Chapitre 6: Optimisation avec contraintes linéaire

Abdelkrim EL MOUATASIM
Professeur Habilité en Mathématique Appliquée
https://sites.google.com/a/uiz.ac.ma/elmouatasim/

FPO - SMI - S6

28 mars 2018

Plan

Organisation du cours

Considérons le problème d'optimisation :

$$\begin{cases} \min \quad f(\mathbf{x}) \\ \text{s.c.} \quad A\mathbf{x} = b \\ \qquad\quad \mathbf{x} \geqslant 0 \end{cases} \tag{1}$$

où

- A est une matrice de taille $m \times n$ (m lignes, n colonnes) de $\operatorname{rang}(A) = m < n$.
- f est une fonction de $\mathbb{R}^n$ à valeur dans $\mathbb{R}$ est de classe C^1.
- b un vecteur de dimension m.

Rappelons que, quelque soit le polyèdre initial, il est toujours possible de se ramener à ce type de contraintes.

Organisation du cours

Considérons le problème de programmation linéaire suivant :

$$\begin{cases} \min & \nabla f(\mathbf{x}_k)\mathbf{y} \\ \text{s.c.} & A\mathbf{y} = b \\ & \mathbf{y} \geqslant 0 \end{cases} \tag{2}$$

Supposons que l'on résolve ce problème en un point quelconque, qui n'est pas un minimum local du programme (1).

Le programme linéaire (2) possédera une solution y_k telle que

$$\nabla f(\mathbf{x}_k)(y_k - x_k) < 0,$$

c'est-à-dire, la direction $(y_k - x_k)$ constitue une direction de descente réalisable pour le programme (1). Nommant d cette direction, nous allons choisir

$$x_{k+1} = x_k + \theta d$$

de sorte que θ minimise f sur la ligne joignant x_k et y_k, tout en demeurant réalisable. Alors, $f(x_{k+1}) < f(x_k)$, et nous pouvons recommencer le processus.

Nous en déduisons la suite de quantités suivants :

$$\begin{aligned} y_k &= \arg\min_{y} \nabla f(x_k) y : Ay = b, y \geqslant 0; \\ d_k &= y_k - x_k; \\ \theta_k &= \arg\min_{0 \leqslant \theta \leqslant 1} f(x_k + \theta d_k); \\ x_{k+1} &= x_k + \theta_k d_k. \end{aligned}$$

Les observations précédentes montrent que la suite $f(x_k)$ est décroissante. Voir :
Jean-Pierre Dussault. Optimisation mathématique. Université de Sherbrooke, Canada. 2015.

L'algorithme de Frank-Wolfe utilise dans son calcul de direction la minimisation d'une fonction linéaire et donc résolue par un algorithme de programmation linéaire, i.e.

$$y_k = \arg\min_{y \in E} \nabla f(x_k) y$$

et $d = y_k - x_k$. Si on résout le programme linéaire avec l'algorithme du simplexe, la solution calculée sera toujours un point extrême de domaine réalisable E et donc on peut toujours choisir $\theta_{\max=1}$.
La condition d'optimalité est considérée satisfaite lorsque $\|\nabla f(x_k)(y_k - x_k)\|$ est assez petit.

TP : Groupe 3

Programmer l'algorithme du Frank-Wolfe avec la règle d'or sous Scilab/Matlab.

Organisation du cours

Méthode du Gradient Réduit

On définit une partition des variables en $B \subseteq \{1, ..., n\}$ et $N = \{1, ..., n\} - B$ telle que $|B| = m$ et A_B, la matrice formée des colonnes de A indexées dans B, soit inversible.
Quitte à permuter les colonnes, A se décompose en

$$(A_B \quad A_N)$$

où A_N est la matrice formée des colonnes de A indexées dans N.
Quitte à permuter les lignes, tout x se décompose en

$$\begin{pmatrix} x_B \\ x_N \end{pmatrix}$$

où x_B est formé des composantes de x indexées dans B et x_N est formé des composantes de x indexées dans N.

Ainsi

$$Ax = b \Leftrightarrow (A_B \;\; A_N) \begin{pmatrix} x_B \\ x_N \end{pmatrix} = A_B x_B + A_N x_N = b$$

et si A_B est inversible alors on peut exprimer x_B en fonction de x_N par la relation

$$x_B = A_B^{-1} b - A_B^{-1} A_N x_N.$$

Example

Soit le système $\begin{cases} x_1 + 2x_2 + 3x_3 - x_4 = 6 \\ x_1 + 3x_2 + 2x_3 + 5x_4 = 4 \end{cases}$

