Master 1 MINT et MMM, Université de Versailles Saint Quentin en Yveline

Optimisation : Première Contribution

Arjeta HETA
Jérémy PIERRE
Sarah TACHET

Table des matières

0.1 Rappels sur la factorisation de matrices

Factorisation de matrices : Soit A une matrice a m lignes et n colonnes avec $m \leq n$.

Nous allons définir trois types de factorisations :

La factorisation LU : elle permet de réduire le problème (variables dépendantes et indépendantes) et construire une base du noyau.

La factorisation QR : elle permet de réduire le problème (variables dépendantes et indépendantes) et construire une base orthogonale du noyau.

La factorisation LL^T pour une matrice définie positive ou LDL^T pour rendre le Hessien défini positif.

Si la factorisation d'une matrice peut s'avérer coûteuse (en $\mathcal{O}(n^3)$, das les trois cas et pour une matrice carrée), elle permet ensuite la résolution rapide (en $\mathcal{O}(n^2)$) du système linéaire $Ax = b$ pour n'importe quel second membre b. Ce procédé est donc particulièrement intéressant si l'on est amené à inverser des systèmes linéaires dont seul le second membre varie à chaque itération d'un algorithme.

Factorisation LU : Pour toute matrice $A = (a_{i,j}) \in M_n(\mathbb{R})$ et tout entier $k \in 1, ..., n$, on note A_k la sous-matrice $(a_{i,j})_{1 \leq i,j \leq k} \in M_k(\mathbb{R})$.

Définition 1 *Une matrice $A \in M_n(\mathbb{R})$ est complètement régulière si $A_k \in GL_k(\mathbb{R})$ pour tout $1 \leq k \leq n$. On notera $\Lambda_n(\mathbb{R})$ l'ensemble de ces matrices.*

Théorème 1 *Pour toute matrice complètement régulière $A \in \Lambda_n(\mathbb{R})$, il existe un unique couple $(L, U) \in M_n(\mathbb{R}) \times M_n(\mathbb{R})$ tel que $A = LU$, avec L triangulaire inférieure de coefficients diagonaux égaux à 1 et U triangulaire supérieure inversible.*

Résolution d'un système linéaire : Étant donnée une telle factorisation, on peut résoudre $Ax = b$ pour n'importe quel second membre b, en résolvant les deux systèmes triangulaires $Ly = b$ (descente L) puis $Ux = y$ (remontée U).

Exemple : En pratique, pour obtenir une factorisation, on peut déterminer de proche en proche les coefficients inconnus des matrices L et U ou alors résoudre un système. Par exemple, si l'on prend $A = \begin{pmatrix} -1 & 1 & 2 \\ 1 & 1 & -3 \\ 1 & 1 & -4 \end{pmatrix}$,

$$
\begin{aligned}
A = \begin{pmatrix} 1 & 0 & 0 \\ ? & 1 & 0 \\ ? & ? & 1 \end{pmatrix} \begin{pmatrix} \boxed{?} & \boxed{?} & \boxed{?} \\ 0 & ? & ? \\ 0 & 0 & ? \end{pmatrix} &\xRightarrow{L_{1,:}U=A_{1,:}} U = \begin{pmatrix} -1 & 1 & 2 \\ 0 & ? & ? \\ 0 & 0 & ? \end{pmatrix} \\
A = \begin{pmatrix} 1 & 0 & 0 \\ \boxed{?} & 1 & 0 \\ \boxed{?} & ? & 1 \end{pmatrix} \begin{pmatrix} -1 & 1 & 2 \\ 0 & ? & ? \\ 0 & 0 & ? \end{pmatrix} &\xRightarrow{LU_{:,1}=A_{:,1}} L = \begin{pmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ -1 & ? & 1 \end{pmatrix} \\
A = \begin{pmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ -1 & ? & 1 \end{pmatrix} \begin{pmatrix} -1 & 1 & 2 \\ 0 & \boxed{?} & \boxed{?} \\ 0 & 0 & ? \end{pmatrix} &\xRightarrow{L_{2,:}U=A_{2,:}} U = \begin{pmatrix} -1 & 1 & 2 \\ 0 & 2 & -1 \\ 0 & 0 & ? \end{pmatrix} \\
&\dots
\end{aligned}
\tag{1}
$$

Factorisation QR : La factorisation QR permet de résoudre un système linéaire $Ax = b$ en résolvant $Qy = b$ puis $Rx = y$. On veut ici que Q soit orthogonale et R triangulaire : la résolution du premier système est donc particulièrement simple puisque $y = Q^T b$. Le second est également facile à résoudre, car R est triangulaire. Pour la factorisation QR, les choses sont simples : toutes les matrices inversibles admettent une unique factorisation.

Théorème 2 *Pour toute matrice inversible $A \in GL_n(\mathbb{R})$, il existe un unique couple de matrices $(Q, R) \in GLn(\mathbb{R}) \times GLn(\mathbb{R})$ tel que $A = QR$, où Q est orthogonale et R triangulaire supérieure de coefficients diagonaux > 0.*

Méthode de Householder

Lemme 1 *Pour tout vecteur non nul $u = (u_i) \in M_{n,1}(\mathbb{R})$, on a :*

$$
H(u - \sigma||u||e_1)u = \sigma||u||e_1, \text{ en notant } \sigma = \begin{cases} -1 & \text{si } u_1 \geq 0 \\ 1 & \text{sinon} \end{cases}
$$

Lemme 2 (Prolongement) *Soient $1 \leq k < n$ avec n, k deux entiers et $w = (w_i) \in M_{n-k,1}(\mathbb{R})$ un vecteur. On note $v = (v_i) \in M_{n,1}(\mathbb{R})$ le vecteur défini par :*

$$
v_i = \begin{cases} 0 & \text{si } 1 \leq i \leq k \\ w_{i-k} & \text{si } k < i \leq n \end{cases}
$$

Alors les matrices $H(v) \in M_n(\mathbb{R})$ et $H(w) \in M_{n-k}(\mathbb{R})$ vérifient la relation :

$$
H(v) = \begin{pmatrix} I_k & 0 \\ 0 & H(w) \end{pmatrix}
$$

Matrices de Householder : Pour tout $v \in M_n, 1(\mathbb{R})$ non nul, on notera :

$$H(v) = I_n - 2\frac{vv^T}{v^T v} \in M_n(\mathbb{R}).$$

Une telle matrice est appelée matrice de Householder. Il est aisé de vérifier qu'une telle matrice est à la fois symétrique et orthogonale.

Le principe de cette méthode consiste à trianguliariser une matrice $A \in GL_n(\mathbb{R})$ en la multipliant par des matrices de Householder $H_1, ..., H_{n-1}$:

$$\underbrace{\begin{pmatrix} . & . & . & . & . \\ . & . & . & . & . \\ . & . & . & . & . \\ . & . & . & . & . \\ . & . & . & . & . \end{pmatrix}}_{A} \longrightarrow \underbrace{\begin{pmatrix} . & . & . & . & . \\ 0 & . & . & . & . \\ 0 & . & . & . & . \\ 0 & . & . & . & . \\ 0 & . & . & . & . \end{pmatrix}}_{H_1 A} \longrightarrow \underbrace{\begin{pmatrix} . & . & . & . & . \\ 0 & . & . & . & . \\ 0 & 0 & . & . & . \\ 0 & 0 & . & . & . \\ 0 & 0 & . & . & . \end{pmatrix}}_{H_2 H_1 A} \longrightarrow, ..., \longrightarrow$$

$$\underbrace{\begin{pmatrix} . & . & . & . & . \\ 0 & . & . & . & . \\ 0 & 0 & . & . & . \\ 0 & 0 & 0 & . & . \\ 0 & 0 & 0 & 0 & . \end{pmatrix}}_{Hn-1,...,H_1 A}$$

On procède en $n-1$ étapes, qui chacune traitent une colonne. Le produit final est une matrice triangulaire $H_{n-1}, ..., H_1 A = R$. Comme les matrices $H_1, ..., H_{n-1}$ sont orthogonales et symétriques, on obtient un produit $A = (H_1, ..., H_{n-1})R$ donnant une factorisation QR au signe des coefficients diagonaux de R près. Détaillons la $(k+1)$-ième étape. Après la k-ième étape, on obtient une matrice :

$$H_k, .., H_1 A = \begin{pmatrix} T_k & B_k \\ 0 & C_k \end{pmatrix}$$

avec $T_k \in T_k(\mathbb{R})$.On sait que $C_k \in GL_k(\mathbb{R})$donc la première colonne c de la matrice C_k est non nulle. Le choix

$$w = c - \sigma||c||e_1 \in M_{n-k,1}(\mathbb{R})$$

assure que $H(w)c = \sigma||c||e_1 \in M_{n-k,1}(\mathbb{R})$ est la première colonne de $H(w)C_k \in M_{n-k}(\mathbb{R})$. Posons maintenant $H_{k+1} = H(v) \in M_n(\mathbb{R})$. Il vient :

$$H_{k+1}H_k, .., H_1 A = \begin{pmatrix} I_k & 0 \\ 0 & H(w) \end{pmatrix} H_k, .., H_1 A = \begin{pmatrix} I_k & 0 \\ 0 & H(w) \end{pmatrix} \begin{pmatrix} I_k & B_k \\ 0 & C_k \end{pmatrix} = \begin{pmatrix} I_k & B_k \\ 0 & H(w)C_k \end{pmatrix} \quad (2)$$

$$H_{k+1}H_k, .., H_1A = \begin{pmatrix} T_{k+1} & B_{k+1} \\ 0 & C_{k+1} \end{pmatrix} \tag{3}$$

avec $T_{k+1} \in T_{k+1}(\mathbb{R})$.

On peut énoncer :

Théorème 3 *Pour toute matrice inversible $A \in GL_n(\mathbb{R})$, il existe $n-1$ vecteurs non nuls $v_1, ..., v_{n-1} \in M_{n,1}(\mathbb{R})$ tels que $(H(v_1), ..., H(v_{n-1}))A$ est triangulaire supérieure.*

Exemple : Soit A la matrice définie tel que :

$$A = \begin{pmatrix} 0 & 1 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

Étape 1 : La première colonne de A est le vecteur e_3. Calculons la matrice de Householder $H_1 \in M_3(\mathbb{R})$ engendrée par le vecteur $e_1 - e_3$:

$$H_1 = H(e_1 - e_3) = I_3 - \tfrac{2}{2}(e_1 - e_3)^T = I_3 - \begin{pmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

On trouve alors :

$$H_1A = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}$$

En reprenant les notations ci-dessus, cette matrice s'écrit par blocs :

$$\begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix} = \begin{pmatrix} T_1 & B_1 \\ 0 & C_1 \end{pmatrix}$$

Étape 2 : Le premier vecteur de la matrice C_1 est le vecteur e_2. Calculons la matrice de Householder $H_2' \in M_2(\mathbb{R})$ engendrée par le vecteur $e_1 - e_2$:

$$H_2' = H(e_1 - e_2) = I_2 - \tfrac{2}{2}(e_1 - e_2)(e_1 - e_2)^T = I_2 - \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

On construit la matrice

$$H_2 = \begin{pmatrix} 1 & 0 \\ 0 & H_2' \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

Le produit donne une matrice triangulaire supérieure

$$H_2 H_1 A = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}$$

Ceci s'écrit encore

$$A = H_1 H_2 \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} = QR \quad (4)$$

Factorisation LL^T

Définition 2 *Une matrice symétrique $A \in S_n(\mathbb{R})$ est définie positive lorsqu'elle vérifie $\forall x \in M_{n,1}(\mathbb{R})$ privé du vecteur nul, $x^T.A.x > 0$.*

Lemme 3 *Si $A \in M_n(\mathbb{R})$ est symétrique définie positive alors $A \in \Lambda_n(\mathbb{R})$.*

Théorème 4 (Cholesky) *Une matrice $A = (a_{i,j}) \in M_n(\mathbb{R})$ est une matrice symétrique définie positive si et seulement s'il existe $L \in M_n(\mathbb{R})$ triangulaire inférieure à coefficients diagonaux > 0 telle que $A = LL^T$. Dans ce cas, la matrice L est unique.*

0.2 Conditionnement

0.2.1 conditionnement d'une matrice

Définition 3 *Soit A une matrice symétrique définie positive. On définit alors le conditionnement par :*

$$\kappa(A) = ||A||_2 ||A^{-1}||_2$$

Propriété 1 *Si on note $\sigma_1 \geq ... \geq \sigma_n > 0$. les valeurs propres de A, on a*

$$\kappa(A) = \frac{\sigma_1}{\sigma_n} \geq 1$$

Démonstration : Par définition de la norme nous avons :

$$\begin{aligned} ||A||_2 &= \sup_{v\in\mathbb{R}^n, v\neq 0} \frac{||Av||_2}{||v||_2} \\ &= \max_{1\leq i\leq n} \sup_{v\in E_i, v\neq 0} \frac{||\sigma_i v||_2}{||v||_2} \\ &= \max_{1\leq i\leq n} \sigma_i \\ &= \sigma_1 \end{aligned}$$

où on note E_i l'espace propre associé à la i-ème valeur propre de A. Comme A est diagonalisable en tant que matrice symétrique définie positive, $\mathbb{R}^n$ est la somme directe de ses sous-espaces propres E_i, ce qui justifie la première étape. Pour la même raison, il existe P inversible telle que $A = PDP^{-1}$ où D est une matrice diagonale possédant les valeurs propres de A. Partant $A^{-1} = PD^{-1}P^{-1}$ et les valeurs propres de A^{-1} sont les inverses de celle de A :

$$\begin{aligned} ||A^{-1}||_2 &= \max_{1\leq i\leq n} \frac{1}{\sigma_i} \\ &= \frac{1}{\sigma_n} \end{aligned}$$

Rappel : Soit la matrice A tel que :

$$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

L'inverse de A s'écrit alors :

$$A^{-1} = \frac{1}{det(A)} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$$

avec $det(A) = ad - bc \neq 0$.
Pour obtenir les valeurs propres on résout :
$det(A - \sigma I) = 0 \Rightarrow (a-\sigma)(d-\sigma) - bc = 0 \Rightarrow \sigma^2 - (a+d)\sigma + det(A) = 0$

Exemple : Soit la matrice A telle que :

$$A = \begin{pmatrix} 0.1 & 1 \\ 0.20002 & 2 \end{pmatrix}$$

Les valeurs propres sont :

$$\sigma^2 - 2.1\sigma - 0.00002 = 0 \Rightarrow \left\{ \begin{array}{l} \sigma_1 = 2.10001 \\ \sigma_2 = -0.00001 \end{array} \right.$$

Le conditionnement :

$$\left\{ \begin{array}{l} ||A||_2 = \sigma_1 \\ ||A^{-1}||_2 = \frac{1}{\sigma_2} \end{array} \right. \Rightarrow \kappa(A) = \frac{\sigma_1}{\sigma_2} = 210001.$$

Intérêt du conditionnement : Le conditionnement quantifie la sensibilité de la solution d'un système linéaire à des perturbations de la matrice ou du second membre. La portée de cette étude est loin d'être uniquement abstraite, puisque l'algorithmique aboutissant *in fine* à une implémentation informatique, l'on doit s'accommoder à d'erreurs d'arrondi, qui s'ajoutent, si le problème sous la main est issu du monde physique, à des erreurs de mesure. On considère donc le problème exact

$$\text{Trouver } x \in \mathbb{R}^n,\ Ax = b\ (P)$$

de solution x^*. Soient, maintenant, les problèmes perturbés au premier membre, respectivement second membre :

$$\text{Trouver } x \in \mathbb{R}^n,\ (A + \Delta A)x = b \quad \text{de solution } x^* + \Delta x_A\ (P_1)$$

$$\text{Trouver } x \in \mathbb{R}^n,\ Ax = b + \Delta b \quad \text{de solution } x^* + \Delta x_b\ (P_2)$$

Commençons par exprimer les solutions :

$$\left\{ \begin{array}{l} Ax^* = b \\ A(x^* + \Delta x_b) = b + \Delta b \\ (A + \Delta A)(x^* + \Delta x_A) = b \end{array} \right. \Rightarrow \left\{ \begin{array}{l} Ax^* = b \\ A\Delta x_b = \Delta b \\ A\Delta x_A + \Delta A x^* = 0 \end{array} \right. \Rightarrow \left\{ \begin{array}{l} b = Ax^* \\ \Delta x_b = A^{-1}\Delta b \\ \Delta x_A = -A^{-1}\Delta A x^* \end{array} \right. \tag{5}$$

Voyons à présent par quoi l'erreur sur la solution (Δx_A, respectivement Δx_b) est contrôlée :

$$\left\{ \begin{array}{l} \Delta x_b = A^{-1}\Delta b \\ \Delta x_A = -A^{-1}\Delta A x^* \end{array} \right. \Rightarrow \left\{ \begin{array}{l} ||\Delta x_b|| \leq ||A^{-1}||||\Delta b|| \\ ||\Delta x_A|| \leq ||A^{-1}||||\Delta A||||x^*|| \end{array} \right. \Rightarrow \left\{ \begin{array}{l} \frac{||\Delta x_b||}{||x^*||} \leq ||A||||A^{-1}||\frac{||\Delta b||}{||b||} \\ \frac{||\Delta x_A||}{||x^*||} \leq ||A||||A^{-1}||\frac{||\Delta A||}{||A||} \end{array} \right. \tag{6}$$

car

$$b = Ax^* \Rightarrow ||b|| \leq ||A||||x^*|| \Rightarrow \frac{1}{||x^*||} \leq \frac{||A||}{||b||}$$

La perturbation est donc, au plus, amplifiée de

$$\kappa(A) = ||A||||A^{-1}||$$

et ce dans les deux cas. Remarquons que la majoration ne saurait être meilleure car l'inégalité

$$Ax = b \Rightarrow ||b|| \leq |||A|||.||x||$$

peut être atteinte, A définissant une appliquation linéaire sur un espace de dimension finie (elle est donc continue et la boule est compacte) et le sup définissant $|||A|||$ est donc un max. Aucune autre majoration que celle-ci n'ayant été employée, l'inégalité finale peut être atteinte.

Ceci justifie que l'on parle communément de matrice *mal conditionnée* lorsque son nombre de conditionnement est grand, puisque l'issue d'une résolution algorithmique d'un système linéaire la concernant est alors susceptible d'être très lointaine de la solution réelle recherchée. L'exemple suivant illustre l'effet dramatique du mauvais conditionnement.

Exemple : Le système linéaire

$$\begin{pmatrix} 0.1 & 1 \\ 0.20002 & 2 \end{pmatrix} x = \begin{pmatrix} 2 \\ 4.0002 \end{pmatrix}$$

a pour solution le vecteur $(10, 1)^T$. Si l'on perturbe la matrice d'un facteur de l'ordre de 10^{-3}, de sorte à résoudre

$$\begin{pmatrix} 0.101 & 1 \\ 0.20002 & 2 \end{pmatrix} x = \begin{pmatrix} 2 \\ 4.0002 \end{pmatrix}$$

on obtient, cette fois, la solution $(-0.101, 2, 010)^T$ en aucun cas proche de la solution réelle. De la même façon, si l'on perturbe le second membre,

$$\begin{pmatrix} 0.1 & 1 \\ 0.20002 & 2 \end{pmatrix} x = \begin{pmatrix} 2.01 \\ 4.0002 \end{pmatrix}$$

la solution devient $(-990, 101.01)^T$! Il est donc clair que les perturbations sur la solution sont sans commune mesure avec les perturbations observées sur la matrice ou sur le second membre.

