >, \forall y \in \mathbb{R}^n\} \\ &= \{s \in \mathbb{R}^n, < s, d > \leq f'(x,d), \forall d \in \mathbb{R}^n\}. \end{aligned}$$

- Conditions d'optimalité : les propositions suivantes sont équivalentes
  - f est minimisée en $\bar{x}$;
  - $0 \in \partial f(\bar{x})$;
  - $f'(\bar{x}, d) \geq 0$ pour tout $d \in \mathbb{R}^n$.

Le cas non différentiable présente beaucoup de spécificités et de difficultés :

- piège du test d'arrêt : la condition "gradient proche de zéro" peut ne jamais être vérifiée même dans des cas très simple comme la fonction $f(x) = |x|$ ou $|g(x)| = 1$ pour tout x non nul.

- Le gradient est souvent calculé d'une façon approchée, donc du fait de la discontinuité du sous-gradient, une petite variation de la valeur de x_k peut donner de grandes variations du sous-gradient et par suite sur le calcul la direction de descente d_k.

Méthodes de descente

- L'idée est de générer une suite (x_k) qui assure une diminution de la fonction f
- Une des façons pour trouver la direction d_k est de choisir à chaque itération la direction de plus forte pente : d_k solution de $\min_{||d||=1} f'(x_k, d)$.
- Les principales difficultés de ces méthodes sont :
  - l'impossibilité d'implémenter le test d'arrêt
  - le calcul de la direction de descente qui requiert la connaissance de tout le sous-gradient.

Le schéma de l'algorithme est le suivant :

- `Etape 1 :` `test d'arrêt` : si $0 \in \partial f(x_k)$ arrêter.
- `Etape 2 :` `direction de descente` : trouver d_k direction de descente pour f en x_k.
- `Etape 3 :` `recherche linéaire` : trouver un pas $t_k > 0$ tels que pour $x_{k+1} = x_k + t_k d_k$ on ait $f(x_{k+1}) < f(x_k)$.
- `Etape 4 :` `boucle` : changer k en $k+1$ et aller à `Etape 1`.

Méthode de sous-gradient (1)

- Cette méthode est basée sur la simple connaissance d'un **seul élément du sous-gradient** et non plus sur la connaissance du sous-gradient en entier.

- Un algorithme du type sous-gradient s'écrit :
  - `Etape 1 : appel de la "boite noire" et test d'arrêt` : appeler la boite noire avec $x = x_k$. Si $0 \in \partial f(x_k)$ arrêter.
  - `Etape 2 : candidat à la direction de descente` : $d_k = -\frac{s(x_k)}{||s(x_k)||}$.
  - `Etape 3 : recherche linéaire` : trouver un pas $t_k > 0$ tels que pour $x_{k+1} = x_k + t_k d_k$ on ait $f(x_{k+1}) < f(x_k)$.
  - `Etape 4 : boucle` : changer k en $k+1$ et aller à `Etape 1`.

Méthode de sous-gradient (2)

- Le terme "boite noire" désigne le programme de l'utilisateur qui à partir d'une valeur x_k calcule $f(x_k)$ et $s(k_k) \in \partial f(x_k)$.
  Exemple : soit f définie par : $f(x) = \min_{i=1,\ldots,N} f_i(x)$, où les fonctions f_i sont différentiables. Si i_0 est un indice pour lequel f et f_{i_0} coïncident : $f(x) = f_{i_0}(x)$, alors $s(x) = \nabla f_{i_0}(x)$ appartient à $\partial f(x)$.
- Reste encore le problème d'implémentation d'un test d'arrêt.
- La direction d_k peut ne pas être une direction de descente, (la recherche linéaire peut échouer).
  - Les méthodes dites de "série divergente" peuvent résoudre ce problème : (t_k) vérifiant les conditions $t_k \mapsto 0$ et $\sum_k t_k = +\infty$. L'idée est de faire tendre t_k vers zéro pas "trop vite".
  - Ces méthodes sont généralement lentes et peuvent être en pratique piégées dans des oscillations.

Méthode des plans sécants (1)

- L'idée est d'utiliser l'information donnée par la "boite noire" pour construire un modèle de la fonction à minimiser.
- Le *faisceau* des valeurs $f_k = f(x_k)$ et $s_k = s(x_k)$ construit une approximation convexe affine par morceaux :

$$\check{f}_k = max_{i=1,..,k}\{f_i + < s_i, y - x_i >\}$$

- A chaque itération on minimise $\check{f}_k$ sur un convexe C (supposé connu) contenant l'optimum de f.
- Un algorithme simplifié sans recherche linéaire s'écrit :
  - `Etape 1 : appel de la "boite noire" et test d'arrêt` : calculer $\delta_k = f(x_k) - \check{f}_{k-1}(x_k)$,
    si $\delta_k \leq \underline{\delta}$ arrêter.
  - `Etape 2 : candidat à la direction de descente` :
    trouver $\check{x}_{k+1} = argmin_{y \in C} \check{f}_k(y)$,
    $d_k = \check{x}_{k+1} - x_k$.
  - `Etape 3 : recherche linéaire` : pas constant $t_k = 1$, $x_{k+1} = x_k + t_k d_k$.
  - `Etape 4 : boucle` : changer k en $k+1$ et aller à `Etape 1`.

Méthode des plans sécants (2)

- Le test d'arrêt porte sur l'erreur δ_k du modèle $\check{f}_k$ qu'on minimise par rapport à la fonction f
- La méthode n'est pas une méthode de descente pour f car le choix de x_{k+1} minimise $\check{f}_k$ mais pas forcément la fonction f.
- L'algorithme peut parfois converger très lentement à cause de la décroissance δ_k qui peut être très faible.
- Un autre désavantage de cette méthode est l'accumulation des plans ajoutés au modèle avec les itérations.