Ici $A = \begin{pmatrix} 1 & 2 & 3 & -1 \\ 1 & 3 & 2 & 5 \end{pmatrix}$

Choisissons $B = \{1, 3\}, \quad N = \{2, 4\}$. Alors $A_B = \begin{pmatrix} 1 & 3 \\ 1 & 2 \end{pmatrix}$,

$A_N = \begin{pmatrix} 2 & -1 \\ 3 & 5 \end{pmatrix}$, $x_B \begin{pmatrix} x_1 \\ x_3 \end{pmatrix}$, $x_N \begin{pmatrix} x_2 \\ x_4 \end{pmatrix}$

$$\begin{aligned} (A_B \;\; A_N) \begin{pmatrix} x_B \\ x_N \end{pmatrix} &= \begin{pmatrix} 1 & 3 \\ 1 & 2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_3 \end{pmatrix} + \begin{pmatrix} 2 & -1 \\ 3 & 5 \end{pmatrix} \begin{pmatrix} x_2 \\ x_4 \end{pmatrix} \\ &= \begin{pmatrix} x_1 + 3x_3 + 2x_2 - x_4 \\ x_1 + 2x_3 + 3x_2 + 5x_4 \end{pmatrix} \end{aligned}$$

En respectant cette partition, le problème (1) peut s'écrire comme suit :

$$\begin{cases} \min \quad f(\mathbf{x}_B, \mathbf{x}_N) \\ \text{s.c.} \quad (i) \quad B\mathbf{x}_B + N\mathbf{x}_N = b, \\ \qquad (ii) \quad \mathbf{x}_B > 0, \\ \qquad (iii) \quad \mathbf{x}_N \geqslant 0. \end{cases} \tag{3}$$

On peut considérer $\mathbf{x}_N$ comme un vecteur de variables indépendantes et $\mathbf{x}_B$ comme un vecteur de variables dépendantes.
En effet, lorsque $\mathbf{x}_N$ est fixé, $\mathbf{x}_B$ est unique et est donné par (3.i).

Hypothèse

- On fait l'hypothèse que le problème (1) admet au moins une solution.
- On fera l'hypothèse de non-dégénérescence à savoir $\forall B$ t.q. A_B inversible, $A_B^{-1} b > 0$.

Comme pour la programmation linéaire, il est plus commode d'utiliser la transposée du gradient de f (vecteur ligne) que le gradient de f (vecteur colonne).
On note

$$\frac{\partial f}{\partial x} = (\frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_n}) = \nabla f^t$$

(On aura reconnu la matrice jacobienne de f).
De manière similaire aux notations précédentes on note

$$\frac{\partial f}{\partial x_N} = (\frac{\partial f}{\partial x_{j_1}}, \ldots, \frac{\partial f}{\partial x_{j_{n-m}}}) \quad j_k \in N$$

et

$$\frac{\partial f}{\partial x_B} = (\frac{\partial f}{\partial x_{i_1}}, \ldots, \frac{\partial f}{\partial x_{i_m}}) \quad i_k \in B.$$

Le principe de la méthode du gradient réduit est de considérer le problème (3) uniquement en terme de variables $\mathbf{x}_N$.
On résout localement le problème :

$$\min_{\mathbf{x}_N \in \mathbb{R}^m} g(\mathbf{x}_N); \quad \mathbf{x}_N \geqslant 0, \tag{4}$$

avec $g(\mathbf{x}_N) = f(\mathbf{x}_B(\mathbf{x}_N), \mathbf{x}_N)$.
Le gradient de g est appelé gradient réduit de f et sa transposée s'exprime par

$$\frac{\partial g}{\partial x_N} = \frac{\partial f}{\partial x_N} - \frac{\partial f}{\partial x_B} A_B^{-1} A_N. \tag{5}$$

Le gradient réduit $r(x)$ correspond au vecteur des coûts réduits en programmation linéaire :

$$r(x) = \frac{\partial f}{\partial x}(x) - \frac{\partial f}{\partial x_B}(x) A_B^{-1} A.$$

On a : $r_N(x) = \frac{\partial g}{\partial x_N}(x)$ et $r_B(x) = 0$.

Example (suite)

Soit le système

$$\begin{cases} x_1 + 2x_2 + 3x_3 - x_4 & = 6 \\ x_1 + 3x_2 + 2x_3 + 5x_4 & = 4 \end{cases}$$

et

$$f(x_1, x_2, x_3, x_4) = x_1^2 + x_2^2.$$

Choisissons $B = \{1, 3\}, \quad N = \{2, 4\}$.
On exprime x_1 en fonction de x_2, x_4, on obtient

$$g(x_2, x_4) = (-5x_2 - 17x_4)^2 + x_2$$

et

$$(\frac{\partial g}{\partial x_2} \quad \frac{\partial g}{\partial x_4}) = (52x_2 + 170x_4 \quad 170x_2 + 578x_4).$$

Comme dans l'algorithme du simplexe, toute variable de base qui s'annule quitte la base pour être remplacée par une variable hors base. Le choix de la variable hors base n'est pas toujours unique ; en effet, même si la solution de base courante n'est pas dégénérée, il se peut qu'il y ait plusieurs variables hors base positives (ces variables sont appelées superbasiques). Il se peut aussi que le minimum de f dans la direction d soit atteint sans qu'aucune variable de base ne s'annule. On répète alors le processus sans changer de base.