0.2.2 Conditionnement d'une fonction

En optimisation (en particulier, quadratique), on est fréquemment amené à résoudre des systèmes linéaires faisant intervenir la hessienne du critère. On définit donc le conditionnement de f deux fois différentiable en x comme le nombre de conditionnement de sa Hessienne notée $H(x)$.

Interprétation : Soit $\sigma_1 \geq ... \geq \sigma_k > 0.$ les valeurs propres de A, et $d_1 \geq ... \geq d_k > 0.$ les vecteurs propres associés aux valeurs propres. Alors on peut écrire $A_k d_k = \sigma_k d_k$.

La courbure de la fonction f dans la direction de d_k est :

$$\frac{d_k^T H(x) d_k}{d_k^T d_k} = \sigma_k.$$

Le Théorème de Rayleigh-Ritz nous assure que :
d_1 est la direction de plus forte courbure ce qui correspond a la valeur propre σ_1 et d_n correspond à la valeur singulière de plus faible courbure ce qui correspond à la valeur propre σ_n.

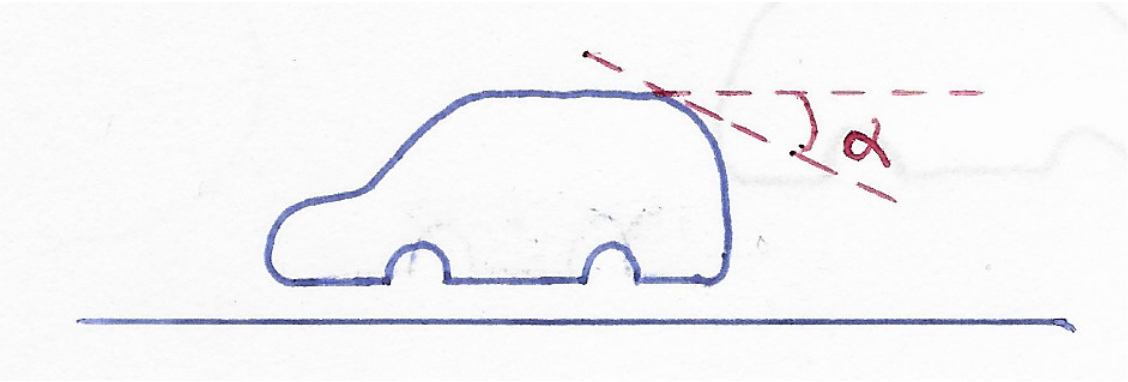
Chapitre 1

Introduction et Rappels

1.1 Quelques exemples de problèmes d'optimisation en ingénierie

Exemple 1 : Un objet se déplaçant dans un fluide subit de ce-dernier une pression et un frottement s'opposant à la marche de cet objet. On s'interroge donc sur la forme d'un véhicule vis à vis d'un critère aérodynamique : On cherche à créer une forme telle que la traînée ou la résistance au vent du véhicule soit la plus faible possible. Le paramètre d'importance dans ce problème est l'angle α formé entre l'horizontal et la forme de la partie arrière de la voiture.

- Paramètre : $\alpha \in [0; \frac{\pi}{2}]$
- Critère : $f(\alpha) = C_x$ (appelé "drag coefficient" ou coefficient de traînée)
- Contraintes : géométrique (design de la carroserie) ; aérodynamique ; ...

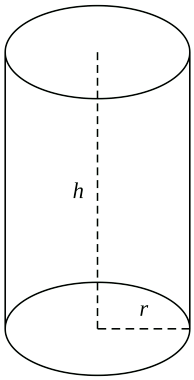


Exemple 2 : Un autre problème courant en ingénierie automobile est la forme d'une pièce (exemple : le capot d'une voiture) vis à vis de critères mécaniques (exemple : la rigidité de la pièce). Pour notre exemple, la forme et la rigidité du capot doivent être telles que toute déformation puissent minimiser le risque de blessures lors de chocs. Ce problème fait intervenir plusieurs paramètres qui doivent être testés à travers des simulations numériques.

- Paramètres : Déterminés par Conception Assistée par Ordinateur (CAO)
- Critère : La compliance, c'est-à-dire la souplesse ou la flexibilité mécaniques (calculée par une méthode de type éléments finis)
- Contraintes : voir *exemple 1*

Exemple 3 : Le problème de la canette : Le but est de produire une canette pouvant contenir un volume de $33cl$ en utilisant le moins de métal possible. La forme de la cannette est simplifiée par un cylindre de hauteur h et de rayon r. Comme le volume est constant, le problème revient à minimiser la surface totale du cylindre exprimée selon r et h sous certaines contraintes. Le problème peut être résumé de la manière suivante :

- Paramètres : $h > 0$ et $r > 0$
- Critère : On doit minimiser $f(r, h) = 2\pi r^2 + 2\pi rh$
- Contraintes : $\pi r^2 h \leq 330cm^3$ et $h, r \in [h_0, h_1] \times [r_0; r_1]$ (design de la canette)



1.2 Définitions

1.2.1 Problèmes d'optimisation

On cherche à résoudre le problème d'optimisation (PO) suivant :

$$\min_{x\in\mathbb{R}} f(x)\ avec \begin{cases} C_I(x) \leq 0 \\ C_E(x) = 0 \\ x \in X \end{cases}$$

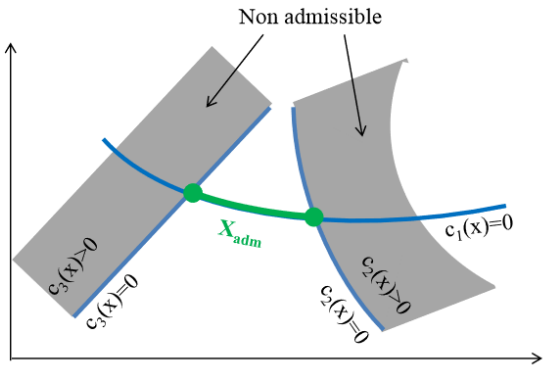
Où f (appelé critère, fonction coût ou fonction objectif) est une fonction de $\mathbb{R}^n$ dans $\mathbb{R}$ ($n \geq 1$ est le nombre de variables ou paramètres du problème).

$C_I : \mathbb{R}^n \longrightarrow \mathbb{R}^q$: contraintes d'inégalités (au nombre de $q \geq 0$).
$C_E : \mathbb{R}^n \longrightarrow \mathbb{R}^p$: contraintes d'égalités (au nombre de $p \geq 0$).
$X \subset \mathbb{R}^n$ un sous-ensemble convexe de $\mathbb{R}^n$ constitué des valeurs admissibles de variables.

1.2.2 Solutions

x est une solution admissible (ou point admissible) de (PO) si et seulement si x satisfait toutes les contraintes suivantes : $\begin{cases} C_E(x) = 0 \\ C_I(x) \leq 0 \\ x \in X \end{cases}$

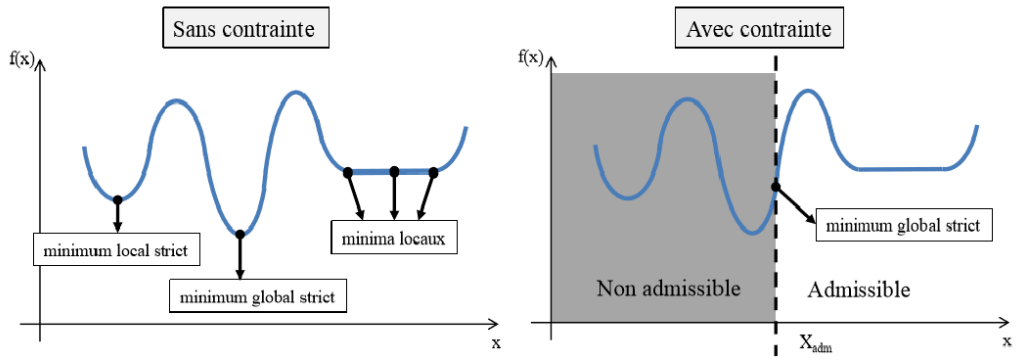
Exemple dans $\mathbb{R}^2$: $C_1(x_1, x_2) = 0 \rightarrow$ courbe.
$C_2(x_1, x_2) \leq 0 \rightarrow$ région du plan.
$C_3(x_1, x_2) \leq 0 \rightarrow$ région du plan.
$X_{adm} = \{x \in \mathbb{R}^n, C_1(x) = 0, C_2(x) < 0,\ et\ C_3(x) < 0\}$.



Exemple dans $\mathbb{R}^n$: $C_E(x) = 0 \rightarrow$ hypersurface de dimension (n-1).
$a^T x = 0 \rightarrow$ hyperplan perpendiculaire à $a \in \mathbb{R}^n$.

Notion de minimum global : $x^* \in X_{adm}$ est un minimum global de (PO) si et seulement si :
$\forall x \in X_{adm}, f(x^*) \leq f(x)$.
$x^* \in X_{adm}$ est un minimum global strict si et seulement si :
$\forall x \in X_{adm} \backslash \{x^*\}, f(x^*) < f(x)$.

Notion de minimum local : $x^* \in X_{adm}$ est un minimum local de (PO) si et seulement si :
$\exists \varepsilon > 0, \forall x \in X_{adm}, ||x - x^*|| \leq \varepsilon \Longrightarrow f(x^*) \leq f(x)$.
$x^* \in X_{adm}$ est un minimum local strict si et seulement si :
$\exists \varepsilon > 0, \forall x \in X_{adm} \backslash \{x^*\}, f(x^*) < f(x)$.



Notion de contrainte active : Une contrainte du problème (PO) est active (ou saturée) en x si et seulement si elle est nulle en ce point (cela concerne surtout les contraintes inégalités).
L'ensemble des contraintes actives s'écrit :
$C_{act} = \{j / C_{Ej}(x) = 0\} \cup \{j / C_{Ij}(x) = 0\}$.

- Contrainte égalité C_E : x admissible $\Longrightarrow C_E(x) = 0 \Longrightarrow C_E$ active en x.
- Contrainte inégalité C_I : x admissible $\Longrightarrow C_I(x) \leq 0 \Longrightarrow C_I$ active en x si $C_I(x) = 0$ et inactive en x si $C_I(x) < 0$.

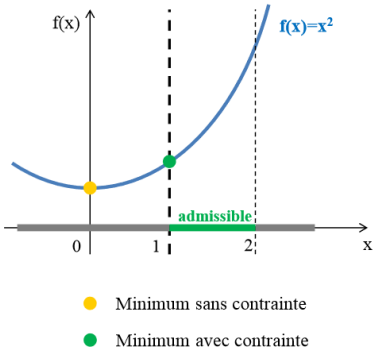
Les contraintes inactives n'ont pas d'influence sur la solution x^* du problème (PO). On peur donc les ignorer si on identifie l'ensemble $C_{act}(x^*)$.

Le problème (PO) est équivalent au problème réduit aux contraintes actives prises comme contraintes égalité.

$$\min_{x\in\mathbb{R}} f(x) \; sous \begin{cases} C_I(x) \leq 0 \\ C_E(x) = 0 \end{cases} \Leftrightarrow \min_{x\in\mathbb{R}} f(x) \; sous \; C_j(x) = 0, \; j \in C_{act}(x^*)$$

Exemple : $\min_{x\in\mathbb{R}} x^2 + 1$ sous $\begin{cases} x \geq 1 \\ x \leq 2 \end{cases} \rightarrow x^* = 1$

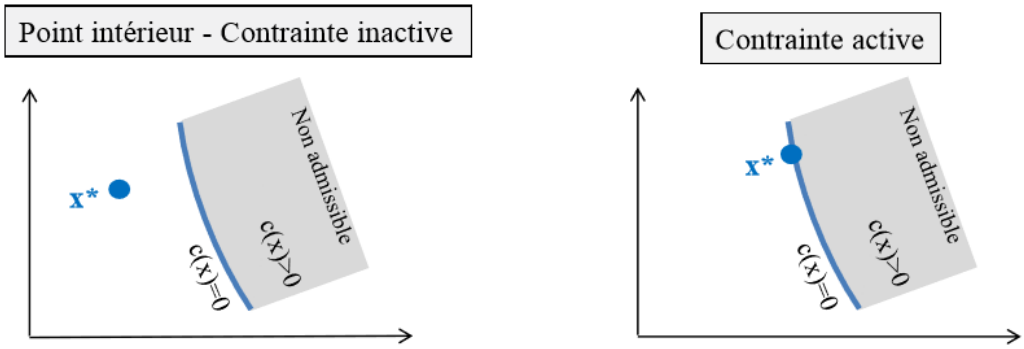
1. Minimum sans contrainte :
   $\min_{x\in\mathbb{R}} x^2 + 1 \rightarrow x^* = 0$
   La contrainte $x \leq 2$ est respectée mais pas la contrainte $x \geq 1$.
2. Minimum avec contrainte active $x = 1$:
   $min_{x\in\mathbb{R}} x^2 + 1 \rightarrow x^* = 1$
   Les deux contraintes sont respectées, $x^* = 1$ est donc solution du problème.
3. Conclusion :
   La contrainte $x \geq 1$ est active (elle est donc tranfsormée en égalité) tandis que la contrainte $x \leq 2$ est inactive (elle est donc ignorée).



Notion de point intérieur : $x \in X_{adm}$ est un point intérieur si :
$\exists \varepsilon > 0, \forall y \in \mathbb{R}^n, ||y - x|| \leq \varepsilon \Longrightarrow y \in X_{adm}$.
Un problème avec contraintes inégalité n'admet pas de point intérieur.

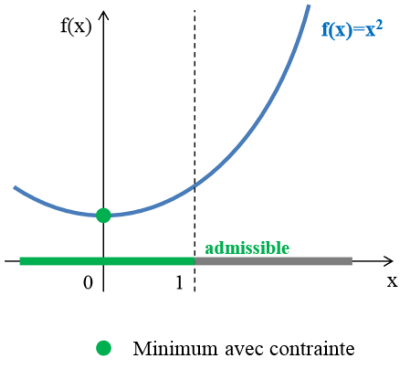
Solution intérieure aux contraintes : Soit x^* un minimum local du problème avec contraintes inégalités $min_{x\in\mathbb{R}} f(x)$ sous $C_I(x) \leq 0$.
Si x^* est un point intérieur, alors x^* est un minimum local du problème sans

contraintes $\min_{x \in \mathbb{R}} f(x)$ (plus simple).



Exemple : $\min_{x \in \mathbb{R}} x^2 + 1 \rightarrow x^* = 0$

1. Ensemble admissible :
   $X_{adm} = \{x \in \mathbb{R} / x \leq 1\} =]-\infty, 1]$
2. Ensemble intérieur à la contrainte :
   $X_{int} = X_{adm} - \{1\} =]-\infty, 1[$
   $x \in X_{int} \rightarrow$ voisinage de x inclus dans X_{int} (intervalle ouvert).
3. Solution :
   $x^* = 0$
   $x \in X_{int}$ intérieur à la contrainte $\rightarrow$ contrainte inactive.



1.2.3 Différentiabilité

Soit la fonction $f : \mathbb{R}^n \rightarrow \mathbb{R}$

Dérivée directionnelle : Soit $d \in \mathbb{R}^n$ une direction.
Si la limite $f_d(x) = \lim_{s \to 0} \frac{f(x+sd)-f(x)}{s}$ existe, alors f admet une dérivée directionnelle f_d, dans la direction d.
$f_d(x) = g(x)^t.d$ où $g(x)$ est le gradient de f.

Fonction différentiable : La fonction f est différentiable en x si et seulement si f admet une dérivée directionnelle pour tout $d \in \mathbb{R}^n$.

Dérivée partielle : Si la limite $f_{x_i}(x) = lim \frac{f(x_1,...,x_i+s,...,x_n)-f(x_1,...,x_i,...,x_n)}{s}$ existe, alors
$f_{x_i}(x) = \frac{\partial f(x)}{\partial x_i}$ est la dérivée partielle de f en x par rapport à x_i.

Gradient : On suppose que f est différentiable, c'est-à-dire :
$\forall x \in \mathbb{R}^n$, $\exists g(x) \in \mathbb{R}^n$ telle que $\forall d \in \mathbb{R}^n$:

$$f(x+d) = f(x) + g(x)^t.d + \varepsilon(||d||)$$

où $\lim_{h \to 0} \frac{\varepsilon(h)}{h} = 0$.
On dit que $g(x)$ est le gradient de ff en x.
Si toutes les dérivées partielles de f existent, alors $g(x) = \nabla f(x)$, où :

$$g : \mathbb{R}^n \to \mathbb{R}^n$$

$$g(x) = \begin{pmatrix} \frac{\partial f(x)}{\partial x_1} \\ . \\ . \\ . \\ \frac{\partial f(x)}{\partial x_n} \end{pmatrix}$$

Hessien : Soit f une fonction C^2 de $\mathbb{R}^n$ dans $\mathbb{R}$.
Il existe $M \in \mathcal{S}_n(\mathbb{R})$ tel que pour tout $d \in \mathbb{R}^n$,

$$f(x+d) = f(x) + g(x)^t.d + \tfrac{1}{2} d^t.M.d + \varepsilon(||d||)$$

où $\lim_{h \to 0} \frac{\varepsilon(h)}{h^2} = 0$.
Alors la matrice $M = H(x) = \nabla^2 f(x)$, $H(x) : \mathbb{R}^n \to \mathbb{R}^{n \times n}$

$$H(x) = \begin{pmatrix} \frac{\partial^2 f(x)}{\partial^2 x_1} & ... & \frac{\partial^2 f(x)}{\partial x_1 \partial x_n} \\ . & ... & . \\ . & ... & . \\ . & ... & . \\ \frac{\partial^2 f(x)}{\partial x_n \partial x_1} & ... & \frac{\partial^2 f(x)}{\partial^2 x_n} \end{pmatrix}$$

$H(x)$ est appelé le Hessien de f en x.

Modèle quadratique-linéaire : La fonction $f_0^*(x)$ définie par :

$$f_0^*(x) = f(x_0) + \nabla f(x_0)^t.(x - x_0) + \frac{1}{2}(x - x_0)^t.\nabla^2 f(x_0).(x - x_0)$$

est appelé modèle quadratique linéaire de f en x_0.
On peut définir les contraintes actives par un modèle linéaire linéaire :

$$c_0^* = c_0(x_0) + \nabla c(x_0)^t.(x - x_0)$$

Le problème quadratique-linéaire ainsi défini est localement "proche" du problème initial et est plus simple à résoudre.