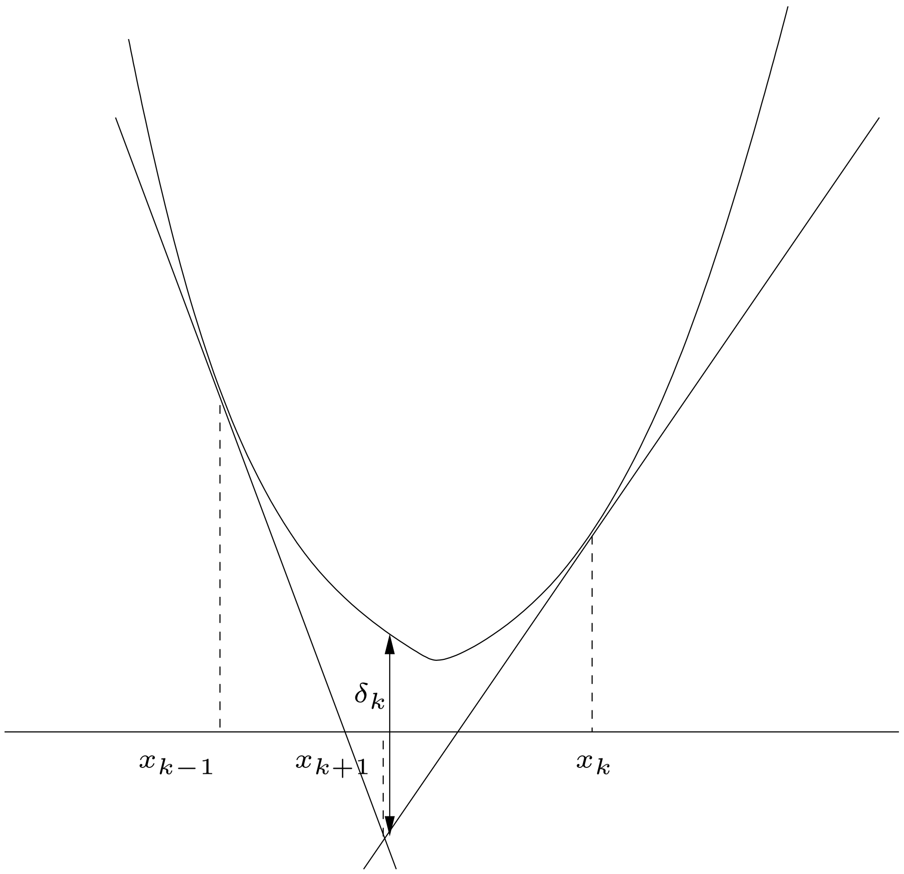


Schéma pour l'algorithme des plans sécants.

Méthode des faisceaux (1)

- C'est une variante stabilisée des méthodes des plans sécants, où la méthode devient une méthode de descente pour la fonction f.
- Principalement à chaque itération on définit une décroissance minimale pour accepter le pas et on réduit l'ensemble C de façon à assurer cette décroissance.
- A chaque étape un modèle simplifié ϕ_k de la fonction f.
- Le schéma général est le suivant :
  - `Etape 0 :` choisir $\underline{\delta} > 0$ et $m \in]0, 1[$. Faire $k = 1$ et $\delta_1 = \infty$.
  - `Etape 1 : test d'arrêt` : si $\delta_k \leq \underline{\delta}$ arrêter.
  - `Etape 2 : candidat à la direction de descente` :
    calculer $y_{k+1} = argmin_{y \in C_k} \phi_k(y)$.
  - `Etape 3 : évaluation du candidat` : définir $\delta_k = \delta_k(\phi_k, x_k, ||_k, y_{k+1})$.
    test de descente :
    $f(x_k) - f(y_{k+1}) \geq m\delta_k$?
    $$\begin{cases} oui & x_{k+1} = y_{k+1} & \text{pas de descente} \\ & \text{déf. } |.|_k & \\ non & x_{k+1} = x_k & \text{pas nul} \end{cases}$$
  - `Etape 4 : amélioration du modèle et boucle` :
    ajouter y_{k+1} au modèle ϕ_{k+1},
    changer k en $k + 1$ et aller à `Etape 1`.

Méthode des faisceaux (2)

- Dans `Etape 2` le modèle ϕ_k coïncide avec le modèle $\check{f}_k$ quand on garde tous les plans.
- Le choix de l'ensemble C_k est assez délicat, certains modèles utilisent l'idée de régions de confiance pour garantir que le nouveau domaine contient l'optimum.
- Dans le cas d'un pas nul (dans `Etape 3`), le nouveau y_{k+1} à l'itération suivante sera différent de celui de l'étape précédente, car on minimise un modèle ϕ_k plus riche.
- \- Des méthodes plus sophistiquées utilisent la compression de faisceaux qui consiste à fixer un nombre maximal de plans pour le modèle ϕ_k.
  \- Une fois ce nombre atteint, on supprime un certain nombre de pièces, l'information ainsi perdue est reportée dans la définition des autres pièces en remplaçant les vecteurs s_k par des vecteurs $\hat{s}_k$ aussi éléments de $\partial f(x_k)$ et définis à partir des vecteurs s_k.

Plan de la deuxième partie.

- Problèmes avec contraintes
  - Conditions d'optimalité.
  - Cas des contraintes d'égalité.
  - Cas des contraintes d'inégalité.
    - Méthode du gradient projeté.
    - Méthode d'activation des contraintes.
    - Approche Min-Max.
    - Programmation Quadratique Successive.
    - Méthodes de points intérieurs.
- Calcul du gradient par différentiation automatique.
- Méthode de diffusion.

Problèmes avec contraintes.

Conditions d'optimalité avec contraintes d'égalités

Le problème :

$$(P_E) \begin{cases} \min f(x) \\ c(x) = 0 \in \mathbb{R}^m \end{cases}$$

Le Lagrangien du problème : $l(x, \lambda) = f(x) + \lambda^T c(x)$.