Definition

On appelle algorithme du **simplexe convexe** la variante de l'algorithme du gradient réduit qui consiste à ne modifier qu'une seule variable hors base à chaque itération.

Principe de la méthode

Plutôt que de considérer f, fonction de n variables, on considère g fonction de $n-m$ variables. Outre le fait que le nombre de variables est ainsi mois important, l'intérêt est que ces $n-m$ variables sont soumises seulement aux contraintes relativement simple de non négativité.

On part d'un point x réalisable et on choisit une partition des variables (B, N) vérifiant les critères énoncés plus haut.
On va effectuer un déplacement dans la direction opposée au gradient réduit mais en prenant soin de ne pas rendre négatives les variables indexées dans N et nulles.

Definition

On définit la direction d, se décomposant en d_B et d_N, de la manière suivant :

- $d_j = \begin{cases} 0 & \text{si } r_j(x) > 0 \text{ et } x_j = 0 \\ -r_j(x) & \text{sinon} \end{cases} \qquad (j \in N)$
- d_N étant défini, d_B est déterminé via les contraintes d'égalité par $d_B = -A_B^{-1} A_N d_N$.

Principe de la méthode

Si $d_N \neq 0$ le déplacement se fait jusqu'à atteindre le minimum de f dans la direction d mais en respectant les contraintes de non négativité des variables. On obtient un nouveau point x'. Si une de ses composantes dans B, disons s, s'annule on construit une nouvelle partition des variables

$$B' = B - \{s\} + \{r\} \quad \text{et} \quad N' = N + \{s\} - \{r\}$$

où r est choisi de sorte que $A_{B'}$ soit inversible et que x'_r soit non nul puis on réitère. Cette opération, appelée changement de base, est nécessaire car sinon on risque de faire du surplace, la nullité de x'_s bloquant le déplacement suivant. Le fait de faire passer s dans N permet de contrôler la variation de la coordonnée s qui être que positive ou nulle par construction de la direction de déplacement.

Critère d'arrêt :

Si $d_N = 0$ les conditions de Kuhn-Tucker sont satisfaites et on arrête.

Theorem

Si la méthode de gradient réduit construit une direction de déplacement nulle, i.e. $d_N = 0$, le point courant x vérifie les conditions de Kuhn-Tucker.

Démonstration.

Les conditions de Kuhn-Tucker s'écrivent :

$$\begin{cases} \frac{\partial f}{\partial x}(x) + \mu A - \lambda = 0 \\ \lambda \geqslant 0 \\ \lambda_i x_i = 0 \quad i = 1, \ldots, n \end{cases}$$

Soient $\mu = -\frac{\partial f}{\partial x_B}(x) A_B^{-1}$, $\lambda_B = 0$, $\lambda_N = r_N(x)$. x, μ, λ vérifient-ils ces conditions ? Pour la première égalité un simple calcul le montre. La deuxième condition ($\lambda \geqslant 0$) est vérifiée car $d_N = 0 \Rightarrow r_N(x) \geqslant 0$ d'après la définition de d_N. La troisième condition (complémentarité) est vérifiée car $d_j = 0 \Rightarrow (r_j > 0 \text{ et } x_j = 0) \text{ ou } (-r_j = 0) \quad \forall j \in N$ toujours par définition de d_N. □

Algorithme du gradient réduit

On se donne une base initiale et un vecteur initial $\mathbf{x}^0 = (\mathbf{x}_B^0, \mathbf{x}_N^0)$ tel que $\mathbf{x}_B^0 > 0$, $k_{\max}$, on pose $k = 0$ et $d_N^0 \neq 0$.

Tant que $(d_N^k \neq 0$ et $k < k_{\max})$ **faire**

Calcul du gradient réduit (5).

Pour $i \in N$ **faire**

Si $(x_i^k = 0$ et $r_i(\mathbf{x}^k) \geqslant 0)$ **Alors**

$d_i^k = 0$

Sinon

$d_i^k = -r_i(\mathbf{x}^k)$

Fin Si

Fin Pour

$d_B^k = B^{-1} N d_N^k$.