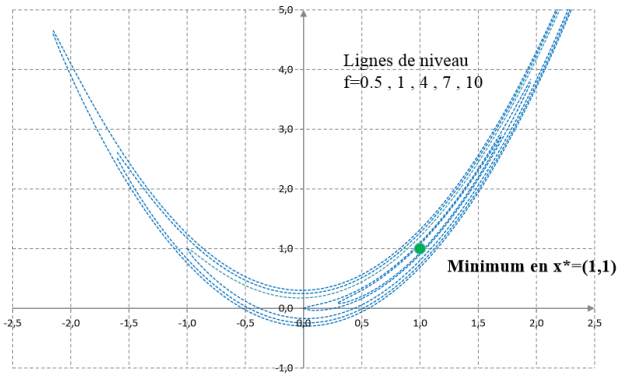
Exemple : Soit la fonction de Rosenbrock C^2 sur $\mathbb{R}^2$ définie par :
$f(x_1, x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$

1. Gradient :
$$\nabla f(x) = \begin{pmatrix} -400(x_2 - x_1^2)x_1 - 2(1 - x_1) \\ 200(x_2 - x_1^2) \end{pmatrix}$$
2. Hessien :
$$\nabla^2 f(x) = \begin{pmatrix} -400(x_2 - 3x_1^2) + 2 & -400x_1 \\ -400x_1 & 200 \end{pmatrix}$$
3. Modèle quadratique en $x_0 = (1, 1)$:
$f(x_0) = 0$, $\nabla f(x_0) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$, $\nabla^2 f(x_0) = \begin{pmatrix} 802 & -400 \\ -400 & 200 \end{pmatrix}$
D'où la forme quadratique suivante :
$f_0^*(x_1, x_2) = \frac{1}{2} \begin{pmatrix} x_1 - 1 \\ x_2 - 1 \end{pmatrix}^t \begin{pmatrix} 802 & -400 \\ -400 & 200 \end{pmatrix} \begin{pmatrix} x_1 - 1 \\ x_2 - 1 \end{pmatrix}$
$f_0^*(x_1, x_2) = 401(x_1 - 1)^2 - 400(x_1 - 1)(x_2 - 1) + 100(x_2 - 1)^2$

Lignes de niveau : Soit la fonction $f : \mathbb{R}^n \to \mathbb{R}$.
On définit la ligne (ou surface) de niveau l_0 de la fonction f par :

$$L_0 = \{x \in \mathbb{R}^n / f(x) = l_0\} \to \text{hypersurface dans } \mathbb{R}^n \text{ (sous-espace de dimension } n - 1)$$

Exemple graphique : Lignes de niveau pour la fonction de Rosenbrock



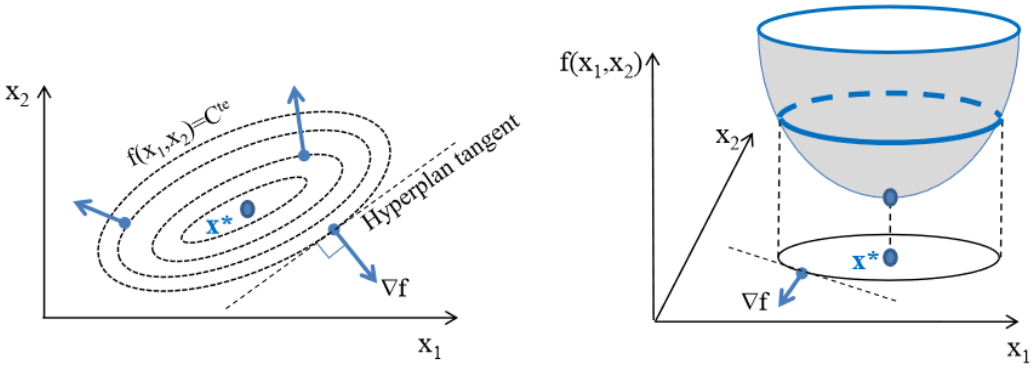
Lien entre gradient et lignes de niveau : Si f est différentiable sur $\mathbb{R}$, pour tout $x_0 \in \mathbb{R}^n$, pour tout $x \in L_{f(x_0)}$,

$$\lim_{x \to x_0} \nabla f(x_0)^t \frac{(x-x_0}{||x-x_0||} = 0$$

En effet :

$$f(x) = f(x_0) + \nabla f(x_0)^t (x - x_0) + \varepsilon(||x - x_0||)$$

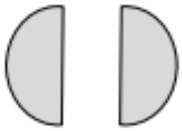
Autrement dit, le gradient de f en x_0 est orthogonal à $L_{f(x_0)}$ et est dirigé vers les valeurs supérieurs de f.



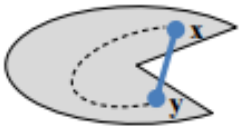
1.2.4 Convexité

Ensemble convexe : $X \subset \mathbb{R}^n$ est convexe si et seulement si :

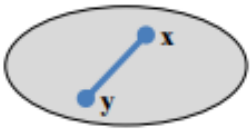
$$\forall (x, y) \in X,\ \forall \lambda \in [0, 1],\ \lambda x + (1 - \lambda) y \in X$$



Non connexe

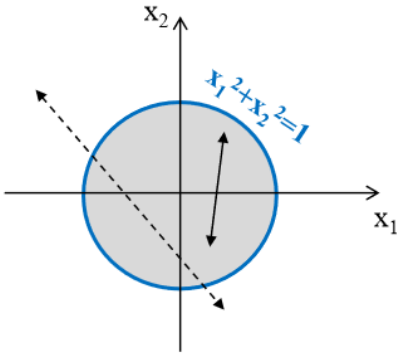


Non convexe



Convexe

Exemple : $X = \{(x_1, x_2)/x_1^2 + x_2^2 \leq 1\} \rightarrow$ convexe.
$X = \{(x_1, x_2)/x_1^2 + x_2^2 \geq 1\} \rightarrow$ non convexe.



Fonction convexe : La fonction $f : \mathbb{R}^n \rightarrow \mathbb{R}$ est convexe si et seulement si :

$$\forall (x, y) \in X,\ \forall \lambda \in [0, 1],\ f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda) f(y)$$

f est strictement convexe si et seulement si :

$$\forall (x, y) \in X,\ x \neq y,\ \forall \lambda \in]0, 1[,\ f(\lambda x + (1 - \lambda)y) < \lambda f(x) + (1 - \lambda) f(y)$$

("f est en-dessous de ses cordes").
Toutes fonctions convexes sur $\mathbb{R}$ sont continues.

Si f est de plus différentiable,

$$f \text{ convexe} \Leftrightarrow \forall (x, y) \in \mathbb{R}^n,\ f(y) \geq f(x) + \nabla f(x)^t (y - x)$$
$$f \text{ strictement convexe} \Leftrightarrow \forall (x, y) \in \mathbb{R}^n,\ x \neq y,\ f(y) > f(x) + \nabla f(x)^t (y - x)$$

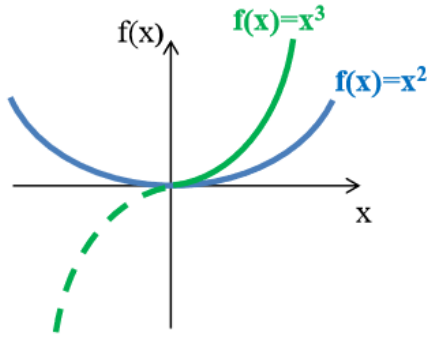
("f est au-dessus de ses tangentes").

Si f est deux fois différentibale,

f convexe $\Leftrightarrow \forall x \in \mathbb{R}^n$, $Hf(x) \geq 0$, c'est-à-dire $\forall d \in \mathbb{R}^n$, $d^t.H.d \geq 0$
f strictement convexe $\Leftrightarrow \forall x \in \mathbb{R}^n$, $Hf(x) > 0$, c'est-à-dire $\forall d \neq 0$, $d^t.H.d > 0$

Si $-f$ est convexe, alors f est concave.

Exemple : $f(x) = x^2$ donc $f''(x) = 2 \rightarrow$ convexe sur $\mathbb{R}$.
$f(x) = x^3$ donc $f''(x) = 6x \rightarrow$ convexe sur $\mathbb{R}^+$ mais non convexe sur $\mathbb{R}$.



1.2.5 Direction de descente

Soit $x \in \mathbb{R}^n$ et $f : \mathbb{R}^n \rightarrow \mathbb{R}$. On dit que $d \in \mathbb{R}^n \backslash \{0\}$ est une direction de descente de f en x si et seulement si :

$$\exists \varepsilon > 0, \forall s \in [0, \varepsilon], f(x + sd) < f(x)$$

si bien que f est strictement décroissante le long de d et dans un voisinage de x.

Propriété 2 *Si f est différentiable, d est une direction de descente de f en x si et seulement si*

$$d^T \nabla f(x) < 0 \quad \textit{ou encore } \langle \nabla f(x), d \rangle < 0 \tag{1.1}$$

De telles directions sont intéressantes en optimisation car, pour faire décroître f, il suffit de faire un déplacement le long de d. Les méthodes à direction de descente utilisent ce principe pour minimiser une fonction, en construisant une suite d'itérer $(x_k)_{k \geq 0}$ approchant une solution $x_* = \arg\min_{x \in \mathbb{R}^n} f(x)$ par la relation

$$x_{k+1} = x_k + \alpha_k d_k$$

où d_k est une direction de descente de f en x_k et $\alpha_k > 0$ un scalaire dit pas de descente. Un algorithme à pas de descente est donc la donnée d'un choix de direction de descente, ainsi que d'une suite de pas. S'ils sont choisis trop petits, l'algorithme n'aboutira qu'au terme d'un très grand nombre d'itérations. S'il est choisi trop grand, on court le risque de "dépasser" un minimum local (et donc de faire un grand nombre d'itérations, voire de diverger). Par exemple, si l'on choisit de minimiser la fonction, pourtant strictement convexe, $f(x) = x^2$ à pas constant de $\alpha = 1.1$ en utilisant $-\nabla f = -2Id$ comme direction de descente, on obtient les itérés suivants

$$\begin{aligned} & x_0 \in \mathbb{R} \\ & x_1 = x_0 - 2.2x_0 = -1.2x_0 \\ & x_2 = -1.2x_0 + 2.4 \times 1.1x_0 = 1.44x_0 \\ & \quad \dots \\ & |x_k| = (1.2)^k |x_0| \to \infty \end{aligned}$$

ce qui justifie la recherche d'un pas convenable, voir optimal : c'est l'objet de la recherche linéaire.

Remarque : $d = -\nabla f(x)$ est toujours une direction de descente (si $\nabla f(x) \neq 0$ car $-\nabla f(x)^T \nabla f(x) = -||\nabla f(x)||^2 < 0$. Ce n'est, toutefois pas la seule. En effet, lorsque f est différentiable, (1.1) définit un demi-espace de $\mathbb{R}^n$. La question se pose alors de savoir, parmi cette infinité de directions, laquelle est la meilleure (au sens qu'elle nous rapproche, au plus vite, d'une solution du problème de minimisation de f). Pour ce faire, on étudie la "tranche" de f le long de la direction de descente d passant par x,

$$\begin{aligned} \Phi : [o, +\infty] & \to \mathbb{R} \\ s & \mapsto f(x + sd) \end{aligned}$$

le but étant d'en déterminer un minimum. On peut commencer par remarquer que, dès lors que f est convexe, ϕ l'est aussi :
$\forall (s,t) \in \mathbb{R}^2,\ \forall 1 < \theta < 1,$

$$\begin{aligned} \phi(\theta s + (1-\theta)t) &= f(x + \theta sd + (1-\theta)td) \\ &= f(\theta(x + sd) + (1-\theta)(x + td)) \leq \theta\phi(s) + (1-\theta)\phi(t) \end{aligned}$$

On en déduit l'existence d'un unique minimiseur s^* de ϕ qui constitue alors un pas optimal de descente le long de d. Il existe un cas simple où s^* se calcule explicitement : c'est l'objet de l'exemple qui suit.

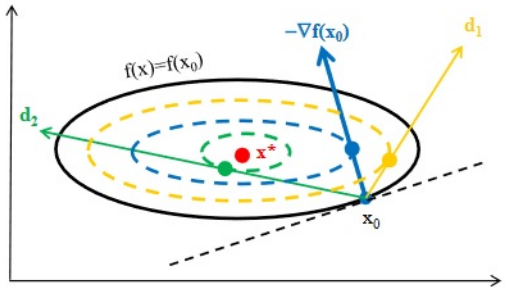


FIGURE 1.1 – Lignes de niveau et directions de descente possibles

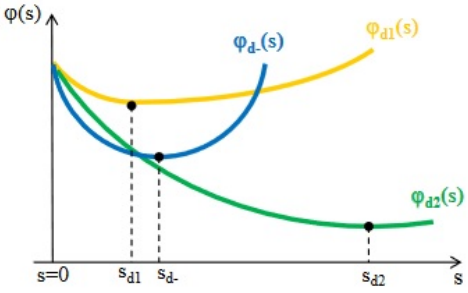


FIGURE 1.2 – Graphes de ϕ définies pour les différentes directions de descente au même point x

Exemple : Pas optimal en optimisation quadratique On se place dans le cas où $f : \mathbb{R}^n \ni x \mapsto \frac{1}{2}x^T.Q.x + b^T.x$ est une forme quadratique avec $Q \in \mathbb{M}_n(\mathbb{K})$ définie positive. $\phi : s \in \mathbb{R}_+ \mapsto f(x_0 + sd_0)$ est alors strictement convexe quels que soient x_0 point de l'espace et d_0 direction de descente de f en x_0. En tant que tel, sa dérivée est strictement monotone et ne s'annule qu'une fois en s^*, minimum global de ϕ. On se convainc facilement que celle-ci s'exprime

$$\phi'(s) = \nabla f(x_0 + sd_0)^T d_0 = (Q(x_0 + sd_0) + b)^T d_0$$

et s'annule donc si, et seulement si,

$$s^* = \frac{-x_0^T Q d_0 - b^T d_0}{d_0^T Q d_0}$$

la division étant licite car Q est définie positive, sauf si $d_0 = 0$. Replaçons-nous dans le contexte d'un algorithme à pas de descente : si la direction de descente calculée est nulle, c'est donc que l'algorithme a convergé, et ce calcul n'a pas lieu d'être.

Le pas s^* ainsi obtenu, lorsque c'est possible, définit un algorithme à pas optimal. Si f n'est pas quadratique, on peut obtenir une approximation de s^* en assimilant le critère à son développement de Taylor d'ordre 2.

1.3 Conditions d'optimalité

1.3.1 Dualité

On verra plus en avant, à l'occasion, par exemple, des conditions d'optimalité de Karush-Kuhn-Tucket que les problèmes de minimisation avec

contrainte ont des variables cachées : les multiplicateurs optimaux de Lagrange. Celles-ci servent non seulement à obtenir des solutions analytiques, mais aussi à définir des algorithmes dits primaux-duaux tel celui d'Uzawa. Or ces multiplicateurs sont, eux-même, solution d'un problème d'optimisation dit problème dual. L'introduction de ce problème définit une première classe de dualité qui fait intervenir le Lagrangien. Il existe une autre classe de dualité dite par pénalisation faisant intervenir une famille de problèmes indexée par un paramètre ρ qui, lorsqu'on le fait tendre vers $+\infty$, rapproche la solution du problème dual (sans contraintes) de la solution du problème primal (avec contraintes).

Problème avec contraintes égalité et inégalité Soit le problème d'optimisation :

$$\min_{x\in\mathbb{R}^n} f(x) \; sous \; \begin{cases} c_I(x) \leq 0 \\ c_E(x) = 0 \end{cases} \tag{PO}$$

Un exemple peut être le problème de la canette.

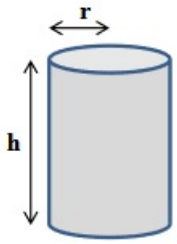
Nous allons prendre en compte les contraintes de différentes manières :

La pénalisation

$$(PO)' : \min_{x\in\mathbb{R}^n} f(x) + \frac{\rho}{2}||c(x)||^2$$

où $c : \mathbb{R}^n \to \mathbb{R}^p$ est une contrainte du problème qui n'est, ici, pas prise en compte comme une contrainte sur le domaine mais sur la fonction coût. Le terme de pénalité est pondéré par un coefficient $\rho > 0$.

Exemple : La canette



On considère le problème de la canette. Nous avons une canette cylindrique de rayon r, de hauteur h, de volume $V = \pi r^2 h$, et de surface $S = 2\pi r^2 + 2\pi rh$.

Soit S la fonction telle que : $S(r, h) = 2\pi r^2 + 2\pi rh = \pi(2r^2 + 2rh)$.

On pose f la fonction telle que : $f(r,h) = 2r^2 + 2rh$ sous la fonction c telle que $c(r,h) = r^2h - 2V$.

On cherche la solution par pénalisation :

$$\min_{r,h} \phi(r,h) = 2r^2 + 2rh + \tfrac{1}{2}\rho(r^2h - 2V)^2$$

On cherche les dérivées partielles du problème :

$$\begin{cases} \partial_r\phi = 0 \\ \partial_h\phi = 0 \end{cases} \Rightarrow \begin{cases} 4r + 2h + 2\rho rh(r^2h - 2V) = 0 \\ 2r + \rho r^2(r^2h - 2V) = 0 \end{cases}$$

La première équation donne $\rho r(r^2h - 2V) = -2$. Dans la deuxième équation, en remplaçant on obtient $h = 2r$. D'où l'on a : $\rho r(2r^3 - 2V) = -2 \Rightarrow \rho r(r^3 - V) + 1 = 0$.

Il suffit de résoudre numériquement l'équation :

$$\rho r^4 - \rho V r + 1 = 0$$

En prenant un volume $V = 1000$. La solution exacte est $r = 10$, $h = 20$, $f = 600$. Pour ρ allant de 10^-3 à 10^2, nous avons le tableau suivant.

ρ	r	h	f	c
0.001	9.641582	19.28316	557.761	792.565
0.01	9.966442	19.93288	595.980	979.933
0.1	9.996664	19.99333	599.600	997.999
1	9.999667	19.99933	599.960	999.800
10	9.999967	19.99993	599.996	999.980
100	9.999997	19.99999	600.000	999.998

On remarque un écart important avec la solution exacte lorsque ρ est petit, on parle de pénalisation faible à contrario pour ρ grand. En général, lorsque $\rho \to +\infty$, la solution approchée tend vers la solution exacte.

Remarque : Lorsqu'on a un problème avec contraintes, on peut se ramener à un problème sans contraintes avec pénalisation. Donc il faudrait trouver le calibrage de ρ.

Problème : Comment fixer ρ ? La technique du Lagrangien augmenté. Afin de remédier à ce problème, on définit un nouveau problème dit dual.Ainsi, on définit la fonction dual.
Soit $\omega(\lambda,\mu) = \min_{x\in\mathbb{R}^n} \mathcal{L}(x,\lambda,\mu)$.
On suppose que ce problème admet une solution : X_ω tel que : $X_\omega = \{(\lambda,\mu), \omega(\lambda,\mu) > -\infty\}$.