CN 1 : si $A_* = c'(x_*)$ est surjective, il existe un unique $\lambda_* \in \mathbb{R}^m$ tel que

$$\begin{cases} \nabla_x l(x_*, \lambda_*) = 0 \\ c(x_*) = 0 \end{cases}$$

CN 2 : si $A_* = c'(x_*)$ est surjective, on a

$$\begin{cases} \nabla_x l(x_*, \lambda_*) = 0 \\ c(x_*) = 0 \\ \nabla^2_{xx} l(x_*, \lambda_*) \text{ est définie positive sur } N(A_*). \end{cases}$$

Conditions d'optimalité avec contraintes d'inégalités. (1)

Le problème :

$$(P_{EI}) \begin{cases} \min f(x) \\ c_E(x) = 0 \in \mathbb{R}^{m_E} \\ c_I(x) \leq 0 \in \mathbb{R}^{m_I} \end{cases}$$

Le Lagrangien du problème ($c = (c_E, c_I)$) : $l(x, \lambda) = f(x) + \lambda^T c(x)$.
On note $I^0(x) = \{i \in I, c_i(x) = 0\}$ (ensemble des contraintes actives) et $I^0_* = I^0(x_*)$

Conditions d'optimalité avec contraintes d'inégalités. (1)

CN 1 : conditions de Karush, Kuhn et Tucker (KKT)
si les contraintes sont qualifiées en x_* il existe $\lambda_* \in \mathbb{R}^m$ tel que

$$\begin{cases} \nabla_x l(x_*, \lambda_*) = 0 \\ c_E(x_*) = 0 \\ c_I(x_*) \leq 0 \\ (\lambda_*)_I \geq 0 \\ (\lambda_*)_I^T c_I(x_*) = 0. \end{cases}$$

- *Définition* : on dit que les contraintes de (P_{EI}) sont qualifiées en x si $T_x X = T'_x X$ où $T'_x X = \{d, c'_E(x) \cdot d = 0, c'_{I^0(x)}(x) \cdot d \leq 0\}$ et $T_x X$ est le cône tangent. On dira que le cône tangent s'obtient en linéarisant les contraintes actives. (le cône tangent à X en x est l'ensemble $\{d,\ \exists d_k \mapsto d,\ \exists t_k \mapsto 0^+,\ x + t_k d_k \in X\}$).

Cas des contraintes d'égalités. (1)

- Dans le cas où $c(x)$ est linéaire ($Ax = 0$) on peut ramener le problème à un problème sans contraintes en se restreignant à $N(A)$.
- Approche à ne pas faire (pas nécessairement convergente) : prendre $x_+ = x + d$ où d est solution de

$$\begin{cases} \min_d \nabla f(x)^T d + \frac{1}{2} d^T \nabla^2 f(x) d \\ c(x) + c'(x) d = 0 \end{cases}$$

Cas des contraintes d'égalités. (2)

- On note $A = A(x) = c'(x)$ et $L = L(x, \lambda) = \nabla^2_{xx} l(x, \lambda)$
  L'approche consiste à résoudre le système non linéaire :

$$\left\{ \begin{array}{l} \min_d \nabla f(x)^T d + \frac{1}{2} d^T L(x, \lambda) d \\ c(x) + c'(x) d = 0 \end{array} \right.$$

- On prend $x_+ = x + d$ et $\lambda_+ = \lambda + \mu$

$$\begin{pmatrix} L(x,\lambda) & A(x)^T \\ A(x) & 0 \end{pmatrix} \begin{pmatrix} d \\ \mu \end{pmatrix} = - \begin{pmatrix} \nabla_x l(x,\lambda) \\ c(x) \end{pmatrix}$$

- ou encore $x_+ = x + d$ et $\lambda_+ = \lambda^{PQ}$

$$\begin{pmatrix} L(x,\lambda) & A(x)^T \\ A(x) & 0 \end{pmatrix} \begin{pmatrix} d \\ \lambda^{PQ} \end{pmatrix} = - \begin{pmatrix} \nabla f(x) \\ c(x) \end{pmatrix}$$

- La difficulté du problème réside alors en l'inversion du système qu'il faut faire à chaque itération.

Problème avec contraintes d'inégalités.

- Le problème avec des contraintes linéaires peut se ramener à :

$$\begin{cases} \min f(x) \\ Ax \leq b. \end{cases}$$

- Les conditions d'optimalité KKT s'écrivent :

$$\begin{cases} \nabla f(x_*) + A^T \lambda_* = 0 \\ Ax_* \leq b, \ \lambda_* \leq 0 \\ \lambda_*^T (Ax_* - b) = 0. \end{cases}$$

- La difficulté est de trouver l'ensemble des contraintes actives $I_* = I(x_*)$.

Problème avec contraintes d'inégalités.

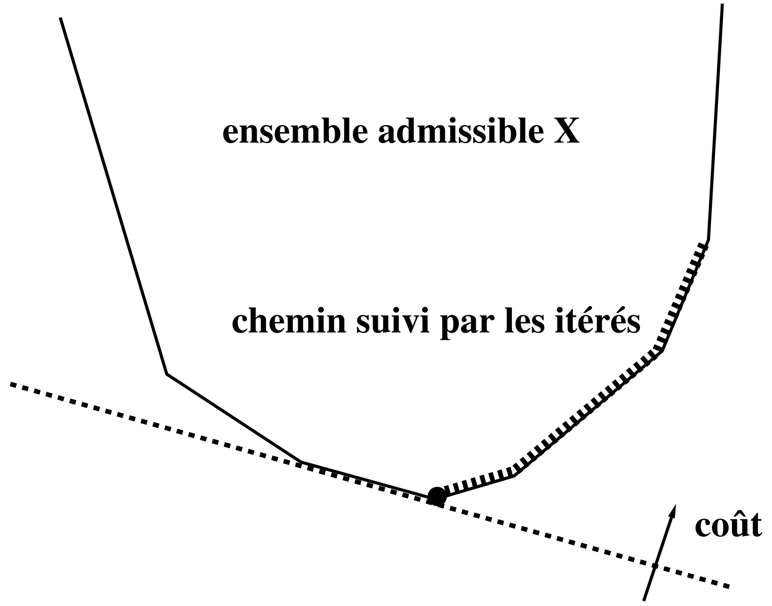


Schéma de l'algorithme du simplexe : on trouve la solution en explorant les sommets de X