$\mu_k = \arg \min_{0 \leqslant \mu \leqslant \mu_{\max}} f(\mathbf{x}^k + \mu d^k)$ avec $\mu_{\max} = \min_{j \in B, d_j < 0} \{-\frac{x_j^k}{d_j}\}$.

$\mathbf{x}^{k+1} = \mathbf{x}^k + \mu_k d^k$.

Si $(\exists s \in B \quad x_s = 0)$ **Alors**

soit $r \in N$ t.q. A'_B inversible et x_r maximum

$B = B'$ et $N = N'$

Fin Si

$k = k + 1$

Fait

Mini-projet : Facultative

Programmer l'algorithme du gradient réduit avec la règle d'or sous Java.

Example

$$\begin{cases} \min \quad f(x) = & x_1^2 + x_2^2 + x_3^2 + x_4^2 - 2x_1 - 3x_4 \\ \text{s.c.} & 2x_1 + x_2 + x_3 + 4x_4 = 7 \\ & x_1 + x_2 + 2x_3 + x_4 = 6 \\ & x \geqslant 0 \end{cases}$$

Soient $x^0 = (2, 2, 1, 0)^t$, x_1 et x_2 les variables de base et $B = \begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix}$ la base.
On a : $\nabla f(x^0) = (2, 4, 2, -3)^T$

$$r(x^0) = (2, 4, 2, -3) - (2, 4) \begin{pmatrix} 1 & -1 \\ -1 & 2 \end{pmatrix} \begin{pmatrix} 2 & 1 & 1 & 4 \\ 1 & 1 & 2 & 1 \end{pmatrix} = (0, 0, -8, -1),$$

par conséquent, $d_3 = 8$ et $d_4 = 1$.

Example (suite)

On a :

$$d_B = B^{-1}Nd_N = (5, -22)^T$$

et $\mu \leqslant \mu_{\max} = \min_{j \in B, d_j < 0} \{-\frac{x_j^k}{d_j}\} = \frac{1}{11}$.

On vérifie que le minimum de la fonction $f(x^0 + \mu d)$ pour $\mu \in [0, \frac{1}{11}]$ est obtenu pour $\mu = \frac{1}{11}$.

La variable x_2 quitte alors la base pour y être remplacée par x_3 ou x_4, au choix.

Remarque pratique :

au cours de l'algorithme, il faut effectuer le changement de base
$B' = B - \{s\} + \{r\} \quad N' = N + \{s\} - \{r\}$ et il faut alors calculer $A_{B'}^{-1}$.
Ce calcul peut être fait à moindre coût.
On a l'identité $A_{B'}^{-1} = (A_B^{-1} A_{B'})^{-1} A_B^{-1}$ $A_{B'}$ étant obtenue à partir de A_B en remplaçant la colonne $A_{\{s\}}$ par la colonne $A_{\{r\}}$, on a

$$A_B^{-1} A_{B'} = \begin{pmatrix} 1 & 0 & y_1 & & 0 \\ 0 & \ddots & \vdots & & \\ \vdots & & p & & \\ & & \vdots & \ddots & \\ 0 & & y_m & & 1 \end{pmatrix} \text{ où } A_B^{-1} A_{\{r\}} = \begin{pmatrix} y_1 \\ \vdots \\ p \\ \vdots \\ y_m \end{pmatrix}$$

si $p \neq 0$ alors $(A_B^{-1} A_{B'})^{-1} = \begin{pmatrix} 1 & 0 & \frac{-y_1}{p} & & 0 \\ 0 & \ddots & \vdots & & \\ \vdots & & \frac{1}{p} & & \\ & & \vdots & \ddots & \\ 0 & & \frac{-y_m}{p} & & 1 \end{pmatrix}$

On peut donc calculer $A_{B'}^{-1}$ en prémultipliant A_B^{-1} par une matrice simple à obtenir. p est appelé pivot de la colonne r.

Example

Reconsidérons le système $\begin{cases} x_1 + 2x_2 + 3x_3 - x_4 = 6 \\ x_1 + 3x_2 + 2x_3 + 5x_4 = 4 \end{cases}$

$A = \begin{pmatrix} 1 & 2 & 3 & -1 \\ 1 & 3 & 2 & 5 \end{pmatrix}$

Choisissons $B = \{1,3\}, \quad N = \{2,4\}$. Alors $A_B = \begin{pmatrix} 1 & 3 \\ 1 & 2 \end{pmatrix}$,

$A_B^{-1} = \begin{pmatrix} -2 & 3 \\ 1 & -1 \end{pmatrix}$

Effectuons le changement de base $B' = \{2,3\}, \quad N' = \{1,4\}$

$A_B^{-1} A_{\{2\}} = \begin{pmatrix} -2 & 3 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 2 \\ 3 \end{pmatrix} = \begin{pmatrix} 5 \\ -1 \end{pmatrix}$ $p = 5 \neq 0$ et donc