Le problème dual en utilisant le Lagrangien

Le problème primal (PO) peut se mettre sous la forme

$$\inf_{x\in\mathbb{R}^n} \sup_{\lambda\in\mathbb{R}^p,\mu\in\mathbb{R}^q_+} \mathcal{L}(x,\lambda,\mu) \qquad \text{(PO')}$$

où $\mathcal{L}(x,\lambda,\mu) = f(x) + \lambda^T C_E(x) + \mu^T C_I(x)$ est le Lagrangien associé au problème. En effet, si l'on fixe x,

$$\sup_{\lambda\in\mathbb{R}^p,\mu\in\mathbb{R}^q_+} f(x) + \lambda^T C_E(x) + \mu^T C_I(x) = \begin{cases} f(x) & \text{si } C_E(x) = 0 \text{ et } C_I(x) \leq 0 \\ +\infty & \text{sinon} \end{cases}$$

car, si l'une des composantes de $C_E(x)$ ou $C_I(x)$ est strictement positive, il suffit de faire tendre la composante de λ ou μ de même rang vers $+\infty$ pour que $\mathcal{L}(x,\lambda,\mu)$ tende vers $+\infty$. De la même façon, si l'une des composantes de $c_E(x)$ est strictement négative il suffit de faire tendre la composante de λ qui lui correspond vers, cette fois, $-\infty$. Nous avons donc montré la seconde ligne de la disjonction de cas. Reste à montrer que si l'on se place dans le cas où $C_E(x) = 0$ et $C_I(x) \leq 0$, le sup vaut bien $f(x)$. Le produit de toute composante non-nulle de C_I par une composante non nulle, donc strictement positive, de μ donne un scalaire strictement négatif. Ainsi, le sup ne peut être atteint que lorsque cette composante de μ est nulle. Partant, μ et $C_I(x)$ sont orthogonaux et le passage au sup ne préserve que $f(x)$. Aussi bien, (PO') devient

$$\inf_{x\in tie\mathbb{R}^n, C_E(x)=0, C_I(x)\leq 0} f(x)$$

soit le problème de départ (PO). Le problème dual est alors celui où les opérateurs sup et inf sont échangés, soit :

$$\sup_{\lambda\in\mathbb{R}^p,\mu\in\mathbb{R}^q_+} \inf_{x\in\mathbb{R}^n} \mathcal{L}(x,\lambda,\mu) \qquad \text{(PO")}$$

Appuyons le fait que la fonction coût de ce problème n'est pas le Lagrangien, mais bien la fonction

$$\lambda,\mu \mapsto \inf_{x\in\mathbb{R}^n} \mathcal{L}(x,\lambda,\mu)$$

Ces problèmes ne sont, *a priori* , pas équivalents. Ils partagent néanmoins le lien suivant :

Théorème 5 (Principe de dualité Faible) *Soit $\varphi : x, y \in X \times Y \to \mathbb{R}$ quelconque. Alors l'inégalité suivante est toujours vraie :*

$$\sup_{y\in Y} \inf_{x\in X} \varphi(x,y) \leq \inf_{x\in X} \sup_{y\in Y} \varphi(x,y)$$

Démonstration : Commençons par remarquer que, pour tous $x_0 \in X$ et $y_0 \in Y$,

$$\varphi(x_0, y_0) \leq \sup_{y \in Y} \varphi(x_0, y)$$

Comme l'inégalité tient pour tout x_0, l'on peut passer à l'inf,

$$\forall\ y_0 \in Y,\ \inf_{x \in X} \varphi(x, y_0) \leq \inf_{x \in X} \sup_{y \in Y} \varphi(x, y)$$

et, comme cette dernière tient pour tout $y_0 \in Y$, le passage au sup ne l'altère pas :

$$\sup_{y \in Y} \inf_{x \in X} \varphi(x, y_0) \leq \inf_{x \in X} \sup_{y \in Y} \varphi(x, y)$$

D'après ce résultat, les solutions du problème dual (PO") sont majorées par celles du problème primal (PO), le cas le plus désirable étant celui d'égalité puisque les problèmes sont, alors, équivalents. Dans ce cas, on dit qu'il n'y a pas de saut de dualité, lequel est, plus généralement, la différence entre les membres de l'inégalité. Le principe de dualité forte donne une condition nécessaire et suffisante à l'absence de saut de dualité, donc à l'équivalence des problèmes. Avant de l'énoncer, introduisions la notion de point selle.

Définition 4 *Soit $\varphi : (x, y) \in X \times Y \to \mathbb{R}$. On dit que $(\overline{x}, \overline{y}) \in X \times Y$ est un point selle de φ si :*

$$\forall (x, y) \in X \times Y,\ \varphi(\overline{x}, y) \leq \varphi(\overline{x}, \overline{y}) \leq \varphi(x, \overline{y})$$

ou, encore, si c'est un point selle de $-\varphi$.

Théorème 6 (Principe de dualité forte) *$(\overline{x}, (\overline{\lambda}, \overline{\mu}))$ est un point selle de $\mathcal{L}$ si, et seulement si, $\overline{x}$ est solution du problème primal* (PO)*, $(\overline{\lambda}, \overline{\mu})$ est solution du problème dual* (PO") *et il n'y a pas de saut de dualité.*

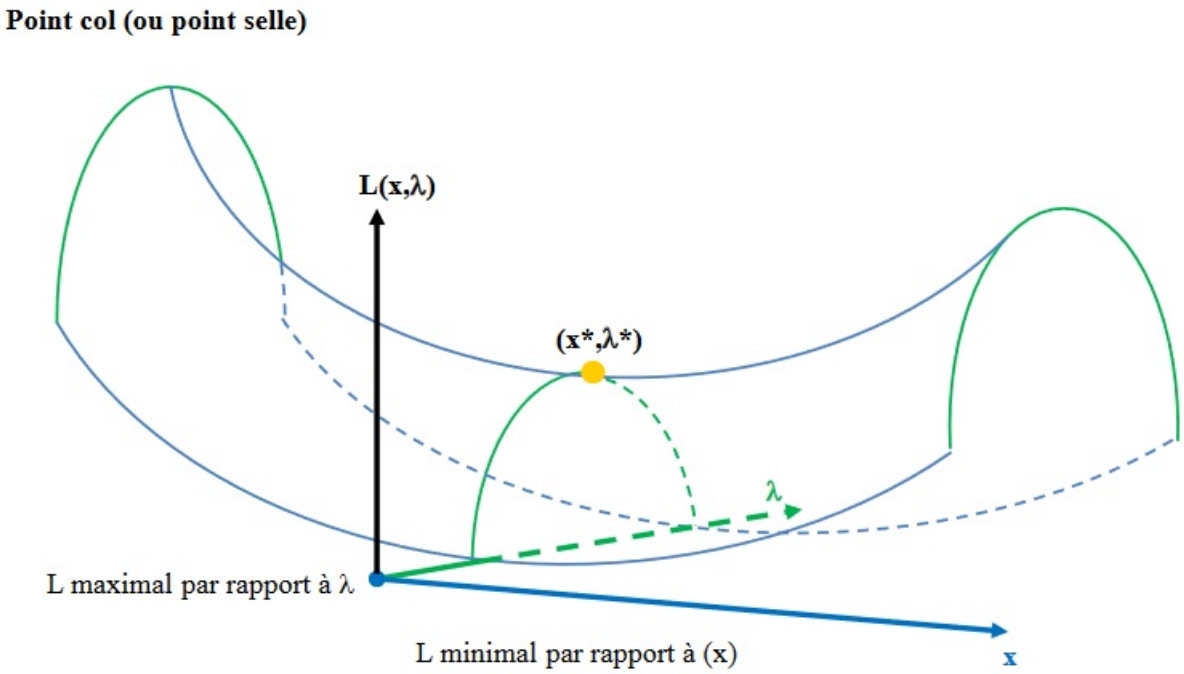


FIGURE 1.3 – Illustration d'un point selle

Optimisation

M1 MINT et MMM 2018

30 octobre 2018

Table des matières

1 Introduction et Rappels

1.1 Quelques exemples d'optimisation de formes

1.2 Condition d'optimalité

1.2.1 Cas d'un problème sans contrainte

3 types de conditions

On s'intéresse à un Problème d'Optimisation(PO) où l'on souhaite à minimiser une fonction f : $\min_{x\in\mathbb{R}^n} f(x)$,où $f(x) \in C^1$ (voir C^2)

On a donc 3 conditions de 2 types :

-CN1 : une condition nécessaire d'ordre 1 soit $f(x) \in C^1$

-CN2 : une condition nécessaire d'ordre 2 soit $f(x) \in C^2$

-CS1 : une condition suffisante d'ordre 1 soit $f(x) \in C^1$

on notera qu'il n'existe pas de condition nécessaire et suffisante (**CNS**)

Cas en 1 dimension

On se place en dimension 1

Pour f : $\mathbb{R} \mapsto \mathbb{R}$

Ainsi nos 3 conditions deviennent :

-CN1 : f admet un minimum implique :

$x^* \Longrightarrow f'(x^*) = 0$

-CN2 : f admet un minimum implique :

$\begin{cases} f'(x^*) & = 0 \\ f''(x^*) & \geq 0 \end{cases}$

on a la condition d'un point stationnaire et aussi d'une dérivée seconde positive .

-CS2 :

$\begin{cases} f'(x^*) & = 0 \\ f''(x^*) & > 0 \end{cases}$

on a la condition d'un point stationnaire et aussi d'une dérivée seconde positive strict qui implique alors que x^* est un minimum local.

Cas général

On se place dans $\mathbb{R}^n$, avec $n \in \mathbb{N}^*$

CN1 : f admet un minimum local.

$x^* \in \mathbb{R}^n \Longrightarrow \nabla f(x^*) = 0$

CN2 : f admet un minimum

$\begin{cases} \nabla f(x^*) & = 0 \\ Hf(x^*) & \geq 0 \end{cases}$

avec $\forall d \in \mathbb{R}^n \ d^T Hf(x^*)d \geq 0$

On a la condition d'un point stationnaire et aussi d'un hessien défini semi positif.

CS2 :si $x^* \in \mathbb{R}^n$ et que

$\begin{cases} \nabla f(x^*) & = 0 \\ Hf(x^*) & > 0 \end{cases}$

avec $\forall d \in \mathbb{R}^n \backslash 0$, $d^T Hf(x^*)d > 0$ ce qui implique que x^* est un minimum local de f. on a la condition d'un point stationnaire et aussi d'un hessien défini positif.

Démonstration

La démonstration se base sur les formules de Taylor à l'ordre 1 et 2 :

Preuve de **CN1 :**

On suppose par un raisonnement par l'absurde que $\nabla f(x^*) \neq 0$.
il existe donc $h \in \mathbb{R}^n$ tel que $\tilde{h}^T \nabla f(x^*) < 0$
on pose la formule de Taylor d'ordre 1 :
$f(x+h) = f(x) + h^T \nabla f(x)$
on pose $t > 0$, $f(x^* + t\tilde{h}) = f(x^*) + t\tilde{h}^T \nabla f(x^*) + o(t)$ avec $f(x^*)$ un minimum local
or $\nabla f(x^*) \neq 0$ alors pour un t suffisamment petit on obtient un $t\tilde{h}^T \nabla f(x^*) < 0$
soit $f(x^* + t\tilde{h}) < f(x^*)$ ce qui impossible car $f(x^*)$ est un minimum locale.

□

Preuve de **CN2 :**

On utilise la formule de Taylor à l'ordre 2 :
$f(x^* + d) = f(x^*) + \nabla f(x^*)^T d + \frac{1}{2} d^T Hf(x^*)d + o(\|d\|^2)$
avec $\nabla f(x^*) = 0$ donc on a $f(x^* + d) = f(x^*) + \frac{1}{2} d^T Hf(x^*)d + o(\|d\|^2)$ pour un d très petit
on a : $d^T Hf(x^*)d < 0$ donc on a $f(x^* + d) < f(x^*)$ ce qui est impossible car
$f(x^*)$ est un minimum local .

□

Preuve de **CS2 :**

Toujours par l'absurde mais cette fois ci on suppose que x^* n'est pas un minimum local,
alors $\exists d \in \mathbb{R}^n \backslash 0$ tel que $f(x^* + d) < f(x^*)$.
On a notre formule de Taylor à l'ordre 2 :
$f(x^* + d) = f(x^*) + \nabla f(x^*)^T d + \frac{1}{2} d^T Hf(x^*)d + o(\|d\|^2)$
avec $\nabla f(x^*) = 0$ (la première condition de la CS2).
on a toujours $f(x^* + d) < f(x^*)$ ce qui implique $d^T Hf(x^*)d < 0$ or $Hf(x^*) > 0$
(deuxième condition de la CS2) donc impossible .

□

méthode

Rechercher des points stationnaires avec $\nabla f(x^*)$ (CN1) :
Pour cela on cherche $\nabla f(x) = 0$ permettant de trouver le ou les x^*.
Pour savoir si un point stationnaire est minimum,maximun ou un point selle :
On calcule $Hf(x^*)$ puis on vérifie que la hessienne est strictement positive ($Hf >> 1$
c'est la CN2 et CS2) soit pour des valeurs propres positives soit grâce au critère de Sylvester :

soit $A \in \mathbb{S}^n = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ sym & \ddots & \vdots \\ & & a_{nn} \end{pmatrix}$ $A >> 1$ si $\begin{cases} a_{11} & > 0 \\ detA & > 0 \end{cases}$

remarques

Pour avoir un minimum global : Avec x^* est un minimum local

Si f est convexe alors x^* est un minimum global.

Si f est *strictement* convexe alors x^* est un *unique* minimum global.

Si f est quelconque , on ne pourra pas savoir.

Exemples

exemple 1 :

la fonction de Rosenbrock : $f(x,y) = 100(y-x^2)^2 + (x-1)^2$

on calcule le point stationnaire : $\nabla f(x) = 0 \Leftrightarrow \begin{cases} -400x(y-x^2) + 2(x-1) & = 0 \\ 200(y-x) & = 0 \end{cases}$

$\Leftrightarrow \begin{cases} (1-x)(400x^2+2) & = 0 \\ y & = x \end{cases} \Leftrightarrow \begin{cases} x & = 1 \\ y & = x \end{cases}$

soit $f(x^*, y^*) = (1,1)$

calculons la hessienne de ce point critique soit $Hf(1,1) = \begin{pmatrix} 802 & -400 \\ -400 & 200 \end{pmatrix} >> 0$

Puisque $802 > 0$ et $detHf(1,1) = 802 * 200 - 400^2 > 0$.

Ainsi le couple (1,1) est bien le minimum locale.

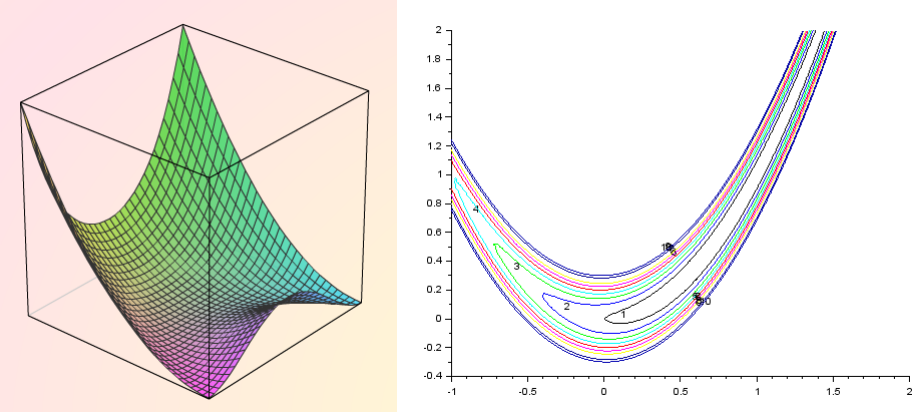


FIGURE 1 – représentation 3D de la fonction de Rosenbrock et ligne de niveau en 2D

exemple 2 :

$f(x,y) = -x^4 - y^3$

on calcule le point stationnaire : $\nabla f(x) = 0 \Leftrightarrow \begin{cases} -4x^3 & = 0 \\ -4y^2 & = 0 \end{cases} \Leftrightarrow \begin{cases} x & = 0 \\ y & = 0 \end{cases}$

soit $f(x^*, y^*) = (0,0)$ calculons la hessienne de ce point critique soit $Hf(0,0) = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \geqslant 0$

nous ne pouvons rien conclure à ce stade puisque que nous somme entre la CS2 et CN2.

Nous pouvons calculer $f(0,0) = 0$ qui est ici un maximum global puisque $f(x,y) \leqslant 0$

exemple 3 :

$f(x,y) = x^2 - y^3$

on calcule le point stationnaire : $\nabla f(x) = 0 \Leftrightarrow \begin{cases} 2x & = 0 \\ -3y^2 & = 0 \end{cases} \Leftrightarrow \begin{cases} x & = 0 \\ y & = 0 \end{cases}$

soit $f(x^*, y^*) = (0,0)$ calculons la hessienne de ce point critique soit $Hf(0,0) = \begin{pmatrix} 2 & 0 \\ 0 & 0 \end{pmatrix} \geqslant 0$

On ne peut rien conclure à ce stade nous sommes entre la CN2 et la CS2.

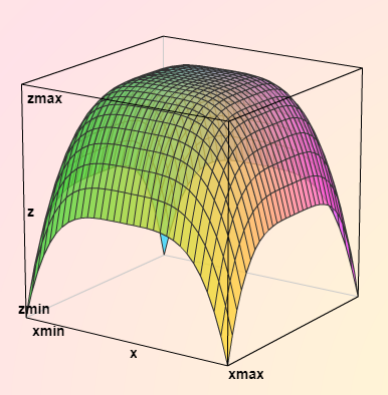


FIGURE 2 – représentation 3D de la fonction

si on fixe $x = 0$ la fonction devient $f(0, y) = -y^3$ soit $\begin{cases} f > 0 & \text{si } y < 0 \\ f < 0 & \text{si } y > 0 \end{cases}$
Il ne s'agit ni d'un minimum ni d'un maximum. C'est un cas dégénéré.

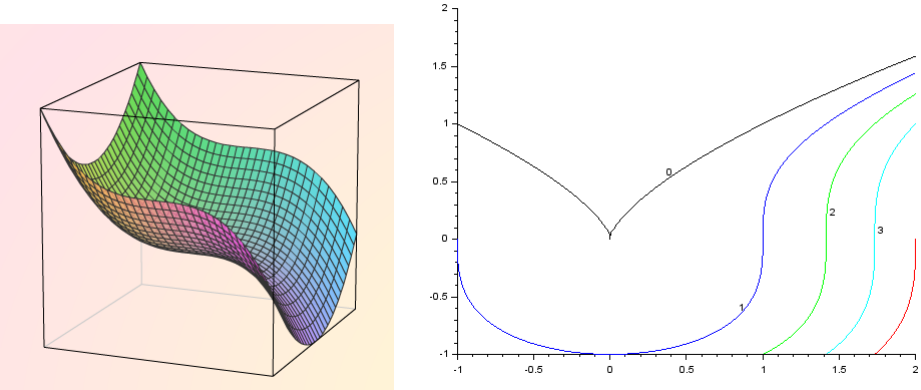


FIGURE 3 – representation 3D de la fonction et ligne de niveau en 2D

2 Condition nécessaire d'optimalité locale

$\min f(x)$ en $x \in \mathbb{R}^n$ sous $Ce(x^*) = 0$ $Ci(x) \leq 0$
Condition nécessaire : x* minimum local$\Rightarrow \nabla f(x^*)\ {}^t d \geq 0$

2.1 Méthode directe

Nécessaire de connaitre l'ensemble des directions admissibles en x*.
Cas de contraintes linéaires :
- Définitions des directions admissibles à partir des directions de bases.
Cas de contraintes non linéaires :
- Définitions des directions admissibles à la limite.
- Pas de caractérisation des directions admissibles.