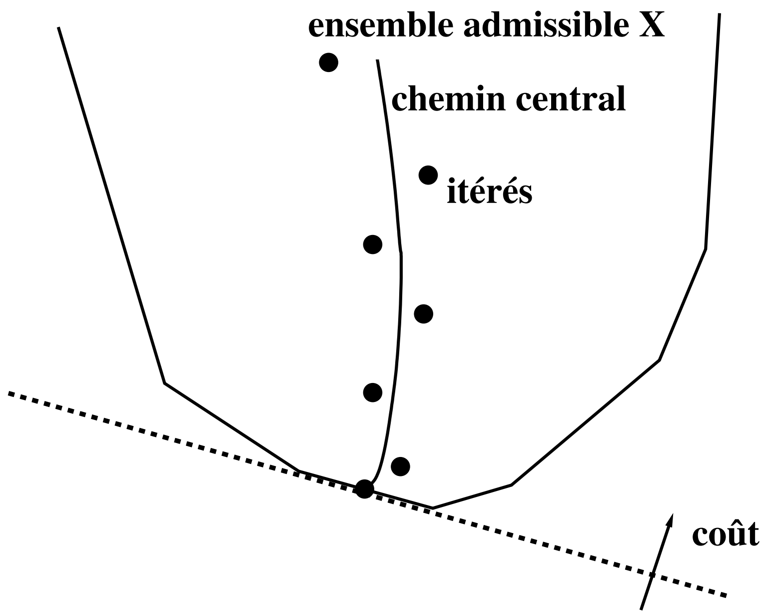


Schéma des algorithmes de points intérieurs : on approche la solution par l'intérieur de X

Méthode du gradient avec projection

- Valable quand X est un convexe fermé
- Facile à mettre en œuvre lorsque l'on a des contraintes de type bornes
- La suite générée est donnée par

$$x_{k+1} = P_X(x_k - \alpha_k g_k)$$

- La méthode étant basée sur la méthode de la plus forte pente, elle donc assez lente
- Il existe des extensions de la méthode pour définir une méthode de Newton avec projection.

Méthode d'activation des contraintes (1)

- `Etape 1` : On active les contraintes avec $i \in I^k$ et on considère le sous-problème

$$(P_{I^k}) \begin{cases} \min f(x) \\ (Ax - b)_{I^k} = 0. \end{cases}$$

et on teste si x_k est une solution approchée de (P_{I^k}) (ou exact si on calcule $\overline{x_k}$ solution de (P_{I^k})) et on teste l'optimalité de x_k pour le problème (P) :

- - si $(\lambda_{I^k}) \leq 0$, x_k est optimal, Stop.
- - sinon, on désactive une unique contrainte j pour laquelle $\lambda_j > 0$ ($I^{k+1} = I^k \backslash \{j\}$) puis retour en `Etape 1`

sinon on peut trouver une direction de descente d en x_k admissible pour les contraintes I^{k+1} ($d_k = \overline{x_k} - x_k$ où $\overline{x_k}$ est la solution de (P_{I^k}) dans le cas quadratique)

Méthode d'activation des contraintes (2)

- `Etape 2 :` Recherche linéaire le long de d_k
  soit $\overline{\alpha_k} > 0$ le pas maximal (au delà duquel on sort de X)
  si la recherche linéaire donne un $\alpha_k < \overline{\alpha_k}$ prendre $x_{k+1} = x_k + \alpha_k d_k$
  sinon ajouter la contrainte bloquante à I^{k+1} et prendre $xk + 1 = x_k + \overline{\alpha_k} d_k$
- `Etape 3 :` retour en `Etape 1`

Cette méthode est valable lorsque les contraintes sont linéaires et bien adaptée aux fonctions quadratiques

Approche Min-Max (1)

- On considère le problème de minimisation suivant : $\inf\limits_{\substack{x \in \mathbb{R}^n \\ c(x)=0}} f(x)$
- Si x_* est solution du problème primal et si λ_* est solution du problème dual alors le couple (x_*, λ_*) est un point selle du Lagrangien $l(x, \lambda)$:

$$\sup_{\lambda \in \mathbb{R}^m} \inf_{x \in \mathbb{R}^n} l(x, \lambda) = \inf_{x \in \mathbb{R}^n} \sup_{\lambda \in \mathbb{R}^m} l(x, \lambda)$$

  Un point selle $(\overline{x}, \overline{y})$ d'une fonction $\phi(x, y)$ est caractérisé par

$$\phi(\overline{x}, y) \leq \phi(\overline{x}, \overline{y}) \leq \phi(x, \overline{y})$$

Approche Min-Max (2)

- Algorithme d'Uzawa :
  - `Etape 1` : initialiser λ_1
  - `Etape 2` : trouver x_k solution du problème $min_x l(x, \lambda_k)$
  - `Etape 3` : Si x_k est satisfaisant, Stop,
  - `Etape 4` : $\lambda_{k+1} = \lambda_k + \alpha_k \nabla_\lambda l(x_k, \lambda_k)$ où $\alpha_k > 0$ et aller en `Etape 2`

- Algorithme d'Arrow et Hurwicz :
  - `Etape 1` : initialiser λ_1 et x_1
  - `Etape 2` : $x_{k+1} = x_k - \alpha_k^1 \nabla_x l(x_k, \lambda_k)$
  - `Etape 3` : $\lambda_{k+1} = \lambda_k + \alpha_k^2 \nabla_\lambda l(x_{k+1}, \lambda_k)$
  - `Etape 4` : si (x_{k+1}, λ_{k+1}) est satisfaisant Stop sinon aller en `Etape 2`.