$A_{B'} = \begin{pmatrix} 2 & 3 \\ 3 & 2 \end{pmatrix}$ est inversible $(A_B^{-1} A_{B'})^{-1} = \begin{pmatrix} \frac{1}{5} & 0 \\ \frac{1}{5} & 1 \end{pmatrix}$ et

$A_{B'}^{-1} = \begin{pmatrix} \frac{1}{5} & 0 \\ \frac{1}{5} & 1 \end{pmatrix} \begin{pmatrix} -2 & 3 \\ 1 & -1 \end{pmatrix} = \begin{pmatrix} \frac{-2}{5} & \frac{3}{5} \\ \frac{3}{5} & \frac{-2}{5} \end{pmatrix}$

Sous l'hypothèse de non-dégénérescence, l'opération changement de base est toujours possible. En effet $x_B + A_B^{-1}A_N x_N = A_B^{-1}b > 0$ et si on note p_j le pivot de la colonne j (relatif à x_s) on a :
$s \in B$ t.q. $x_s = 0 \Rightarrow \sum_{j \in N} p_j x_j > 0 \Rightarrow \exists r \in N$ t.q. $p_r x_r \neq 0$.

Example (suite)

Pour $B = \{1, 3\}$, on a $\begin{pmatrix} x_1 \\ x_3 \end{pmatrix} + A_B^{-1}A_N \begin{pmatrix} x_2 \\ x_4 \end{pmatrix} = A_B^{-1}b$ soit

$$\begin{pmatrix} x_1 \\ x_3 \end{pmatrix} + \begin{pmatrix} -2 & 3 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 2 & -1 \\ 3 & 5 \end{pmatrix} \begin{pmatrix} x_2 \\ x_4 \end{pmatrix} = \begin{pmatrix} -2 & 3 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 6 \\ 4 \end{pmatrix}$$

soit encore $\begin{pmatrix} x_1 \\ x_3 \end{pmatrix} + \begin{pmatrix} 5 & 17 \\ -1 & -6 \end{pmatrix} \begin{pmatrix} x_2 \\ x_4 \end{pmatrix} = \begin{pmatrix} 0 \\ 2 \end{pmatrix}$

$x_3 = 0 \Rightarrow -x_2 - 6x_4 = 2$

Les pivots des colonnes A_2 et A_4 (relatifs à x_3) sont -1 et -6. S'ils étaient nuls on aurait $0 = 2$. On voit que la condition de non-dégénérescence n'est pas nécessaire puisqu'ici $A_B^{-1}b$ a une composante (celle associée à x_1) nulle et pourtant on peut effectuer les changements de base $B - \{1\} + \{2\}$ et $B - \{1\} + \{4\}$, les pivots (5 et 17) étant non nuls.

Example (Récapitulatif)

On considère le problème d'optimisation sous contraintes suivante :

$$\begin{cases} \min & x_1^2 + x_2^2 \\ \text{s.c.} & x_1 + x_2 \geqslant 1 \\ & x_1 \geqslant 0, x_2 \geqslant 0 \end{cases} \qquad (P)$$

(P) revient à le pb d'optimisation avec contraintes d'égalité

$$\begin{cases} \min & x_1^2 + x_2^2 \\ \text{s.c.} & x_1 + x_2 + x_3 = 1 \\ & x_1 \geqslant 0, x_2 \geqslant 0, x_3 \geqslant 0 \end{cases}$$

Choisissons le point initial $P_0 = \begin{pmatrix} 2 \\ 0 \\ 1 \end{pmatrix}$ et $B = \{3\}$, $N = \{1, 2\}$.

Example (suite)

Itération 1.

On a $x_3 = -1 + x_1 + x_2$. On en déduit $g(x_1, x_2) = f(x_1, x_2, -1 + x_1 + x_2)$, $\frac{\partial g}{\partial x_N} = (\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}) = (2x_1, 2x_2)$ et $r(P_0) = (4\ 0)$. La direction de déplacement est $\begin{pmatrix} -4 \\ 0 \\ -4 \end{pmatrix}$.

$$\begin{pmatrix} 2 \\ 0 \\ 1 \end{pmatrix} + \alpha \begin{pmatrix} -4 \\ 0 \\ -4 \end{pmatrix} \geqslant 0 \Rightarrow \begin{cases} \alpha \leqslant \frac{1}{2} \\ \alpha \leqslant \frac{1}{4} \end{cases} \Rightarrow \alpha_{\max} = \frac{1}{4}$$

Le minimum de $f(2 - 4\alpha, 0, 1 - 4\alpha)$ sous la contrainte $0 \leqslant \alpha \leqslant \frac{1}{4}$ est atteint en $\alpha_1 = \frac{1}{4}$.