2.2 Méthodes indirecte

Soit la langragienne $L(x, \lambda, \mu) = f(x) + \lambda\ {}^t Ce(x) + \mu^r Ci(x)$
tel que $\lambda \in \mathbb{R}^p; \mu \in \mathbb{R}^p; \mu \geq 0$

A partir des multiplicateurs de Lagrange :
- Conditions d'optimalité dans le cas général
- (Cas particulier) Si les multiplicateurs sont nuls les contraintes sont inactives.

2.3 Problèmes avec contraintes (KKT)

Soit $\min f(x)$ sous $Ce(x) = 0$ $Ci(x) \leq 0$
Condition nécessaire :
Hypothèse : Contraintes linéairement indépendantes en x*.
x* est plus minimum local $\Rightarrow \exists$ un unique $\lambda^* \in \mathbb{R}^p$ et un unique $\mu^* \in \mathbb{R}^q$
tel que :

$$\text{Ordre 1 : } \begin{cases} \nabla L(x^*; \lambda^*; \mu^*) = 0 \\ \nabla_\lambda L(x^*; \lambda^*; \mu^*) = 0 \\ \nabla_\mu L(x^*; \lambda^*; \mu^*) \leq 0 \\ \mu^* \geq 0 \\ \mu^* Ci(x^*) = 0 \end{cases}$$

Ordre 2: Pour toutes direction (d) tangente aux contraintes actives $(c(x^*) = 0)$:

$$\begin{cases} d\ {}^t\nabla^2{}_{xx} L(x*; \lambda*; \mu*) d \geq 0 \\ \quad \forall d\ ,\ d\ {}^t\nabla C(x^*) = 0 \end{cases}$$

3 Exemples

$\min \frac{1}{2}({x_2}^2 - {x_1}^2)$ sous $x_1 \leq 1$ $L(x_1; x_2; \mu) = \frac{1}{2}({x_2}^2 - {x_1}^2) + \mu(x_1 - 1)$

$$\text{Ordre 1 : } \begin{cases} -x1 + \mu = 0 \\ x2 = 0 \\ x1 \leq 1 \\ \mu \geq 0 \\ \mu(x1 - 1) = 0 \end{cases}$$

Notre système est vérifié en $x^* = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ $\mu^* = 1$

Ordre 2 : $d\ {}^t\nabla C(x^*) = 0 \Rightarrow \begin{pmatrix} d1 \\ d2 \end{pmatrix}\ {}^t \begin{pmatrix} 1 \\ 0 \end{pmatrix} = 0 \Rightarrow d1 = 0$

$d\ {}^t\nabla^2{}_{xx} L(x*; \lambda*; \mu*)d = \begin{pmatrix} d1 \\ d2 \end{pmatrix}\ {}^t \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} d1 \\ d2 \end{pmatrix} = -{d_1}^2 + {d_2}^2 = {d_2}^2 \geq 0$

Donc $x^* \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ et $\mu^* = 1$(vérifie les conditions nécéssaires)

3.1 Définitions

Problèmes d'optimisation avec contraintes

$\min_{x \in \mathbb{R}^n} f(x)$ sous $\begin{cases} c_I(x) \leq 0 \text{ avec q contraintes d'inégalité} \\ c_E(x) = 0 \text{ avec p contraintes d'égalité} \end{cases}$

Conditions nécessaires :

Hypothèse : Contraintes linéairement indépendantes en x^*

Ainsi sous les conditions KKT(ou Karush-Kuhn-Tucker) , on reprend les conditions nécessaires précédentes en ajoutant un multiplicateur poids pour la fonction avant de l'injecter dans le Lagrangien :

$L(x, \lambda, \lambda_0) = \lambda_0 f(x) + \lambda^T c(x)$ avec $\lambda \in \mathbb{R}^m$: multiplicateurs des contraintes et $\lambda_0 \in \mathbb{R}$: multiplicateur du critère $\lambda_0 \geq 0$, selon deux cas :

.Celui où $\lambda_0 > 0$ soit lorsque le Lagrangien prends en compte un poids de la fonction étudié, on prendra $\lambda_0 = 1$ bien souvent.

.Ou bien $\lambda_0 = 0$ le cas dit anormal, où le système lagrangien n'admet de solutions que si $\lambda_0 = 0$. La solution satisfait les contraintes, mais la valeur du critère n'est pas « minimisable ». on pourra dire que ceci équivaut à une valeur infinie des multiplicateurs λ.

Les conditions KKT donnent un système de $n + m$ équations à $n + m + 1$ inconnues :

$\begin{cases} \nabla_x L(x, \lambda, \lambda_0) = 0 \longrightarrow n \\ \nabla_\lambda L(x, \lambda, \lambda_0) = 0 \longrightarrow m \end{cases}$

Soient les conditions suffisantes : s'il existe $x^* \in \mathbb{R}^n$ un minimum local de f, $\lambda^* \in \mathbb{R}^p, \mu^* \in \mathbb{R}^q$ tels que :

. Ordre 1 : $\begin{cases} \nabla_x L(x^*, \lambda^*, \mu^*) = 0 \\ \nabla_\lambda L(x^*, \lambda^*, \mu^*) = C_E(x^*) = 0 \\ \nabla_\mu L(x^*, \lambda^*, \mu^*) = C_I(x^*) = 0 \\ \mu^* \geq 0 \\ \mu^* C_I(x^*) = 0 \\ \text{Car } C_I(x^*) > 0 \text{ , alors } \mu^* = 0 \end{cases}$

. Ordre 2 : Pour toute direction d tangente aux contraintes actives ($c(x^*) = 0$) : $\begin{cases} d^T \nabla^2_x x L(x^*, \lambda^*, \mu^*) d > 0 \\ \forall \text{ d tel que } d^T \nabla c(x^*) = 0 \end{cases}$

. Démonstration :

Éléments de la démonstration Cas de contraintes égalité : c(x)=0 On suppose que (x*,O*) vérifie les conditions suffisantes. On considère le problème sans contrainte en utilisant le lagrangien augmenté $\min_{x \in \mathbb{R}^n} L_p(x, \lambda^*) + \frac{1}{2}\rho ||c(x)||^2 = f(x) + \lambda^{*T} c(x) + \frac{1}{2}||c(x)||^2$ p>0 joue le rôle de pénalisation de la violation des contraintes. . $\nabla_x L_p(x^*, \lambda^*) = \nabla f(x^*) + \nabla c(x^*)\lambda^* + \rho \nabla c(x^*) c(x^*) =$

$\nabla_x L(x^*, \lambda^*) = 0$. $\nabla^2_{xx} L_p(x^*, \lambda^*) > 0$ pour ρ assez grand. Au voisinage de x^* : $L_p(x^*, \lambda^*) \leq L_p(x, \lambda^*) => f(x^*, \lambda^*) \leq f(x, \lambda^*)$, x tel que $c(x) = 0$. Ainsi x^* est bien un minimum local de f

En pratique : Toute résolution de problème d'optimisation avec contraintes nécessite d' identifier les contraintes actives : $c_E(x)$, $c_I(x)$ Ce qui peut se ramener à un problème plus simple de contraintes :

-> résolution des conditions KKT d'ordre 1

-> vérification des conditions réduites d'ordre 2

Ex : Conditions suffisantes

$min_{x_1,x_2} \frac{1}{2}(x_2^2 - x_1^2)$ sous $x_1 \leq 1$ $x^* = \begin{matrix}1\\0\end{matrix}, \mu^* = 1$ vérifie les conditions nécessaires

. CS1 : contrainte active -> multiplicateur >0 $x^* - 1 = 0$ $\mu^* = 1 > 0$

. CS2 : d direction tangente aux contraintes actives : $d^T \nabla c(x^*) = 0 => \begin{cases} {d_1}^T \\ d_2 \end{cases}$

$x^* = \begin{pmatrix} -1 \\ 0 \end{pmatrix}, \mu^* = 1$ vérifie les conditions suffisantes d'ordre 1 et 2 -> minimum local strict.

Optimisation sans contraintes

El Mahi Adib

Boussalem Mohamed

Lemhaba Cheick Mohamed

Ben Omar Marrakchi Marouan

2018
Décembre

Sommaire

1 Rappels :

Nous allons dans cette partie voir quelques rappels utiles pour la suite du cours et TD.

- Pour touver un minimum local x : $\begin{cases} \nabla f(x) = 0 \\ \nabla^2 f(x) = 0 \end{cases}$

Avec $\nabla f(x) = \begin{pmatrix} \frac{\partial f(x)}{\partial x_1} \\ \frac{\partial f(x)}{\partial x_2} \\ . \\ . \\ . \\ \frac{\partial f(x)}{\partial x_n} \end{pmatrix}$ et $\nabla^2 f(x) = \begin{pmatrix} \frac{\partial f(x)}{\partial^2 x_1} & \cdots & \\ \frac{\partial^2 f(x)}{\partial x_1 x_2} & & \\ . & & \\ . & & \\ . & & \\ \frac{\partial^2 f(x)}{\partial x_1 x_n} & \cdots & \frac{\partial^2 f(x)}{\partial {x_n}^2} \end{pmatrix}$

$\nabla f(x)$ est appelé gradient de f(x)

$\nabla^2 f(x)$ est appelé Hessien de f(x)

- une direction de descente est une direction le long de laquelle la fonction à minimiser a une dérivée directionnelle strictement négative :

d $\in (\mathbf{R^n})^*$ est une direction de descente en $x_0 \in \mathbf{R^n}$ si
$\exists\ \epsilon > 0$ tel que $f(x_0 + Sd) < f(x_0)$ si S $\in]0, \epsilon]$, d est caracterise.
Si $\nabla f(x_0) \neq 0$, d est caracterise par la condition $d^t \nabla f(x_0) \neq 0$.

Par exemple:
$d_k = -\nabla f(x_0)$ est une direction de descente.

Principe général:
$x_0 \in \mathbb{R}^{\ltimes}$ fixe.
k $\rightarrow k+1 : \Rightarrow$ si $\nabla f(x_k) = 0$, on s'arrête
$\Rightarrow$ sinon, on choisit d_k tel que $d_k \nabla f(x_k) < 0$ et α_k tel que $f(x_k + \alpha_k d_k) < f(x_k)$ et on définit :

$$x_{k+1} = x_k + \alpha_k d_k$$

$d_k \in (\mathbf{R^n})^*$: direction de descente

$\alpha_k \in (\mathbf{R}_+)^*$: pas de descente

- Méthode de Newton : la méthode de Newton est, dans son application la plus simple, un algorithme efficace pour trouver numériquement une approximation précise d'un zéro (ou racine) d'une fonction réelle,d'une variable réelle.

On cherche à nouveau à minimiser (localement) une fonction.

$f : \mathbf{R^n} \rightarrow \mathbf{R}, C^2$.

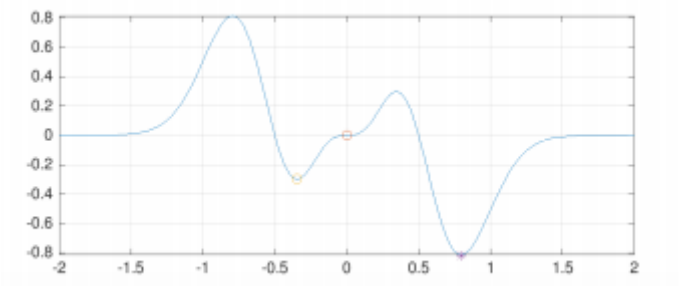
Si on applique la méthode de Newton précédente a g(x) = ∇ f(x) , on trouve :

$$x_0 \in \mathbf{R^n}$$
$$x_{k+1} = x_k - Hf^{-1}(x_k)\nabla f(x_k)$$

-Notion de minimum:

$_*x^* \in X_{adm}$ est un minimum global de (PO) ssi $\forall x \in X_{adm}, f(x^*) \leq f(x)$

$_*x^* \in X_{adm}$ est un minimum local de (PO) ssi $\exists \epsilon > 0$,
$\forall x \in X_{adm}, \|x - x^*\| \leq \epsilon \Rightarrow f(x^*) \leq f(x)$.



Dans la figure précédente nous avons plusieurs minimums locaux en -0.4 , 0 et 0.75 mais un seul minimal global en 0.75.

- Algorithme de Méthode de Newton :

On initialise (n=0) : On choisit x_0 pour $f^{''}(x_0)f(x_0) > 0$

$$x_0 = a \qquad et \qquad r_0 = f(x_0)$$

Tant que ($n < n_{max}$ et $(a_n - b_n) > \epsilon$ et $r_n > \epsilon$)

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

n=n+1
$r_n = f(x_n)$

Converge localement

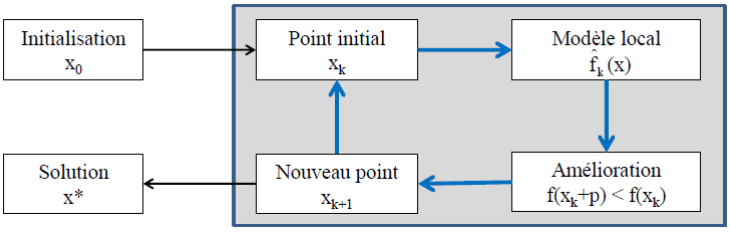
2 Méthodes de descente

En optimisation différentiable, qui est une discipline d'analyse numérique en mathématiques étudiant en particulier les algorithmes minimisant des fonctions différentiables sur des ensembles, une direction de descente est une direction le long de laquelle la fonction à minimiser a une dérivée directionnelle strictement négative. Ces directions sont utilisées par les méthodes à directions de descente. C'est le long de ces directions qu'un déplacement est effectué afin de trouver l'itérée suivant, en lequel la fonction à minimiser prend une valeur inférieure à celle qu'elle a en l'itérée courant. Des directions de descente peuvent être calculées par de nombreuses techniques dont les plus classiques sont présentées ci-dessous.

Problème sans contrainte

$min_{x \in R^n} f(x)$: x minimum local $\begin{cases} \nabla f(x) = 0 \\ \nabla^2 f(x) \geq 0 \end{cases}$

On ne sait pas trouver le minimum global dans le cas général $\forall f$
Méthode locale
Initialisation $x_0 \rightarrow$ recherche d'un minimum local au voisinage de x_0
Itérations $\rightarrow$ passage du point x_k à un meilleur point x_{k+1}
Arrêt $\rightarrow$ solution x ou blocage



2.1 Itérations

Modèle local : prédiction

Point courant x_k ,$f_k = f(x_k)$
Evaluation de $g_k = \nabla f(x_k)$ ou approximation (différences finies)
$H_k = \nabla^2 f(x_k)$ ou approximation (quasi Newton)
Modèle quadratique : $min f_k(x_k + p) = f_k + P^t g_k + \frac{1}{2} P^t H_k P$
$\rightarrow x_{k+1} = x_k + p$
Méthodes de Newton ou quasi-Newton
Amélioration : correction
Nouveau point $x_{k+1} = x_k + p$ tel que $f(x_k + p) < f(x_k)$
Déplacement p à partir de x_k
par recherche linéaire suivant $d_k = x_{k+1} - x_k$
Méthodes de globalisation
La méthode de Newton appliquée directement ne converge pas systématiquement.
La globalisation est nécessaire pour contrôler la convergence.

2.2 Initialisation et arrêt

Initialisation

Les méthodes locales recherchent le minimum au voisinage du point de départ.
Objectifs : rapidité de convergence
précision de convergence
$\rightarrow$ Méthodes à base de dérivées
Le minimum local trouvé est le plus proche du point initial x_0.
$\rightarrow$ initialisation à modifier pour trouver un autre minimum local

Les méthodes $\ll$ globales $\gg$ explorent $\ll$ aléatoirement $\gg$ les solutions
$\rightarrow$ localisation possible de plusieurs minima locaux
Conditions d'arrêt
Déplacement insuffisant : $\|x_{k+1} - x_k\| < \epsilon_k$
Amélioration insuffisante : $f_k - f_{k+1} < \epsilon_k$
Condition d'ordre 1 vérifiée : $\|g_k\| < \epsilon_g$

Nombre maximal d'itérations ou d'appels fonction : N_{iter}, N_{fonc}

2.3 Résolution d'équation:

L'idée consiste cette fois ci à remplacer l'arc de courbe (AB) par la droite tangente à la courbe aux points A ou B, l'intersection de cette dernière avec l'axe détermine le second élément x_1 de la suite cherchée On réitère en passant la tangente à la courbe au point d'abscisse x etc.... D'une manière générale les itérations de Newton pour résoudre système d'équations non linéaires $g(x) = 0$ $x_{k+1} = x_k - \frac{g(x)}{g'(x)}$ $n = 0, 1, 2...$

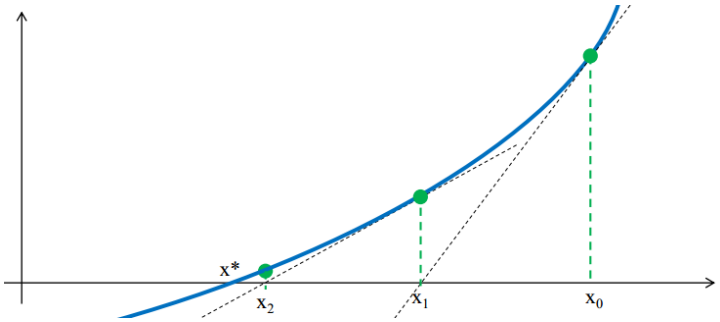


Figure 1: Méthode de Newton

2.4 Fonction modèle:

2.4.1 Algorithme de Newton dans R :

Développement de Taylor à l'ordre 1 de g en x_k.
$g(x) = g(x_k) +^T (x - x_k) + \|x - x_k\|$
$x_k + 1 = x_k - \frac{g(x)}{g'(x)}$
critére d'arrêt: $\|x_{k+1} - x_k\| < \varepsilon$, on stop sinon on prend $k = k + 1$, et on retourne à un nouveau point initiale 2. - chaque itération $x_k + 1$ est obtenu à

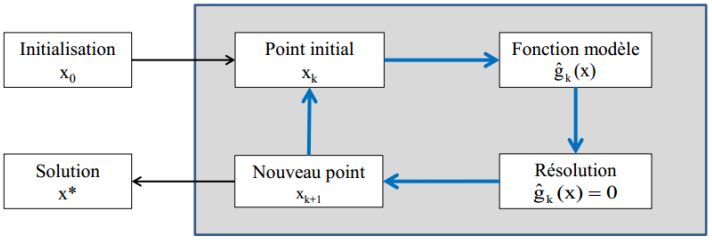


Figure 2: Méthode de Newton

partir du précédent en traçant la tangente à la courbe, j au point x_k, $f(x_k)$, et en considérant l'intersection à l'axe des abscisse de x.