Programmation Quadratique Successive (PQS) (1)

- Rappel des conditions (KKT) : si les contraintes sont qualifiées en x_* il existe $\lambda_* \in \mathbb{R}^m$ tel que

$$\begin{cases} \nabla_x l(x_*, \lambda_*) = 0 \\ c(x_*)^{\#} = 0 \\ (\lambda_*)_I \geq 0 \\ (\lambda_*)_I^T c_I(x_*) = 0. \end{cases}$$

Avec $(v^{\#})_i = \begin{cases} v_i \text{ si } i \in E, \\ v_i^+ \text{ si } i \in I \end{cases}$ où $v_i^+ = max(v_i, 0)$.

Programmation Quadratique Successive (PQS) (2)

- La linéarisation de ce système au point (x_k, λ_k) donne un déplacement (d_k, μ_k) solution du système :

$$\begin{cases} L_k d_k + A_k^T \mu_k = \nabla_x l_k \\ (c_k + A_k d_k)^{\#} = 0 \\ (\lambda_k + \mu_k)_I \geq 0 \\ (\lambda_k + \mu_k)_I^T (c_k)_I + (\lambda_k)_I^T (A_k d_k)_I = 0. \end{cases}$$

  où $c_k = c_k(x)$, $A_k = A(x_k) = c'(x_k)$, $\nabla_x l_k = \nabla_x l(x_k, \lambda_k)$ et $L_k = \nabla^2_{xx} l(x_k, \lambda_k)$.

- Ce dernier système est formé des conditions d'optimalité du *problème quadratique tangent* :

$$(P_{PQT}) \begin{cases} \min \nabla f(x_k)^T d + \frac{1}{2} d^T L(x_k, \lambda_k) d \\ c_E(x_k) + A_E(x_k) d = 0 \\ c_i(x_k) + A_I(x_k) d \leq 0 \end{cases}$$

Programmation Quadratique Successive (PQS) (3)

- Schéma de l'algorithme de Programmation Quadratique Successive (PQS)
  - `Etape 0` : prendre un point initial (x_1, λ_1) et $k = 1$
  - `Etape 1` : trouver la solution (d_k, λ_k^{PQ}) du problème quadratique (P_{PQT})
  - `Etape 2` : prendre $x_{k+1} = x_k + d_k$ et $\lambda_{k+1} = \lambda_k^{PQ}$
  - `Etape 3` : faire $k = k + 1$ et retour en `Etape 1`.
- Dans l'`Etape 1` à chaque itération on simplifie le problème en se ramenant à un problème quadratique qu'on pourra résoudre par une *méthode d'activation des contraintes* ou une *méthode de points intérieurs*.
- Le problème de déterminer l'ensemble des contraintes actives est reporté sur les sous-problèmes quadratiques plus facile à résoudre.
- On n'a qu'un résultat de convergence local de l'algorithme.
- Il existe aussi des versions quasi-Newtonnienne de PQS.

PQS : globalisation par recherche linéaire et pénalisation (1)

- Pour globaliser la méthode PQS on peut utiliser la recherche linéaire ou la méthode de région de confiance.
- Dans les deux cas on a besoin d'une fonction de mérite qui mesure l'avancement dans la direction d_k et qui prend en compte à la fois f et c.
- on utilise des fonction de pénalisation de la forme:

$$\Theta_\sigma(x) = f(x) + p(x) \text{ avec } p(x) = 0 \text{ ssi } x \in X.$$

- La fonction Θ_σ peut ne pas être différentiable.
- Elle ne sert que dans l'étape de recherche linéaire et non pour le calcul de la direction d_k.
- On parle de pénalisation exacte quand une solution du problème $(P)_{EI}$ est solution de $\min_x \Theta_\sigma(x)$.
  Ceci se réalise sous certaines conditions qui sont facilement vérifiables au cours des itérations.

PQS : globalisation par recherche linéaire et pénalisation (2)

Quelques exemples de fonctions de pénalisation :

- Pénalisation extérieure l_2

$$f(x) + \frac{\sigma}{2}\|c(x)^{\#}\|_2^2.$$

- Lagrangien

$$f(x) + \mu^T c(x).$$

- Lagrangien augmenté (cas $I = \emptyset$)

$$f(x) + \mu^T c(x) + \frac{\sigma}{2}\|c(x)\|_2^2.$$

  Lagrangien augmenté (cas général)

$$\begin{aligned} f(x) \quad &+ \quad \mu_E^T c_E(x) + \frac{\sigma}{2}\|c_E(x)\|_2^2 \\ &+ \quad \sum_{i \in I} \left(\mu_i \max\left(-\frac{\mu_i}{\sigma}, c_i(x)\right) + \frac{\sigma}{2}\left[\max\left(-\frac{\mu_i}{\sigma}, c_i(x)\right)\right]^2 \right). \end{aligned}$$

- Pénalisation de Han

$$f(x) + \sigma\|c(x)^{\#}\|.$$

PQS : globalisation par recherche linéaire et pénalisation (3)

Schéma de l'algorithme PQS avec recherche linéaire :

- trouver (x_k, λ_k^{PQ}) solution du $(P)_{PQT}$
- éventuellement, mettre à jour σ_k pour satisfaire les conditions donnant la pénalisation exacte.
- recherche linéaire : trouver le pas α_k qui permet de diminuer suffisamment Θ_σ.
- prendre $x_{k+1} = x_k + \alpha_k d_k$ et par exemple $\lambda_{k+1} = \lambda_k + \alpha_k(\lambda_k^{PQ} - \lambda_k)$.