Le nouveau point est $P_1 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$. Sa coordonnée 3 est nulle. Il faut effectuer un changement de base. L'indice 1 est l'indice dans N associé à la coordonnée de P_1 la plus grande : on forme la nouvelle décomposition $B = \{1\}$ et $N = \{2, 3\}$.

Example (suite)

Itération 2.

On a $x_1 = 1 - x_2 + x_3$ d'où $g(x_2, x_3) = f(1 - x_2 + x_3, x_2, x_3)$,
$\frac{\partial g}{\partial x_N} = (-\frac{\partial f}{\partial x_1} + \frac{\partial f}{\partial x_2} \quad \frac{\partial f}{\partial x_1} + \frac{\partial f}{\partial x_3}) = (-2x_1 + 2x_2 \quad 2x_1)$ et $r(P_1) = (-2\ 2)$.

$d_N = \begin{pmatrix} 2 \\ 0 \end{pmatrix}$ et $d_B = -2$ donc la direction de déplacement est $\begin{pmatrix} -2 \\ 2 \\ 0 \end{pmatrix}$.

$$\begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} + \alpha \begin{pmatrix} -2 \\ 2 \\ 0 \end{pmatrix} \geqslant 0 \Rightarrow \begin{cases} \alpha \leqslant \frac{1}{2} \\ \alpha \leqslant 0 \end{cases} \Rightarrow \alpha_{\max} = \frac{1}{2}$$

Le minimum de $f(1 - 2\alpha, 2\alpha, 0)$ sous la contrainte $0 \leqslant \alpha \leqslant \frac{1}{2}$ est atteint en $\alpha_2 = \frac{1}{4}$.

Le nouveau point est $P_2 = \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \\ 0 \end{pmatrix}$.

Example (suite)

Itération 3.

Le gradient réduit en P_2 est $r(P_2) = (0 \quad 1)$ donc $d_N = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$. On arrête.

Vérifions que les conditions de Kuhn-Tucker sont satisfaites : ici
$A = (1 \quad 1 \quad -1)$
soient $\mu = -\frac{\partial f}{\partial x_1}(\frac{1}{2}, \frac{1}{2}, 0)A_1^{-1} = -2 \times \frac{1}{2} = -1$ (car $A_1 = 1$), $\lambda_1 = 0$,
$(\lambda_2 \quad \lambda_3) = r(P_2) = (0 \quad 1)$
$\frac{\partial f}{\partial x}(\frac{1}{2}, \frac{1}{2}, 0) + \mu A - \lambda = 0 \Leftrightarrow (2 \times \frac{1}{2} \quad 2 \times \frac{1}{2} \quad 0) - (1 \quad 1 \quad -1) - (0 \quad 0 \quad 0)$,
donc vérifié.
On aura remarqué que l'on a une écriture "transposée" des équations de Kuhn-Tucker.
$\lambda \geqslant 0$ et les conditions de complémentarités
$\lambda_1 \times \frac{1}{2} = 0, \lambda_2 \times \frac{1}{2} = 0, \lambda_3 \times 0 = 0$ sont vérifiées.

Remarque

en partant du même point on obtient des cheminements différents selon le choix du B initial.

Example (suite)

Si l'on choisit $B = \{1\}$ et $N = \{2, 3\}$, on passe par les 3 points

$$P_1 = \begin{pmatrix} \frac{2}{5} \\ \frac{4}{5} \\ \frac{1}{5} \end{pmatrix}, P_2 = \begin{pmatrix} \frac{2}{5} \\ \frac{3}{5} \\ 0 \end{pmatrix}, P_3 = \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \\ 0 \end{pmatrix}$$ sans changer de base B.

Organisation du cours

Une idée assez naturelle, pour adapter les méthodes d'optimisation sans contraintes aux problèmes avec contraintes, est de projeter à chaque étape le déplacement sur la frontière du domaine, afin de s'assurer que le nouveau point obtenu appartienne à l'ensemble des solutions réalisables. L'ensemble des contraintes linéaires forme alors un domaine de points réalisables qui s'écrit sous la forme

$$S = \{\mathbf{x} \in \mathbb{R}^n : \sum_{j=1}^{n} a_{ij} x_j - b_i \leqslant 0, \quad i = 1, 2, ..., m\},$$

où les a_{ij} ont été normalisés afin que

$$\|a_i\|^2 = \sum_{j=1}^{n} (a_{ij})^2 = 1 \qquad i = 1, 2, .., m.$$

Soit A_q la sous matrice de A constituée par les lignes des contraintes saturées en un point $\mathbf{x}$ de S,