2.4.2 Méthode de Newton dans R^n:

$\hat{g} = g(x_k) + G_k(x - x_k)$
avec $G_k = g(x_k) + (x - x_k)^T + \Theta(\|x - x_k\|)$
pour choisir la matrice G_k on à deux méthodes :
- La méthode de Newton $G_k = \nabla g(x_k)^T$ matrice jacobienne de g en point x_k
- ou bien la méthode de quasi-Newton $G_k = \nabla g(x_k)^T$
- x_k converge vers x^*

exemple 1:
-fonction: $g(x) = x^2 - 1$
-dérivée: $g'(x) = 2x$
solution: $x = 1 \longrightarrow g'(1) = 2$

Iteration	x(k)	g(x)=x**2-1	g'(x)=2x	Erreur
0	4,00000000	1,5E+01	8,0000	3,0E+00
1	2,12500000	3,5E+00	4,2500	1,1E+00
2	1,29779412	6,8E-01	2,5956	3,0E-01
3	1,03416618	6,9E-02	2,0683	3,4E-02
4	1,00056438	1,1E-03	2,0011	5,6E-04
5	1,00000016	3,2E-07	2,0000	1,6E-07
6	**1,00000000**	**2,5E-14**	**2,0000**	**1,3E-14**

Figure 3: convergence quadratique $g'(x^*)$

2.5 Intérêt de la méthode de Newton:

- Convergence quadratique au voisinage de la solution $\longrightarrow$ très rapide et précise.
- Méthode à privilégier dans les algorithmes d'optimisation.

2.5.1 Difficultés:

- Calcul explicite et inversion du Hessien $\nabla^2 f(x_k)$ à chaque itération $\longrightarrow$ coûteux
- Convergence non garantie même près de la solution$\longrightarrow$ mêmes difficultés que pour la résolution d'équations
- 1ère condition nécessaire de minimum:$\nabla f(x^*) = 0$

$\longrightarrow$ point stationnaire x^*=minimum local, maximum local au point selle
$\longrightarrow$ 2ème condition nécessaire de minimum à vérifier :$\nabla f(x^*) \geq 0$.

2.5.2 Adaptation:

-Méthodes de quasi-Newton $\longrightarrow$ G_k approximation du gradient , $\nabla g(x_k)$ construit à partir des itérations précédentes sans calcul explicite du gradient
-Techniques de globalisation$\longrightarrow$ Contrôle du point x_{k+1} meilleur que x_k $g\|x_{k+1}\| < g\|x_k\|$

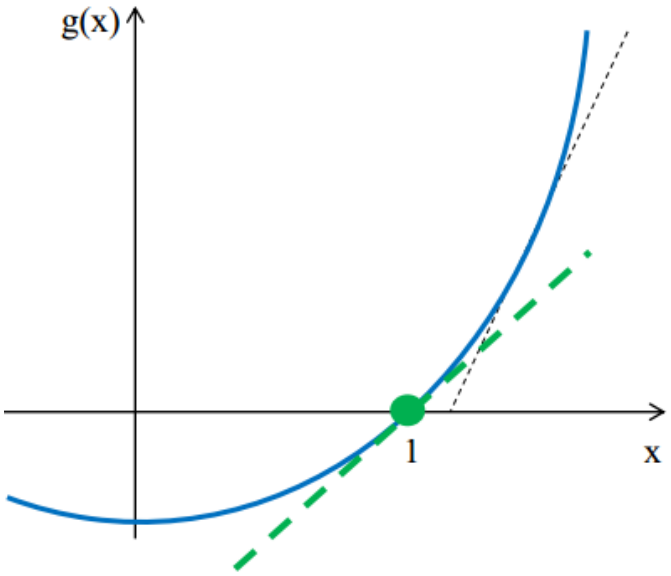


Figure 4: la convergence d'une fonction

2.6 Méthode de SR1

SR1 la méthode de SR1 permet de déterminer la matrice H_k de l'équation sécante $H_k d_{k-1}$=y_{k-1} On cherche la matrice $H_{k-1} + UU^T$
formule de SR1: $H_k = H_{k-1} + \frac{(y_{k-1}-H_{k-1}d_{k-1})(y_{k-1}-d_{k-1})^T}{d_{k-1}^T(y_{k-1}-H_{k-1}d_{k_1})}$
avec : $d_{k-1} = x_k - x_{k-1}$
$y_{k-1} = g(x_k) - g(x_{k-1})$
-la comparaison entre la méthode de BFGS-DFP-SR1:
-la méthode la plus efficace c'est la méthode du BFGS car elle converge rapidement.

2.7 Technique de globalisation :

-la méthode de quasi Newton ne garantie pas toujours la convergence .
pour évaluer il y a deux méthodes :
-par évaluation d'une solution$\longrightarrow$ recherche linéaire de point x_n par la méthode de Newton
-par région de confiance $\longrightarrow$ à l'intérieur d'une sphère centré sur le point x_n de Newton et de Cauchy parx_c .

	BFGS	DFP	SR1
Matrice mise à jour	Hessien H_k	Inverse hessien H_k^{-1}	Hessien H_k
Equation résolue	Sécante	Inverse sécante	Sécante
Méthode	Broyden	Broyden	Résolution directe
Forme solution	Symétrique AA^T Rang 2 **Définie positive**	Symétrique AA^T Rang 2 **Définie positive**	Symétrique uu^T Rang 1 **Indéfinie**
Minimisation d_k	**Précision moyenne**	**Précision forte**	**Précision faible**
Fonction quadratique	Hessien exact : H_n=Q Directions conjuguées Q^{-1}	Hessien exact : H_n=Q Directions conjuguées Q	Hessien exact : H_n=Q
Limitations	Hessien de f indéfini	Hessien de f indéfini **Précision minimisation**	Matrices H_k **non définies positives**

Figure 5: la comparaison de BFGS-DFP-SR1

2.8 Problème de minimisation

Le problème écrit ci-dessous est un problème de minimisation sans contrainte

$min_{x \in R^n} f(x) \rightarrow$ gradient: $g(x) = \nabla f(x)$ et hessien : $H(x) = \nabla^2 f(x)$

Condition nécessaire de minimum local

$$\text{x minimum local} \rightarrow \begin{cases} \nabla g(x) = 0 \\ \nabla^2 g(x) \geq 0 \end{cases} \quad \text{(hessien semi-défini positif)}$$

Recherche des points stationnaires

Application de la méthode de Newton au système d'équations non linéaires $g(x) = 0$ Modèle linéaire de g en x_k : $g_k(x) = g(x) + G(x - x_k)$

avec

$$\begin{cases} g(x_k) = \nabla f(x_k) \\ G_k = \nabla g(x_k)^T = \nabla^2 g(x) = H_k \end{cases}$$

Méthode de Newton : $G = H \rightarrow$ calcul explicite du hessien à chaque itération

Méthode de quasi-Newton : G = approximation de H construite à partir des itérations précédentes sans calcul explicite du hessien.

3 Méthode de Newton

Modèle linéaire de $g = \nabla f$ en x_k
$g_k(x)= g(x) + G(x - x_k)$

avec

$$\begin{cases} g(x_k) = \nabla f(x_k) \\ G_k = \nabla g(x_k)^T = \nabla^2 g(x) = H_k \end{cases}$$

Itération : $x_{k+1} = x_k - \nabla^2 f(x_k)^-1 \nabla f(x_k) \rightarrow$ équations de Newton
Condition d'arrêt : $\|\nabla f(x_k)\| \leq \epsilon$

Difficultés
Calcul explicite et inversion du hessien $\nabla^2 f(x_k)$ à chaque itération $\rightarrow$ coûteux Convergence non garantie même près de la solution mêmes difficultés que pour la résolution d'équations
1ère condition nécessaire de minimum : $\nabla f(x) = \rightarrow$ point stationnaire $x =$ minimum local, maximum local ou point selle $\rightarrow$ 2ème condition nécessaire de minimum à vérifier : $\nabla^2 f(x) \geq 0$

Adaptations Méthodes de quasi-Newton $\rightarrow G_k =$ approximation du hessien $\nabla^2 f(x_k)$
Techniques de globalisation $\rightarrow$ Contrôle de la décroissance de f

Minimisation du modèle de f en x_k

- Conditions suffisantes de minimum local : $min_{x \in R^n} f(x) \begin{cases} \nabla f_k(x) = g_k(x) \\ \nabla^2 f_k(x) = H_k > 0 \end{cases}$

Si le hessien de f en xk est défini positif :$\nabla^2 f(x_k) > 0$ Minimisation du modèle quadratique de f en $x_k \Leftrightarrow$ Méthode de Newton en x_k pour résoudre $\nabla f(x) = 0$

- Sinon la méthode de Newton n'est pas directement applicable pour une minimisation

3.1 Exemple

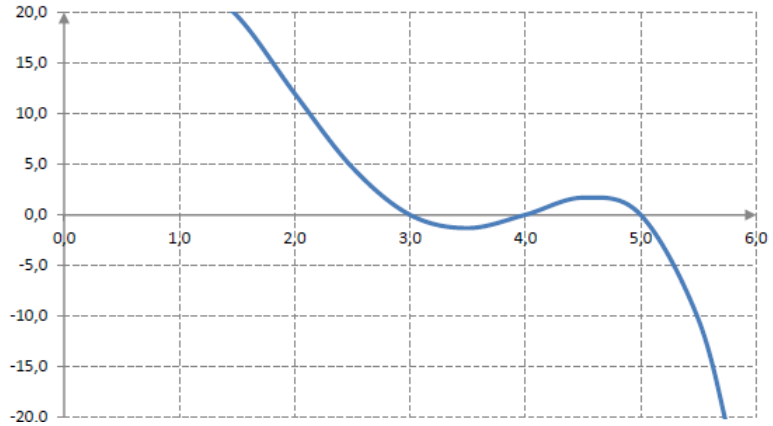
Méthode de Newton
Fonction :

$$f(x) = -x^4 + 12x^3 - 47x^2 + 60x$$

Dérivée :

$$f'(x) = -4x^3 + 36x^2 - 96x + 60$$

$\rightarrow$ 3 zéros $\rightarrow$ 1 minimum local, 2 maxima locaux

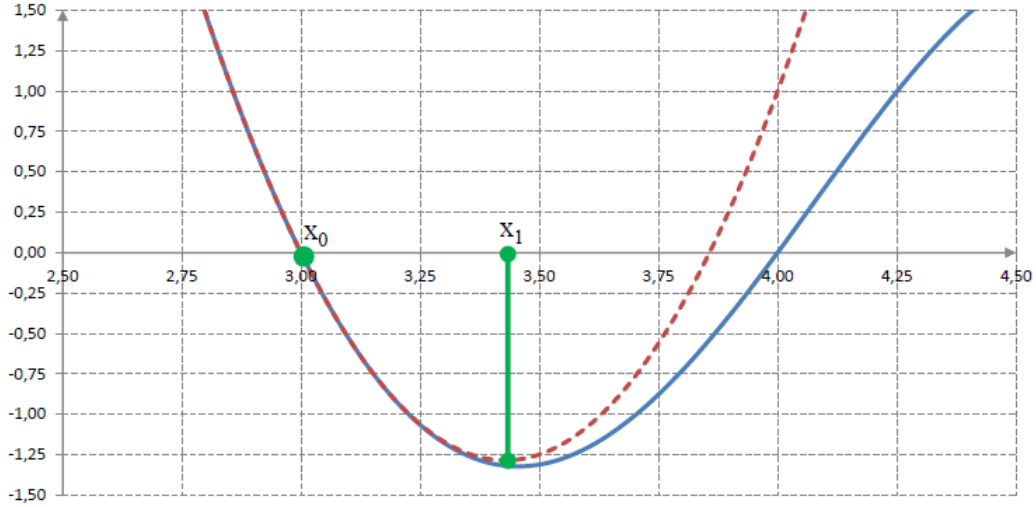


Méthode de Newton

Modèle quadratique en $x_0 = 3$:

$$f_0(x) = 7x^2 - 48x + 81$$

Itération de Newton en $x_0 = 3$: $min f_0(x) \rightarrow x_1 = \frac{24}{7} \rightarrow$ Meilleur que x_0 $f(x_1) = -1.32 < 0 = f(x_0)$



4 Méthode de quasi-Newton

La méthode de Quasi-Newton est une méthode numérique utilisée pour résoudre des systèmes d'équations non linéaires

Objectif

Conserver un modèle quadratique de f en x_k

$f_k(x) = f_k(x_k) + g_k^T(x - x_k) + \frac{1}{2}(x - x_k)^T H_k(x - x_k)$ avec $H_k \approx \nabla^2 f(x_k)$

sans calculer explicitement H_k

Mise à jour de Broyden On peut appliquer la méthode de Broyden à la résolution de $g(x) = \nabla f(x) = 0$

$H_k = H_{k-1} + \frac{(y_{k-1} - H_{k-1}d_{k-1})d_{k-1}^T}{d_{k-1}^T d_{k-1}}$ avec $\begin{cases} d_{k+1} = x_k - x_{k-1} \\ y_{k-1} = g(x_k) - g(x_{k-1} \end{cases}$

Inconvénients

H_k n'est pas forcément symétrique

H_k n'est pas forcément positive

→ Modifications de la méthode si l'on souhaite avoir $H_k \approx \nabla^2 f(x_k)$

→ Formules BFGS, DFP ou SR1

5 Méthode BFGS

La méthode BFGS est une solution souvent utilisée lorsque l'on veut un algorithme à directions de descente.

L'idée principale de cette méthode est d'éviter de construire explicitement la matrice hessienne et de construire à la place une approximation de l'inverse de la dérivée seconde de la fonction à minimiser, en analysant les différents gradients successifs. Cette approximation des dérivées de la fonction conduit à une méthode quasi-Newton (une variante de la méthode de Newton) de manière à trouver le minimum dans l'espace des paramètres.

Equation sécante

Pour la résolution d'équations, on cherche la matrice H_k

vérifiant l'équation sécante $H_k d_{k-1} = y_{k-1}$

la plus $\ll$ *proche* $\gg$ possible de H_{k-1} → formule de Broyden

sans condition particulière sur la forme de la matrice

Pour une minimisation, on impose de plus à la matrice H_k

d'être symétrique définie positive → formule BFGS

Méthode de résolution

La matrice H_{k-1} issue de l'itération précédente est symétrique, définie positive.

On part de la factorisation de Cholesky de H_{k-1} : $H_{k-1} = L_{k-1}L_{k-1}$

On cherche la matrice H_k à partir de H_{k-1} sous la forme : $H_k = A_k A_k^T$

Il faut exprimer la matrice Ak en fonction de L_{k-1}.

→ on décompose l'équation sécante en 2 équations.

$H_k d_{k-1} = d_{k-1} \Leftrightarrow A_k A_k^T d_{k-1} = y_{k-1} \Leftrightarrow \begin{cases} x = A_k^T d_{k-1} \\ A_k x = d(x_{k-1}) \end{cases}$

Pour x donné, on cherche A_k la plus $\ll$ *proche* $\gg$ possible de L_{k-1} vérifiant : $A_k x = y_{k-1}$

On détermine ensuite x en reportant l'expression de A_k dans : $x = A_k^T d_{k-1}$ On obtient $H_k = A_k A_k^T$ que l'on exprime en fonction de H_{k-1}.

Résolution Notations sans indices : $L = L_{k-1}$, $A = A_k$

$y = y_{k-1}$, $d = d_{k-1}$

En appliquant la formule de Broyden à L on obtient A en fonction de x : $A = L + \frac{(y - Lx)x^T}{x^T x}$

En reportant : $x = A^T d = L^T d + \frac{x(y-Lx)^T d}{x^T x} = L^T d + \frac{(y-Lx)^T dx}{x^T x}$

Il faut résoudre $x = L^T d + \frac{(y-Lx)^T dx}{x^T x}$ pour trouver x.

6 Comparaison

Méthodes de quasi-Newton BFGS DFP SR1

	BFGS	DFP	SR1
Matrice mise à jour	Hessien H_k	Inverse hessien H_k^{-1}	Hessien H_k
Equation résolue	Sécante	Inverse sécante	Sécante
Méthode	Broyden	Broyden	Résolution directe
Forme solution	Symétrique AA^T Rang 2 **Définie positive**	Symétrique AA^T Rang 2 **Définie positive**	Symétrique uu^T Rang 1 **Indéfinie**
Minimisation d_k	**Précision moyenne**	**Précision forte**	**Précision faible**
Fonction quadratique	Hessien exact : H_n=Q Directions conjuguées Q^{-1}	Hessien exact : H_n=Q Directions conjuguées Q	Hessien exact : H_n=Q
Limitations	Hessien de f indéfini	Hessien de f indéfini **Précision minimisation**	Matrices H_k **non définies positives**

↓

Méthode BFGS réputée la plus efficace

Comparaison Newton - Quasi-Newton

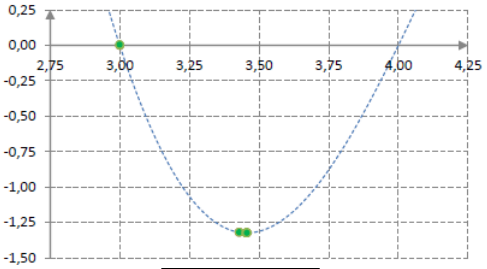
Fonction :

$$f(x) = -x^4 + 12x^3 - 47x^2 + 60x$$

Dérivée :

$$f'(x) = -4x^3 + 36x^2 - 96x + 60$$

Quasi-Newton: $h_k = \frac{f'(x_k) - f'(x_{k-1})}{x_k - x_{k-1}}$ Point initial : $x_0 = 3 \rightarrow$ convergence Autres points $\rightarrow$ divergence ou maximum de f



Quasi - Newton

Itération	x(k)	f(x)	f'(x)	h(k)	Erreur
0	3,00000000	0,00000000	-6,00E+00	1,000	-4,56E-01
1	2,99900000	0,00600700	-6,01E+00	14,000	-4,57E-01
2	3,42857155	-1,31945027	-3,15E-01	13,267	-2,70E-02
3	3,45230465	-1,32362420	-3,79E-02	11,672	-3,28E-03
4	3,45554876	-1,32368634	-4,68E-04	11,527	-4,06E-05
5	3,45558934	-1,32368635	-7,27E-07	11,509	-6,32E-08
6	3,45558940	-1,32368635	-1,40E-11	11,509	-1,22E-12
7	**3,45558940**	**-1,32368635**	**-5,68E-14**	**11,462**	**-4,44E-15**

Newton

Itération	x	f(x)	f'(x)	f''(x)	Erreur
0	3,00000000	0,00000000	-6,00E+00	14,000	-4,56E-01
1	3,42857143	-1,31945023	-3,15E-01	11,796	-2,70E-02
2	3,45526446	-1,32368574	-3,74E-03	11,513	-3,25E-04
3	3,45558935	-1,32368635	-5,77E-07	11,509	-5,01E-08
4	**3,45558940**	**-1,32368635**	**-5,68E-14**	**11,509**	**-4,88E-15**

7 TD 3 Optimisation: optimisation locale sans contraintes

Exercice 1:

On cherche à minimiser sur R^3 la fonction:

$$J(x, y, z) = x^2 + 2y^2 + z^2 - x + y - z \tag{1}$$

Pour cela on peut trouver la solution exacte au problème directement en trouvant le gradient ∇f.
Cela consiste à trouver les trois différentes dérivée selon x puis selon y et pour finir selon z :
Ensuite on doit trouver les valeur x, y et z en résolvant l'équation $\nabla f = 0$.

$$\nabla f = \begin{cases} 2x - 1 = 0 \\ 4y + 1 = 0 \\ 2z - 1 = 0 \end{cases}$$

$$\nabla f = \begin{cases} x = \frac{1}{2} \\ y = \frac{-1}{4} \\ z = \frac{1}{2} \end{cases}$$

$$HJ = \begin{cases} \frac{1}{2} \\ \frac{-1}{4} \\ \frac{1}{2} \end{cases}$$

On peut aussi utiliser une méthode de gradient avec une stratégie de type backtracking avec condition d'Armijo pour approcher la solution à partir d'un point initiale et trouver la direction de descente.
En sachant que la condition d'Armijo s'écrit:

$$J(X_0 + \alpha d) \leq J(X_0) + \beta\alpha < d, \nabla J(X_0) >$$

Par exemple le point $X_0 = \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}$
$d_k = -\nabla f(X_0)$.
On remplace ensuite les valeur x, y et z avec celle du point X_0 pour avoir d_k.On peut trouver ensuite la direction de descente à la première itération.

$$d_0 = - \begin{cases} -1 \\ 5 \\ 1 \end{cases}$$

On peut après calculer explicitement $J(X_0 + \alpha d)$ ainsi que les valeurs de α vérifiant la condition d'Armijo pour $\beta = 0.1$ Pour cela on réécrit la formule en remplaçant par les valeurs correspondantes:

$$J(X_0 + 0.1d_0) \leq 3 + 0.1 \times 0.1 \times (-27)$$

$$0,82 \leq 2,73$$

La condititon est donc vérifée.
On peut aussi trouver X_1 en ayant des constantes de backtrading fixer par exemple $\tau = 0.1$ $\alpha_{init} = 1$.