Remarques :

- La mise en œuvre pratique de la recherche linéaire est assez technique et dépend de la fonction de pénalisation choisie.
- Sous certaines conditions on peut montrer que l'on a soit
  - divergence de la suite $\{\sigma_k\}$
  - la suite $\{\sigma_k\}$ est stationnaire et auquel cas soit
    * $\Theta_\sigma(x_k)$ diverge
    * x_k sort du domaine admissible mais avec la limite de la distance qui tend vers 0
    * **convergence** !
- Les méthodes de région de confiance peuvent résoudre certains de ces problèmes.

Méthodes de points intérieurs (1)

- L'idée est de trouver le point optimal par l'intérieur du domaine admissible dans le cas de problèmes avec contraintes linéaires.
- Elles ont l'avantage d'avoir une convergence rapide (en $O(\sqrt{n}L)$) et sont en pratique employées pour résoudre des problèmes quadratiques (problèmes quadratiques tangents issus de PQS).
- Problème à résoudre (P) : $\min_{x \in \mathbb{R}^n} f(x) = c^t x + \frac{1}{2} x^t H x,\ Ax = b,\ x \geq 0$
- On introduit une *pénalisation logarithmique* : $\pi(x) = -\sum_{i=1}^n \log(x_i)$ et on s'intéresse aux problème avec pénalisation (P_μ) : $\min_{x \in \mathbb{R}^n} f(x) + \mu\pi(x),\ Ax = b,\ x \geq 0$
  Remarque : ici la pénalisation intervient dans la minimisation et non seulement dans l'étape de recherche linéaire comme vu précédemment.
  On montre que (P_μ) admet une unique solution qui tend vers la solution de (P) quand $\mu \mapsto 0$.
- Conditions d'optimalité de (P_μ) : $\left\{ \begin{array}{l} \nabla f(x) - \mu x^{-1} + A^t \lambda = 0, \\ Ax = b,\ x > 0, \end{array} \right.$ avec $x^{-1} = (x_i^{-1})$

Méthodes de points intérieurs (2)

- On remplace la non linéarité en x^{-1} par l'introduction de $s = \mu x^{-1}$ ce qui donne $xs = \mu \mathbf{1}$ avec $\mathbf{1} = (1, \cdots, 1)$.
- Le système d'optimalité est équivalent à : $\begin{cases} xs = \mu \mathbf{1}, \\ \nabla f(x) + A^t \lambda = s, \\ Ax = b, \\ x \geq 0, \ s \geq 0 \end{cases}$
- En multipliant la deuxième ligne du système précédent par une matrice $M \in N(A) = Im(A)^{\perp}$ on peut ramener ce dernier au système (on fait disparaître λ)

$$\begin{cases} xs = \mu \mathbf{1}, \\ Qx + Rs = h, \\ x \geq 0, \ s \geq 0 \end{cases}$$

- On appellera *trajectoire centrale* l'ensemble des (x, s) solution du système précédent.

Méthodes de points intérieurs (3) : algorithmes prédicteurs correcteurs

- On définit le *petit voisinage* $\mathcal{V}_\alpha$ et le *grand voisinage* $\mathcal{N}_\epsilon$ par :

$$\mathcal{V}_\alpha = \{(x, s, \lambda, \mu),\ \|\frac{xs}{\mu} - \mathbf{1}\| \leq \alpha,\ (x, s, \lambda) \text{ réalisable } \}$$

$$\mathcal{N}_\alpha = \{(x, s, \lambda, \mu),\ \epsilon\mathbf{1} \leq \frac{xs}{\mu} \leq \epsilon^{-1}\mathbf{1},\ (x, s, \lambda) \text{ réalisable } \}$$

  avec $\alpha, \epsilon \in (0, 1)$ et où un point (x, s, λ, μ) est dit réalisable s'il satisfait les contraintes linéaires du système définissant la *trajectoire centrale*.

- L'algorithme va s'arrêter lorsque l'on trouve un point (x, s, μ) dans $\mathcal{V}_\alpha$ par exemple et tel que $\mu \leq \mu_\infty$.

- La valeur de $\overline{L}$ donnée dans l'expression de la vitesse de convergence est $\overline{L} = \log(\frac{\mu_0}{\mu_\infty})$ où $(x_0, s_0, \mu_0) \in \mathcal{V}_\alpha$ est le point initial.

Méthodes de points intérieurs (4)

Schéma de l'algorithme pour $\mathcal{G}$ peut être $\mathcal{V}_\alpha$ ou $\mathcal{N}_\epsilon$:

- `Etape 0 :` test d'arrêt : $\mu_k < \mu_\infty$?
- `Etape 1 :` $x = x_k$, $s = s_k$ et $\mu = \mu_k$
- `Etape 2 :` *Centralisation* calculer (u^c, v^c) solution de
  $Qu + Rv = 0$ et $su + xv = \mu\mathbf{1} - xs$
  * $x(\theta) = x + \theta u^c$, $s(\theta) = s + \theta v^c$
  * calculer $\theta^c \in]0, 1]$ tel que $(x(\theta^c), s(\theta^c), \mu) \in \mathcal{G}$
  * $x = x(\theta^c)$ et $s = s(\theta^c)$
- `Etape 3 :` *Déplacement affine* calculer (u^a, v^a) solution de
  $Qu + Rv = 0$ et $su + xv = -xs$
  * $x(\theta) = x + \theta u^a$, $s(\theta) = s + \theta v^a$, $\mu(\theta) = (1 - \theta)\mu$
  * calculer la **plus grande valeur** $\theta^a \in]0, 1]$ tel que
  $(x(\theta^a), s(\theta^a), \mu(\theta^a)) \in \mathcal{G}$, $\forall \theta \in [0, \theta^a]$
  * $x_{k+1} = x(\theta^a)$, $s_{k+1} = s(\theta^a)$, $\mu_{k+1} = (1 - \theta^a)\mu_k$, $k = k + 1$ et aller en `Etape 0`.