$$A_q \equiv [a_1 \quad a_2 \quad \ldots a_q]^t.$$

Puisque les vecteur a_i sont supposés linéairement indépendants, la matrice symétrique $A_q A_q^t$ est non singulière. Par conséquent, son inverse $(A_q A_q^t)$ existe.
On définit une matrice symétrique $n \times n$ par

$$P_q = I - A_q^t (A_q A_q^t)^{-1} A_q.$$

En modifiant ainsi le moins possible la méthode d'origine, on peut espérer conserver son efficacité. De nombreux algorithmes basés sur ce schéma général ont été proposés. Un des premiers algorithmes bâtis suivant ce principe est la méthode du gradient projeté de Rosen qui consiste à projeter le gradient sur l'intersection des contraintes saturées.

La méthode du gradient projeté consiste à prendre comme direction de déplacement la projection du vecteur gradient sur la frontière du domaine et dont l'algorithme s'écrit sous la forme générale suivante :

(*i*) On part de l'initialisation : $\mathbf{x}^0 \in S$.

(ii) On génére la suite de vecteurs $\{\mathbf{x}^k\}_{k\geqslant 1} \in S$ définis par :

$$\forall k \geqslant 0 \quad \mathbf{x}^{k+1} = \mathbf{x}^k - \omega_k P_q \nabla f(\mathbf{x}^k), \tag{6}$$

où ω_k est le pas de déplacement.

Stratégie de pénalisation des contraintes

Soit le programme mathématique

$$\begin{cases} \text{minimiser la fonction : } f(\mathbf{x}) \\ \text{sous les contraintes : } g_i(\mathbf{x}) \leqslant b_i \qquad i = 1, \ldots m. \end{cases}$$

Une façon simple de se débarrasser des contraintes est d'intégrer dans la fonction objectif une fonction qui pénalise les vecteurs qui violent les contraintes, on a

$$\min_{\mathbf{x} \in \mathbb{R}^n} \quad f(\mathbf{x}) - \sum_{i=1}^{m} p_i(\mathbf{x})$$

où $p_i(\mathbf{x})$ prend une valeur élevée lorsque $g_i(\mathbf{x})$ est strictement positif, ce qui décourage l'adoption de ces vecteurs $\mathbf{x}$. On peut par exemple poser

$$p_i(\mathbf{x}) = \lambda \max\{0, g_i(\mathbf{x}) - b_i\}$$

où λ est un coefficient positif et grand. Si l'on souhaite que la fonction objectif soit différentiable, on peut élever l'expression précédente au carré. Pour résoudre le problème pénalisé, il suffit alors de trouver les points où le gradient de la fonction objectif s'annule.

Principe de pénalisation

Optimisation sous contraintes d'égalité

Le problème formel

$$\begin{cases} C = \{x \in V \text{ tel que } F_j(x) = 0\ ,\ j = 1, \ldots, p\} \\ \\ \forall x \in C \quad F(\bar{x}) \leqslant F(x) \end{cases} \tag{7}$$

Principe

On supprime les contraintes mais on punit leur non-respect

Principe de pénalisation

Optimisation sous contraintes d'égalité

Le problème formel

$$\begin{cases} C = \{x \in V \text{ tel que } F_j(x) = 0\ ,\ j = 1, \ldots, p\} \\ \\ \forall x \in C \quad F(\bar{x}) \leqslant F(x) \end{cases} \tag{7}$$

Principe

On supprime les contraintes mais on punit leur non-respect

Pénalisation

Définition (Problème pénalisé)

$$\begin{cases} F_\epsilon(x) = F(x) + \sum_i \dfrac{F_i(x)^2}{2\epsilon} \\ \forall x \in V \quad F_\epsilon(x_\epsilon) \leqslant F_\epsilon(x) \end{cases} \tag{8}$$

Remarques

ϵ doit être petit, mais : problème des erreurs de troncature.

Pénalisation

Définition (Problème pénalisé)

$$\begin{cases} F_\epsilon(x) = F(x) + \sum_i \dfrac{F_i(x)^2}{2\epsilon} \\ \forall x \in V \quad F_\epsilon(x_\epsilon) \leqslant F_\epsilon(x) \end{cases} \tag{8}$$

Remarques

ϵ doit être petit, mais : problème des erreurs de troncature.