On a donc $X_1 = \begin{pmatrix} 0,1 \\ 0,5 \\ 0,9 \end{pmatrix}$

Exercice 2 : Point de Cauchy

On s'intéresse ici à un nouveau principe de recherche linéaire pour une méthode à direction de descente.
On considère f une fonction C^1 de $\mathbb{R}^n \to \mathbb{R}$, telle que $\lim_{x \to +\infty} f(x) = +\infty$.
On note g la fonction gradient de f définie de $\mathbb{R}^n \to \mathbb{R}$.
On suppose que g est Lipschitz sur $\mathbb{R}^n$avec une constante de Lipschitz égale à L.
On rappelle qu'une méthode de descente consiste à définir une suite partant d'un point $x_0 \in \mathrm{R}^n$ avec la relation :

$$x_{k+1} = x_k + t_k d_k$$

Grâce à une égalité de Taylor on a :

$$h(t_k) \leq h(t) \leq h(0) + t(d_k, g(x_k)) + t^2 L \frac{|dk|^2}{2}$$

On peut ensuite trouver les valeurs t appeler t_{min} où le terme a droite de l'inégalité est minimal ce qui signifie le plus petit possible.

$$t^2 L \frac{|dk|^2}{2} + t(d_k, g(x_k)) = 0$$

$$t_{min} = \frac{-2 \times |d_k, g(x_k))}{L d_k^2} > 0$$

Exercice 3 : Règle de Wolfe

On considère f une fonction C^1 de $\mathbb{R}^n \to \mathbb{R}$, telle que $\lim_{x \to +\infty} f(x) = +\infty$.
On note g la fonction gradient de f définie de $\mathbb{R}^n \to \mathbb{R}$.
On suppose que g est Lipschitzienne sur tout ensemble:

$S_{x_0} = x \in \mathbb{R}^n$,f(x) $\leq f(x_0)$.

Avec certaine méthode on peut trouver un minimum local x^* de f qui correspond enfaite à $g(x^*) = 0$.
Pour cela on pose A l'ordonnée d'une droite parallèle à l'axe des abscisses et on pose r la longueur quand la fonction f(x) ce trouve sous cette droite.

Démonstration:

On prend $A = f(0) + 1$
Si |x| $\geq r_0, f(x) \geq f(0) + 1$ sur B(0,r_0) compact
f allant au min x^*, sur $x \in B(0, r_0), f(x) \geq f(x^*)$
si x n'appartient pas à B $\to f(x) \geq A$, $f(x) \geq f(0) + 1$ Donc $f(x^*) + 1 \geq f(x^*)$
Donc x^* est bien un minimum global

On peut ensuite crée une suite $(x_k)_{k \in n}$ convergent vers un minimum global ou local de f.
On peut la définir avec $x_0 \in \mathrm{R}^n$ et avec la relation $x_{k+1} = x_k + t_k d_k$.
Avec d_k la direction de descente et t_k le pas dans cette direction supposé satisfaire la condition.

$$q(t_k) \leq q(0) + m_1 t_k q'(0) et q'(t_k) \geq m_2 q'(0)$$

Avec q(t) = f(x_k + td_k) et où m_1 et m_2 sont deux réels tels que:
$0 < m_1 < m_2 < 1$.
On peut ensuite trouver un exemple de t_k admissible
Ceci est la règle de wolfe qui contrairement à la condition d'armijo elle utilise 2 critères avec en plus le critère de courbure avec deux paramètre m_1 et m_2 m $\in]0; 1[$.

8 Programmation

Voici un exemple de programme écrit sur matlab d'optimisation sans contrainte: Le but de ce celui ci est de trouver un minimum local d'une fonction tout en sachant qu'il ne sera pas forcement un minimum global.

```
Annexe 1 Minimisation de la fonction Rosenbrock (Matlab)\\

syms x y
format short


%fonction consideree
f(x,y) =  100*(y-(x^2))^2+(x-1)^2 ;


%gradient de la fonction
g1 = diff(f,x) ;
g2 = diff(f,y) ;
grad = gradient(f(x,y),[x,y]) ;
g(x,y) = gradient(f(x,y),[x,y]) ;


%donnees du probleme
x0 = [0,1] ;
x1 = [0,1] ;
alphainit = 1 ;
beta = 0.1 ;
tau = 0.3 ;
N = 100;


%calcul du point critique avec le gradien
for i = 1:N
    a = alphainit ;
    d = -g(x0(1),x0(2)) ;
    t = transpose(d) ;
    while f(x0(1)+a*d(1),x0(2)+a*d(2)) > f(x0(1),x0(2))+beta*a*t*g(x0(1),x0(2))
        a = a*tau ;
    end
    x0(1) = x0(1)+a*d(1);
    x0(2) = x0(2)+a*d(2);
end
xfinal0 = x0


%lignes de niveau de la fonction et points solution
r = 1:10 ;
fimplicit(f(x,y)==r)
grid on
```

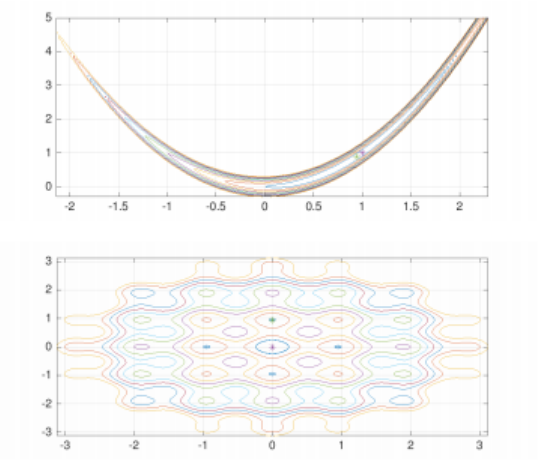
```
hold on
plot(0,0,'*');
plot(xfinal0(1),xfinal0(2),'*')
hold off
```

Annexe 2 Minimisation de la fonction de Rastrigin (Matlab)\\

```
syms x y
format short


%fonction consideree
f(x,y) = x^2 + y^2 -cos(2*pi*x) -cos(2*pi*y) +2 ;


%gradient de la fonction
g1 = diff(f,x) ;
g2 = diff(f,y) ;
grad = gradient(f(x,y),[x,y]) ;
g(x,y) = gradient(f(x,y),[x,y]) ;


%donnees du probleme
x0 = [0,1] ;
x1 = [0,1] ;
alphainit = 1 ;
beta = 0.1 ;
tau = 0.3 ;
N = 100;


%calcul du point critique avec le gradien
for i = 1:N
    a = alphainit ;
    d = -g(x0(1),x0(2)) ;
    t = transpose(d) ;
    while f(x0(1)+a*d(1),x0(2)+a*d(2)) > f(x0(1),x0(2))+beta*a*t*g(x0(1),x0(2))
        a = a*tau ;
    end
    x0(1) = x0(1)+a*d(1);
    x0(2) = x0(2)+a*d(2);
```

```
end
xfinal0 = x0


%lignes de niveau de la fonction et points solution
r = 1:10 ;
fimplicit(f(x,y)==r)
grid on
hold on
plot(0,0,'*');
plot(xfinal0(1),xfinal0(2),'*')
hold off
```

Il est bon de savoir que plusieurs méthodes d'algorithme existe afin d'atteindre notre but tel que la méthode de Newton, de quasi Newton, du trapèze, de simpsons etc...
Dans cette algorithme le gradient de la fonction est calculer puis la valeur initiale x_0 est choisit aléatoirement .Par itération ainsi que par la méthode de plus forte pente, une condition d'armijo et un backtracking nous atteignons notre objectif. Celui ci prend fin si l'utilisateur a posé un nombre d'itérations max ou autre conditions tel qu'un gradient faible.
Les programmes qui précèdes sont utilisés afin d'avoir un minimum local dans la fonction de Rosenbrock et celle de Rastrigin. Ils s'arrêtent après 100 itérations.



Le programme fonctionne correctement et l'on peut voir le résultat obtenu pour la fonction de Rastrigin. Il est a noté que la méthode doit être adaptée en fonction du problème.

Cours Partie 4

Laura Martin
Mathilde Thollet

18 décembre 2018

1 Principe de Newton

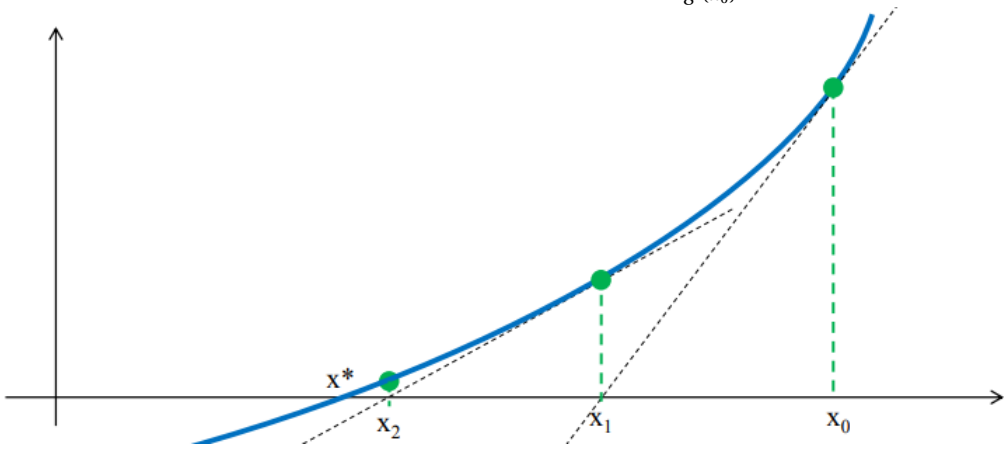
On va chercher à résoudre efficacement la condition necessaire 1 (CN1) d´optimalité $\nabla f = 0$ où $f : \mathbb{R}^n \to \mathbb{R}^n$ est C^2, tout en essayant de s´assurer qu´il s´agit bien d´un minimum local. En particulier, la méthode sera utilisée pour rechercher un minimum local vérifiant la CN2 : $H_f(x^*) >> 0$.

Méthode de Newton "usuelle"

Le problème est le suivant :
On cherche à resoudre $g(x) = 0$ où $g : \mathbb{R}^n \to \mathbb{R}^n$ est C^1. Ce système à n equations, n inconnues et est non linéaire. Dans le cas $\mathbb{R}^n = \mathbb{R}$, on l´appelle dichotomie.

Illustration à une variable

On résoud $g(x) = 0$: Le point initial est x_0 et $y_0 = g(x_0)$. La sécante en x_0 est donnée par $y = y_0 + g'(x_0)(x - x_0)$. Donc si on cherche le point où $y = 0$ on obtient $x_1 = x_0 - \frac{y_0}{g'(x_0)}$ et on peut répéter le processus.



On construit une suite d´approximations $(x_k)_{k\in\mathbb{N}}$ tel qu´on a un point initial donné x_0 et une approximation linéaire $g'(x) = g(x_k) + \nabla g(x_k)(x - x_k)$ à la fonction ∇f autour le point x_k, où

$$\begin{cases} g(x_k) = \nabla f(x_k) \in \mathbb{R}^n \\ \nabla g(x_k) = H_f(x_k) \in \mathbb{M}_n(R) \quad \text{(notation: } G_k = \nabla g(x_k)) \end{cases} \tag{1}$$

On suppose G_k inversible. G_k prend alors x_{k+1} tel que

$$g'(x_{k+1}) = 0 \quad \Longrightarrow \quad x_{k+1} = x_k - G_k^{-1} g(x_k)$$

Notre condition d'arrêt sera $\| g(x_k) \| < \epsilon$

Avantages	**Inconvénients**
·Rapidité de convergence	·Calcul et inversion de G_k
(quadratique si $\nabla g(x^*)$inversible)	· Convergence non assurée
	·Caractére inestable par rapport à x_0

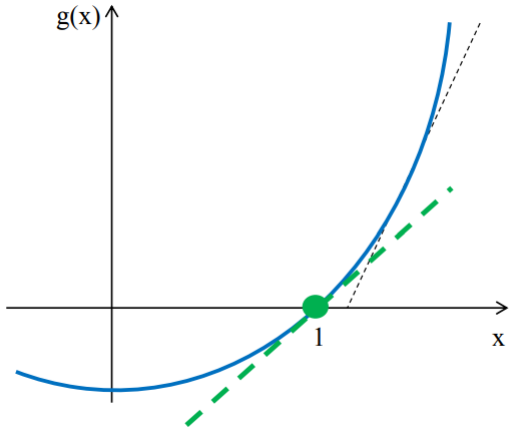
(2)

On adapte la méthode de Newton pour éviter les inconvénients et on obtient la méthode de Quasi-Newton.

Exemples

Exemple 1

$$\begin{cases} \text{Fonction:} & g(x) = x^2 - 1 \\ \text{Derivée:} & g'(x) = 2x \\ \text{Solution:} & x = 1 \rightarrow \mathbf{g'(1)=2} \end{cases} \quad (3)$$

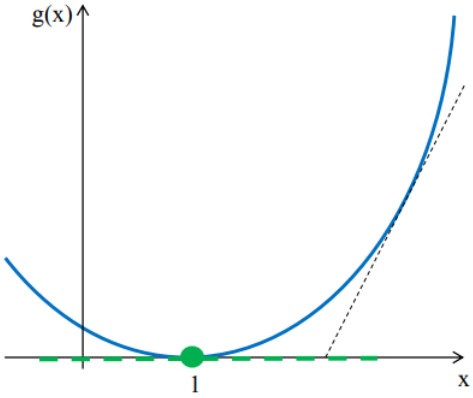


Iteration	x(k)	g(x)=x**2-1	g'(x)=2x	Erreur
0	4,00000000	1,5E+01	8,0000	3,0E+00
1	2,12500000	3,5E+00	4,2500	1,1E+00
2	1,29779412	6,8E-01	2,5956	3,0E-01
3	1,03416618	6,9E-02	2,0683	3,4E-02
4	1,00056438	1,1E-03	2,0011	5,6E-04
5	1,00000016	3,2E-07	2,0000	1,6E-07
6	**1,00000000**	**2,5E-14**	**2,0000**	**1,3E-14**

On peut voir que la convergence est quadratique, donc g' est inversible.

Exemple 2

$$\begin{cases} \text{Fonction:} & g(x) = (x-1)^2 \\ \text{Derivée:} & g'(x) = 2(x-1) \\ \text{Solution:} & x = 1 \rightarrow \mathbf{g'(1)=0} \end{cases} \quad (4)$$

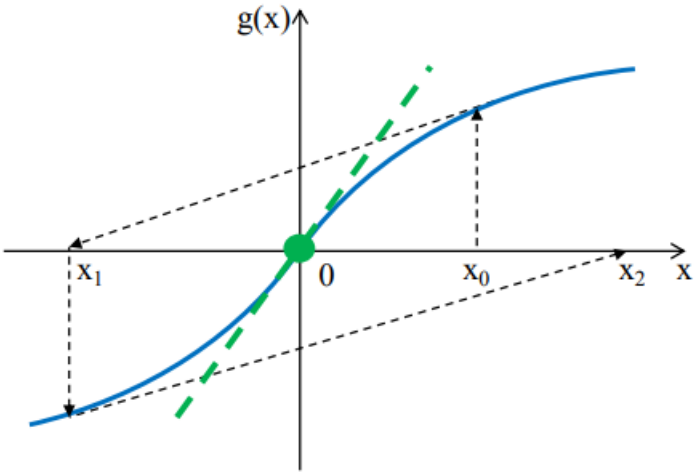


Iteration	x(k)	g(x)=(x-1)**2	g'(x)=2(x-1)	Erreur
0	4,00000000	9,0E+00	6,0000	3,0E+00
1	2,50000000	2,3E+00	3,0000	1,5E+00
2	1,75000000	5,6E-01	1,5000	7,5E-01
3	1,37500000	1,4E-01	0,7500	3,8E-01
4	1,18750000	3,5E-02	0,3750	1,9E-01
5	1,09375000	8,8E-03	0,1875	9,4E-02
6	1,04687500	2,2E-03	0,0938	4,7E-02
7	1,02343750	5,5E-04	0,0469	2,3E-02
8	1,01171875	1,4E-04	0,0234	1,2E-02
9	1,00585938	3,4E-05	0,0117	5,9E-03
10	1,00292969	8,6E-06	0,0059	2,9E-03
15	1,00009155	8,4E-09	0,0002	9,2E-05
20	**1,00000286**	**8,2E-12**	**0,0000**	**2,9E-06**

Ici la convergence est lente, donc g' n'est pas inversible.

Exemple 3

$$\begin{cases} \text{Fonction:} & g(x) = Arctan(x) \\ \text{Derivée:} & g'(x) = \frac{1}{1+x^2} \\ \text{Solution:} & x = 0 \rightarrow \textbf{g"(1)=0} \end{cases} \quad (5)$$



Iteration	x(k)	g(x)=Arctan(x)	g'(x)=1/(1+x**2)	Erreur
0	**1,300**	**0,915**	**0,372**	**1,3E+00**
1	-1,162	-0,860	0,426	-1,2E+00
2	0,859	0,710	0,575	8,6E-01
3	-0,374	-0,358	0,877	-3,7E-01
4	0,034	0,034	0,999	3,4E-02
5	0,000	0,000	1,000	-2,6E-05
6	**0,000**	**0,000**	**1,000**	**1,2E-14**

Convergence

Iteration	x(k)	g(x)=Arctan(x)	g'(x)=1/(1+x**2)	Erreur
0	**1,500**	**0,983**	**0,308**	**1,5E+00**
1	-1,694	-1,038	0,258	-1,7E+00
2	2,321	1,164	0,157	2,3E+00
3	-5,114	-1,378	0,037	-5,1E+00
4	32,296	1,540	0,001	3,2E+01
5	-1575,317	-1,570	0,000	-1,6E+03
6	**38)4976,008**	**1,571**	**0,000**	**3,9E+06**

Divergence

La convergence depend du point initial.