Méthodes de points intérieurs (5)

- On appelle algorithme de *petit voisinage* le même algorithme précédent avec $\mathcal{G} = \mathcal{V}_\alpha$ avec $\alpha \leq \frac{1}{2}$ et où le pas de centralisation est pris égal à 1.
- On appelle algorithme de *grand voisinage* le même algorithme précédent avec $\mathcal{G} = \mathcal{V}_\epsilon$ avec $\epsilon \leq \frac{1}{2}$ et où le pas de centralisation n'est plus égal à 1 mais calculé en général approximativement.
- L'algorithme de *grand voisinage* est théoriquement moins bon que l'algorithme de *petit voisinage* mais il est meilleur numériquement.
- Dans les deux cas on suppose que l'on part d'un point initial *réalisable* ce qui n'est pas toujours facile à trouver. Il existe des classes d'algorithmes dit *non réalisables* qui s'affranchissent de cette condition.
- Il existe aussi des variantes dites à *un pas* où à chaque itération on ne calcule qu'un seul pas ou lieu du calcul d'un *pas de centralisation* suivi d'un *pas affine*.

Calcul du gradient

différentes approches pour calculer le gradient d'une fonction $f : \mathbb{R}^n \mapsto \mathbb{R}^m$:

- calcul symbolique
- différences finies $\dfrac{\partial f}{\partial x} = \lim\limits_{h \mapsto 0} \dfrac{f(x+h) - f(x)}{h}$
  très sensible numériquement
  très coûteuse : il faut un calcul de f pour chaque direction
- en utilisant l'adjoint
- par différentiation automatique
  - le principe est de générer à partir d'un programme (Fortran, C, C++, ...) P_1, prenant en entrée x et calculant à la sortie $f(x)$, un programme P_2 qui calcule $\nabla_x f(x)$.
  - la plupart des programmes proposent deux modes: un mode tangent (similaire aux différences finis) et un mode adjoint (similaire à la méthode de l'adjoint)
  - si $n \ll m$ utiliser le mode tangent
  - si $n \gg m$ utiliser le mode adjoint

Différentiation automatique sur un exemple (1)

- Soit la fonction $f(v_1, v_2) = e^{\frac{v_1^2 \sin(v_1+v_2^2)}{v_1+v_2^2}}$
- P_1 :
  - ligne 1 : `v3 = v1 + v2**2`
  - ligne 2 : `v4 = v1**2*sin(v3)`
  - ligne 3 : `v4 = v4/v3`
  - ligne 4 : `v5 = exp(v4)`

Différentiation automatique sur un exemple (2) : mode linéaire tangent

- code linéaire tangent calculant les dérivées directionnelles dans la direction $d = (d_1, d_2)$:
  - `v1tl = d1`
  - `v2tl = d2`
  - `v3tl = v1tl + 2*v2*v2tl`
  - `v3 = v1 + v2**2`
  - `v4tl = 2*v1*v1tl*sin(v3) + v1**2*cos(v3)*v3tl`
  - `v4 = v1**2*sin(v3)`
  - `aux = v4/v3`
  - `v4tl = (v4tl - aux*v3tl)/v3`
  - `v4 = aux`
  - `aux = exp(v4)`
  - `v5tl = aux*v4tl`
  - `v5 = aux`

Différentiation automatique sur un exemple (3) : Mode adjoint

sauvegarde de la trajectoire

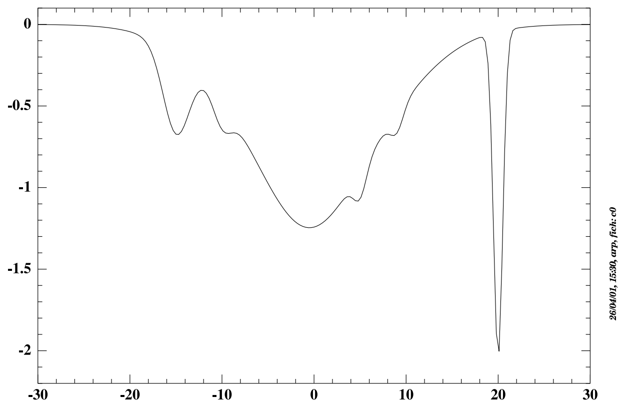
```
- save3 = v3
- v3 = v1 + v2**2
- saveaux = aux - aux =
sin(v3)
- save4 = v4
- v4 = v1**2*aux
- save41 = v4
- v4 = v4/v3
- save5 = v5
- v5 = exp(v4)
```

code linéaire cotangent

```
- v1ad=0, v2ad=0, v3ad=0, v4ad=0, auxad=0, v5ad = 1
- v5 = save5
- v4ad = v4ad + exp(v4)*v5ad
- v5ad = 0
- v4 = save41
- v3ad = v3ad - v4ad*v4*v3ad/v3**2
- v4ad = v4ad/v3
- v4 = save4
- auxad = auxad + v4ad*2*v1*aux
- v1ad = v1ad + 2*v1*auxad
- v4ad = 0
- aux = saveaux
- v3ad = v3ad +auxad*cos(v3ad)
- auxad = 0
- v3 = save3
- v1ad = v1ad + v3ad
- v2ad = v2ad + v3ad*2*v2
- v3ad = 0
```

Méthode de diffusion

- Le but est de trouver le minimum robuste d'une fonction ϕ de la forme suivante :