Interprétation
Un problème de mécanique

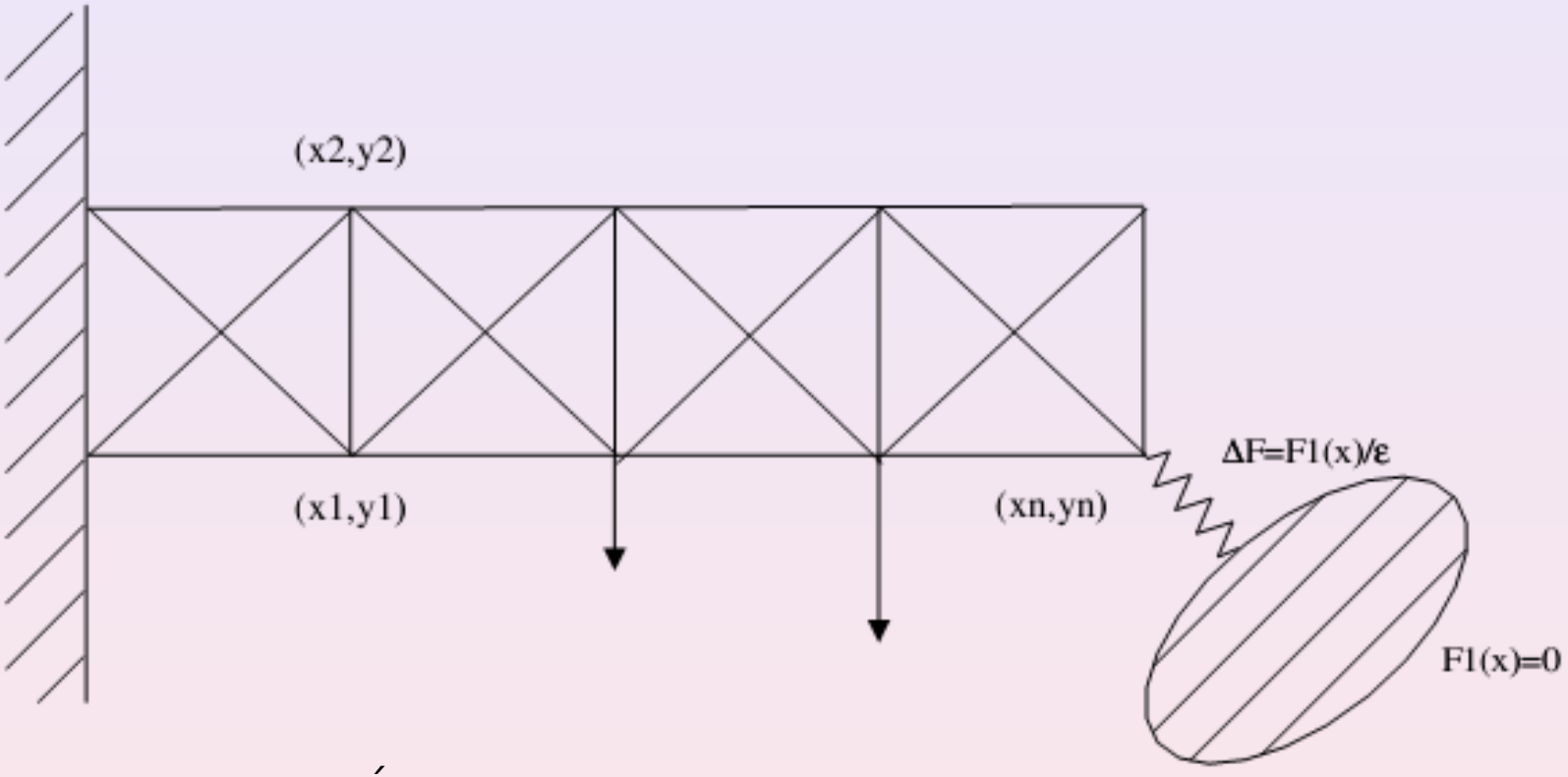

FIGURE : Équilibre d'une structure avec contacte unilatéral

Interprétation
Un problème de mécanique

Équilibre d'un treillis

Un treillis de barres, avec contact unilatéral.
La position d'équilibre pénalisée minimise

$$\min_x \frac{1}{2}\langle \mathbf{K}x, x\rangle - \langle b, x\rangle + \frac{F_1(x)^2}{2\epsilon}$$

qui est l'énergie potentielle + l'énergie interne du ressort de liaison.

Intérêt
Pénalisation

Théorème

La solution du problème pénalisé tend vers la solution du problème avec contraintes quand $\epsilon \to 0$.
L'erreur est en $\sqrt{\epsilon}$

Difficulté

Le problème pénalisé est mal conditionné, la précision est médiocre

Intérêt
Pénalisation

Théorème

La solution du problème pénalisé tend vers la solution du problème avec contraintes quand $\epsilon \to 0$.
L'erreur est en $\sqrt{\epsilon}$

Difficulté

Le problème pénalisé est mal conditionné, la précision est médiocre