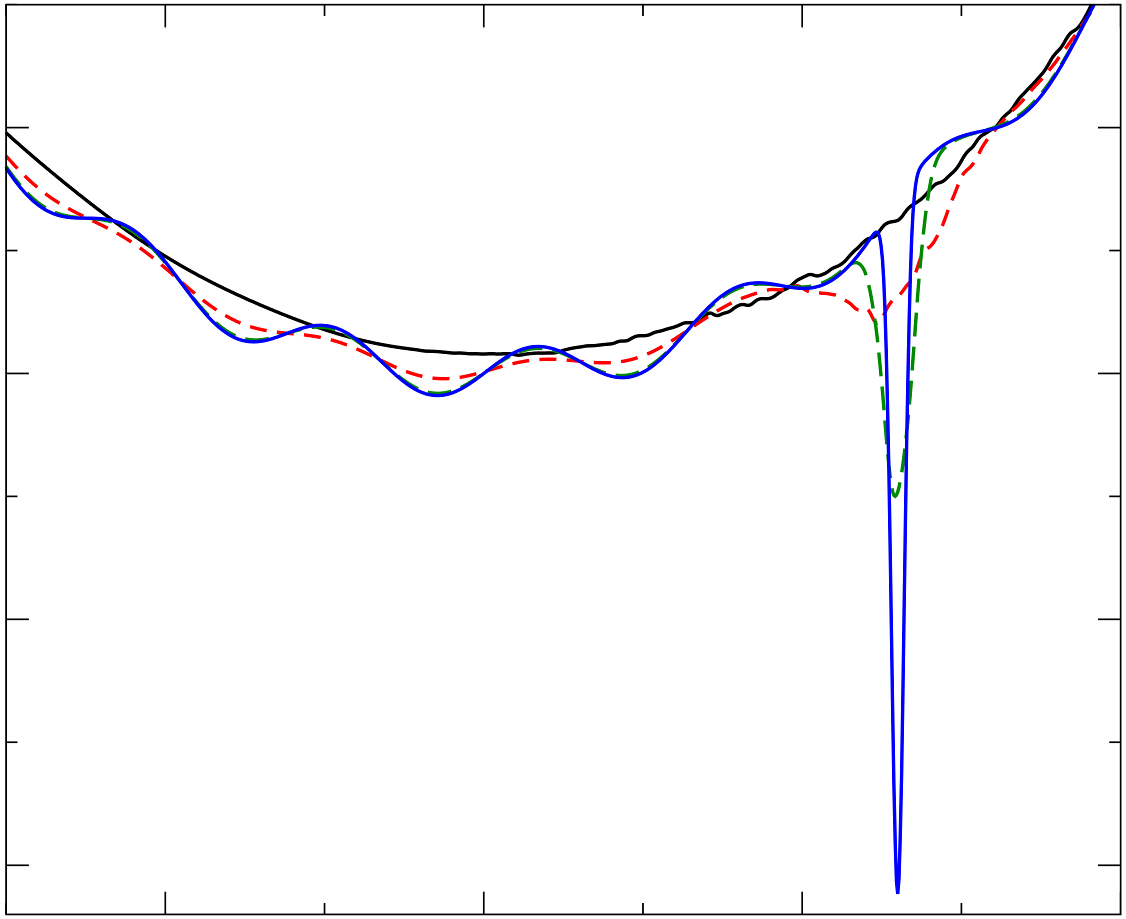


- L'idée est de "lisser" la fonction en la convolant contre le noyau de la chaleur :

$$\begin{cases} \dfrac{\partial u}{\partial t} - \Delta u = 0, \\ u(0,x) = \varphi(x). \end{cases}$$

- Dans $\mathbb{R}^d$, la solution de l'équation de la chaleur est donnée par

$$u(t,x) = \frac{1}{(4\pi t)^{\frac{d}{2}}} \int_{\mathbb{R}^d} e^{\frac{-\|x-y\|2}{4t}} \varphi(y) dy.$$



Les fonctions $u(t, x)$ pour différentes valeurs de t.

Intégration de $u(t,x)$

L'intégration de $u(t,x)$ se fait par une méthode de Monte Carlo. Le principe de cette méthode est le suivant :

- Générer une suite (G_n) de nombres aléatoires suivant la loi Normale $N(0,1)$.
- on calcule $u(t,x)$ avec :

$$u(x,t) = \sum_{i=1}^{n} \varphi(x + g_i\sqrt{2dt}),$$

- pour le calcul du gradient de la fonction $u(x,t)$ on peut soit :
  - utiliser la même formule appliquée au gradient :

$$\nabla u(x,t) = \sum_{i=1}^{n} \nabla\varphi(x + g_i\sqrt{2dt})$$

  - porter la dérivée sur le noyau et on obtient alors :

$$\nabla u(x,t) = \sum_{i=1}^{n} \frac{g_i}{\sqrt{2dt}}\, \varphi(x + g_i\sqrt{2dt})$$

Algorithme

Principe de la méthode :

- `Etape 0 :` initialiser le temps t^0 et le point de départ x_0 et fixer $0 < a < 1$.
- `Etape 1 :` recherche du minimum $\overline{x}_1^k$ de $u(t^k, x)$ en partant de x_0^k.
- `Etape 2 :` si $\overline{x}_1^k = \overline{x}_1^{k-1}$ on arrête, et sinon prendre $x_0^{k+1} = \overline{x}_1^k$, $t^{k+1} = at_k$ et aller en `étape 1`.

Un version plus robuste consiste à répéter l'algorithme précédent avec à chaque fois $t^0 = \beta t^0$ ($\beta > 1$) et en prenant comme point initial x_0 le résultat $\overline{x}^1$ donnée par l'itération précédente.

Bibliographie

- J. Frédéric BONNANS, Jean Charles GILBERT, Claude LEMARECHAL, and Claudia SAGASTIZABAL. *Optimisation Numérique : aspects théoriques et pratiques.* Springer, 1997.

- J. C. GILBERT. *Optimisation Différentialble : Théorie et Algorithmes.* *Notes de cours.*

- Maxime BARRAULT and Claude LE BRIS. *Optimisation numérique et différentiation automatique pour un problème industriel.* Rapport CERMICS 99-179, juillet 1999.