2 Méthode quasi Newton

On cherche toujours à résoudre l'equation $g = 0$. Mais cette fois, afin de s'affranchir du calcul de $\nabla g(x_k)$, on définit une nouvelle matrice proche de $\nabla g(x_k)$, inversible, encore notée G_k.

C'est la méthode de Broyden.

On prend $G_0 = I_d$ et on cherche à définir G_k en fonction de $G_{k-1}, g(x_k), g(x_{k-1}), x_k$ et x_{k-1}.

Variation du modèle :

On a le modèle linéaire de g en x_{k-1} :$g'_{k-1}(x) = g(x_{k-1}) + G_k(x - x_{k-1})$
Et le modèle linéaire de g en x_k :$g'_k(x) = g(x_k) + G_k(x - x_k)$ Donc on a :

$$g'_k(x) = g(x_k) + G_k(x - xk)$$

or $g(x_k) = g(x_{k-1}) + G_k(x_k - x_{k-1})$ par définition de G_k, donc

$$g'_k(x) = g(x_{k-1}) + G_k(x_k - x_{k-1}) + G_k(x - x_k)$$

or $g'_{k-1}(x) = g(x_{k-1}) + G_{k-1}(x - x_{k-1})$ par définition de g'_{k-1}, donc

$$g'_k(x) = g'_{k-1}(x) - G_{k-1}(x - x_{k-1}) + G(x - x_{k-1})$$

c'est-à-dire $g'_k(x) = g'_{k-1}(x) + (G_k - G_{k-1})(x - x_{k-1})$ donc on a $g'_k(x) - g'_{k-1}(x) = (G_k - G_{k-1})(x - x_{k-1})$
L'objectif est de conserver un modèle quadratique de g en x_k :

$$g'_k = g_k(x_k) + g_k^T(x - x_k) + \frac{1}{2}(x - x_k)^T G_k(x - x_k)$$

avec $G_k \approx \nabla^2 g(x_k)$ sans calculer explicitement G_k.

Et on impose que G_k vérifie l'equation de la sécante entre x_k et x_{k-1} :

$$g(x_k) - g(x_{k-1}) = G_k \cdot (x_k - x_{k-1})$$

où

$$G_k \in \mathbb{M}_n(\mathbb{R}^n) \iff y_{k-1} = G_k d_{k-1} \begin{cases} d_{k-1} = x_k - x_{k-1} \in \mathbb{R}^n \\ y_{k-1} = g(x_k) - g(x_{k-1}) \in \mathbb{R}^n \end{cases}$$

Il existe une infinité de matrice G vérifiant l'equation de la sécante. On a donc la formule de Broyden :

$$G_k = G_{k-1} + \frac{(y_{k-1} - G_{k-1} d_{k-1}) d_{k-1}^T}{d_{k-1}^T d_{k-1}}$$

On applique maintenant ces méthodes à un problème de minimisation sans contraite :

$$(PO) : \underset{x \in \mathbb{R}^n}{min} f(x)$$

3 Application au problème de minimisation

On cherche à nouveau à minimiser (localement) une fonction $f : \mathbb{R}^n \to \mathbb{R}, C^2$.
On pose le gradient $g(x) = \nabla f(x)$ et le hessien $H(x) = \nabla^2 f(x)$.
Si on applique la mèthode de Newton précedente à $g(x) = \nabla f(x)$, on trouve la mèthode suivante (Newton) :

$$\begin{cases} x_0 \in \mathbb{R}^n \\ x_{k+1} = x_k - H_f^{-1}(x_k) \nabla f(x) \quad (si H_f^{-1}(x_k) inversible) \end{cases} \quad (6)$$

Afin d´avoir un espoir de convergence vers un minimun de f, on se place dans la situation où x^* verifie la CN2 : $H_f(x^*) >> 0$. Dans ce cas, il existe $r > 0$ tel que partout $x \in B(x^*, r), H_f(x) >> 0$. On essaiera de choisir $x_0 \in B(x^*, r)$. Dans ce cas, si $x_k \in B(x_0, r), d_k = -H_f(x_k) \nabla f(x_k)$ est une direction de descente :

$$-(\nabla f(x_k))^T (H_f(x_k) \nabla f(x_k)) < 0 \qquad (si \nabla f(x_k) \neq 0)$$

En général, il s´agit d´une méthode de descente ($f(x_{k+1}) < f(x_k)$) qui ne necessite pas de recherche linéaire. On peut aussi adapter la méthode de type Quasi-Newton BFGS au cas présent ($g(x) = \nabla f(x)$). Cette méthode construit une suite $(H_k)_{k \in N}$ de matrices symétriques, définies positives sous certaines conditions, approchant $H_f(x_k)$ sans avoir à le calculer. On peut éventuellement améliorer sa robustesse en rajoutent une strategie de recherche linéaire.

On a x^* minimum local $\Rightarrow g(x^*) = 0$ et $H(x^*) \geq 0$. C'est une condition nécessaire de minimum local.
On applique la méthode de quasi-Newton à la fonction $\nabla f(x)$ pour résoudre $g(x) = \nabla f(x) = 0$.
On peut appliquer la méthode de Broyden à la résolution de ce problème mais alors H_k n'est pas forcément symétrique, ni positive.
Il faut donc modifier la méthode et utiliser d'autres formules : BFGS, DFP ou SR1.
Ici, on va présenter la méthode BFGS qui impose de plus que H_k est symétrique et définie positive.
La méthode BFGS est réputée la plus efficace. Elle a été élaborée par Broyden, Fletcher, Goldfarb et Shanno à la fin des années 1960.

Méthodes de quasi-Newton BFGS – DFP – SR1

	BFGS	DFP	SR1
Matrice mise à jour	Hessien H_k	Inverse hessien H_k^{-1}	Hessien H_k
Equation résolue	Sécante	Inverse sécante	Sécante
Méthode	Broyden	Broyden	Résolution directe
Forme solution	Symétrique AA^T Rang 2 **Définie positive**	Symétrique AA^T Rang 2 **Définie positive**	Symétrique uu^T Rang 1 **Indéfinie**
Minimisation d_k	**Précision moyenne**	**Précision forte**	**Précision faible**
Fonction quadratique	Hessien exact : H_n=Q Directions conjuguées Q^{-1}	Hessien exact : H_n=Q Directions conjuguées Q	Hessien exact : H_n=Q
Limitations	Hessien de f indéfini	Hessien de f indéfini **Précision minimisation**	Matrices H_k **non définies positives**

↓

Méthode BFGS réputée la plus efficace

Méthode de résolution :

La matrice H_{k-1} est symétrique, définie positive.
• On part de la factorisation de Cholesky de H_{k-1} sous la forme $H_{k-1} = L_{k-1}L_{k-1}^T$.
• On cherche H_k à partir de H_{k-1} sous la forme $H_k = A_k A_k^T$.
On veut exprimer A_k en fonction de L_{k-1}.
On décompose l'equation de la sécante en deux equations :

$$H_k d_{k-1} = y_{k-1} \iff A_k A_k^T d_{k-1} = y_{k-1} \iff x = A_k^T d_{k-1} et A_k x = y_{k-1}$$

• Pour un x donné, on cherche A_k la plus proche de L_{k-1} vérifiant : $A_k x = y_{k-1}$ et $x = A_k^T d_{k-1}$
• On détermine ensuite x en reportant l'expression de A_k dans $x = A_k^T d_{k-1}$
• On obtient $H_k = A_k A_k^T$ que l'on exprime en fonction de H_{k-1}.
On obtient finalement la formule BFGS :

$$H_k = H_{k-1}\frac{y_{k-1}y_{k-1}^T}{y_{k-1}^T d_{k-1}} - \frac{H_{k-1}d_{k-1}d_{k-1}^T H_{k-1}}{d_{k-1}^T H_{k-1} d_{k-1}}$$

avec $\begin{cases} d_{k-1} = x_k - x_{k-1} \\ y_{k-1} = g(x_k) - g(x_{k-1}) \end{cases}$ et H_k symétrique et définie positive.

BFGS appliqué à la méthode de quasi-Newton :

Le déplacement correspondant à une itération de la méthode de Newton doit vérifier :

$$H_k d_k = -\nabla f(x_k) \Rightarrow d_k = -H_k^{-1}\nabla f(x_k)$$

On a besoin de H_{k-1}^{-1} pour l'itération de Newton :
On l'obtient en inversant les deux membres de la formule BFGS :

$$H_{k-1}^{-1} = (I - \frac{d_{k-1}y_{k-1}^T}{d_{k-1}^T y_{k-1}})H_{k-1}^{-1}(I - \frac{y_{k-1}d_{k-1}^T}{d_{k-1}^T y_{k-1}}) + \frac{d_{k-1}d_{k-1}^T}{d_{k-1}^T y_{k-2}}$$

si $y_{k-1}^T d_{k-1} > 0$ avec $\begin{cases} d_{k-1} = x_k - x_{k-1} \\ y_{k-1} = g(x_k) - g(x_{k-1}) \end{cases}$

Convergence de BFGS :

Si le hessien n'est pas définit positif, la méthode BFGS ne converge pas vers hessien.

Propriété 1 :

BFGS donne une matrice définie positive si $d_{k-1}^T y_{k-1} > 0$.
Cette condition est réalisée si x_k est obtenu par minimisation exacte dans la direction $-H_{k-1}^{-1} g(x_k)$.

Propriété 2 :

Pour une fonction f quadratique $f(x) = \frac{1}{2} s^T Q x + c^T x$. L'algorithme BFGS donne des directions successives vérifiant :

$$\begin{cases} u_i Q^{-1} u_j = 0 & 0 \leq i \neq j \leq k \\ H_k^{-1} G^{-1} u_i = u_i & 0 \leq i \leq k \end{cases}$$

On obtient des directions conjuguées par rapport à G^{-1} et une convergence en n itérations avec l'itération n : $H_n = Q$

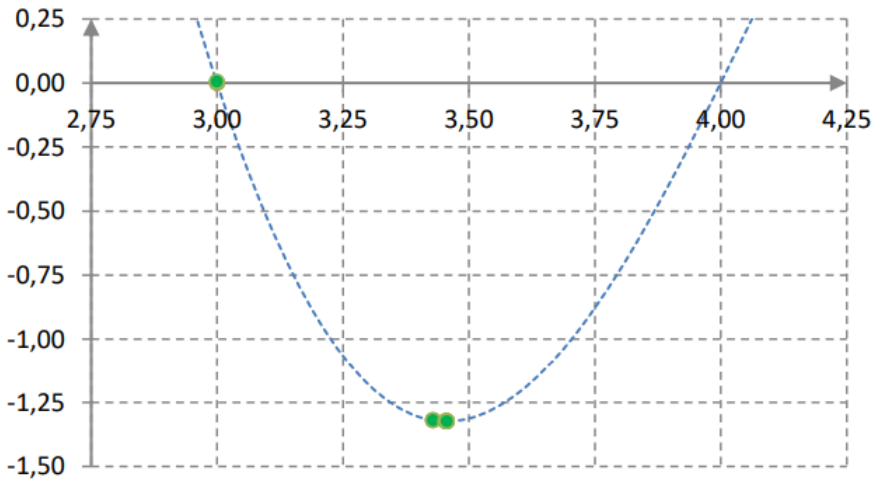
Méthode de quasi-Newton à une variable :

La formule BFGS se simplifie pour une fonction à une variable : $f(x), x \in \mathbb{R}$
Alors BFGS devient : $H_k = \frac{y_{k-1}}{d_{k-1}} = \frac{g(x_k) - g(x_{k-1})}{x_k - x_{k-1}}$
On retrouve la formule de la sécante appliquée au gradient de f.

3.1 Comparaison Newton et Quasi-Newton

$$\begin{cases} \text{Fonction:} & f(x) = -x^4 + 12x^3 - 47x^2 + 60x \\ \text{Derivée:} & f'(x) = -4x^3 + 36x^2 - 94x + 60 \\ \text{Quasi-Newton:} & h_k = \frac{f'(x_k) - f'(x_{k-1})}{x_k - x_{k-1}} \\ \text{Point initial:} & x_0 = 3 \end{cases} \quad (7)$$

Si on prend $X_0 = 3$, alors Quasi-Newton converge. Sinon, si le point initial est différent de 3, soit la méthode diverge soit on obtient un maximum de f.



Quasi - Newton

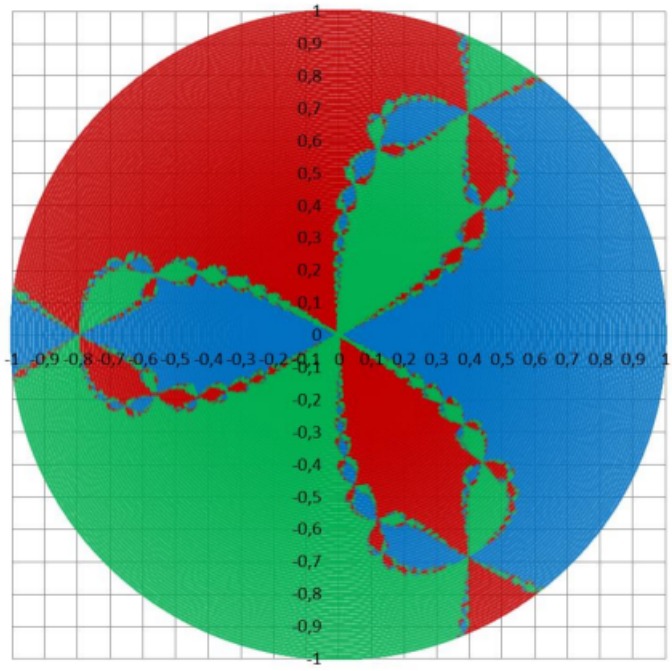
Iteration	x(k)	g(x)=x**2-1	dg/dx	Erreur
	5,00000000	2,4E+01		4,0E+00
0	4,00000000	1,5E+01	9,0000	3,0E+00
1	2,33333333	4,4E+00	6,3333	1,3E+00
2	1,63157895	1,7E+00	3,9649	6,3E-01
3	1,21238938	4,7E-01	2,8440	2,1E-01
4	1,04716672	9,7E-02	2,2596	4,7E-02
5	1,00443349	8,9E-03	2,0516	4,4E-03
6	1,00010193	2,0E-04	2,0045	1,0E-04
7	1,00000023	4,5E-07	2,0001	2,3E-07
8	**1,00000000**	**2,3E-11**	**2,0000**	**1,1E-11**

Newton

Iteration	x(k)	g(x)=x**2-1	g'(x)=2x	Erreur
0	4,00000000	1,5E+01	8,0000	3,0E+00
1	2,12500000	3,5E+00	4,2500	1,1E+00
2	1,29779412	6,8E-01	2,5956	3,0E-01
3	1,03416618	6,9E-02	2,0683	3,4E-02
4	1,00056438	1,1E-03	2,0011	5,6E-04
5	1,00000016	3,2E-07	2,0000	1,6E-07
6	**1,00000000**	**2,5E-14**	**2,0000**	**1,3E-14**

Convergence

La méthode de Newton est très sensible au point initial. Une modification du point initial peut conduire à une solution différente ou à une divergence. Par exemple avec l'équation complexe $z^3 = 1$, on voit sur le graphique les trois couleurs qui représentent la racine vers laquelle la méthode converge en chaque point.



TP 1 : Optimisation locale sans contraintes

L'objectif de cette séance est d'utiliser le logiciel Scilab (ou Matlab, ou Python) afin de comparer différents algorithmes de recherche linéaire pour la minimisation locale d'une fonction définie sur R^n.

Exercice 1

1.

On cherche à programmer tout d´abord la méthode du gradient avec une stratégie de type backtracking avec condition d´Armijo. On rappelle que la condition d´Armijo pour un point de départ X_0 et une direction de descente d s´écrit :

$$J(X_0 + \alpha_d) \leq J(X_0) + \beta\alpha < d, \nabla J(X_0) > (*)$$

2.

Ecrire une fonction Scilab ayant pour arguments $J, \nabla J, \beta, \alpha_{init}, \tau$ (paramètres du procédé de backtracking), X_0 (point initial) et N et renvoyant la valeur de X_N.

Resolution

```
function XN=GradEx1(J,nablaJ,X0,beta,alphainit,tau,N)
XN=X0;
for i=1:N
d=-nablaJ(X0)
alpha=alphainit
while (J(X0+alpha*d)>J(X0)-beta*alpha*d'*d)
//on choisit alpha s'il verifie la équation (*)
      alpha=alpha*tau
//tau<1 donc le nouveau alpha est plus petit: blacktracking
end
X0=X0+alpha*d
XN=[XN,X0]; // concatenation
end
endfunction
```

On choisit $d = -\nabla J(X_0)$ parce qu´on sait qu'il est une direction de descente. $\alpha = \alpha_{init}$ pour garder l'information initial sur α_{init}. Le boucle *while* arrête quand on atteint un bon alpha (c´est à dire $J(X_0 + \alpha_d) \leq J(X_0) + \beta\alpha < d, \nabla J(X_0) >$). Sinon, on diminue α. À la fin, on garde X_0 comme $X_0 + \alpha d$. On garde tous les X_0 dans X_N.

En plusieurs dimensions

Si on veut travailler cet méthode dans trois dimensions, c'est suffisant de mettre un autre vecteur pour garder les valeurs de y dans le programme, mais la structure est la même.

```
function [XN,YN,F]=GradEx1q3(J,nablaJ,X0,Beta,alphainit,tau,N)
    XN=zeros((1,100))
    YN=zeros((1,100))
    F=zeros((1,100))
         XN(1,1)=X0(1)
         YN(1,1)=X0(2)
         F(1,1)=J(X0)
    for i=2:N
        d=-nablaJ(X0)
```

```
        alpha=alphainit
        while J(X0+alpha*d)>J(X0)+Beta*alpha*d'*nablaJ(X0)
            alpha=alpha*tau
         end
         X0=X0+alpha*d
         XN(1,i)=X0(1)
         YN(1,i)=X0(2)
         F(1,i)=J(X0)
     end
endfunction
```

3.

On cherche à minimiser la fonction de Rosenbrock en dimension 2 :

$$J(x,y) = 100(y-x^2)^2 + (x-1)^2$$

On prend pour cela $\beta = 0.1, \alpha_{init} = 1, \tau = 0.3, X_0 = (0,1)$. Représenter graphiquement sur deux figures séparées :
(i) les lignes de niveau de la fonction de Rosenbrock sur $[-1,2]^2$ et les 100 premières itérations de l´algorithme précédent.
(ii) la fonction $N \to J(X_N)$ pour $N \in 0,...,100$

Resolution

Pour le point (i), il y a deux formes de reponses. Le premier s´obtient quand on a l'égalité $J(x,y) = C(cte)$. Avec cette information, on obtient un polinôme de degré 2 sur y qui depend de x, donc on a deux solutions pour y : *niveauplus* et *niveaumoins*. Dans le boucle, donné un vecteur $\mathbf{x}$ et la constant C qui change avec chaque iteraction, on obtien les lignes de niveaux $Y1$ et $Y2$ correspondant aux constantes C.

```
h=0.05
x=(-1:h:2)

function Y1=niveauplus(x,C)
    Y1=x.^2+(1./10)*sqrt(C-(x-1).^2)
endfunction

function Y2=niveaumoins(x,C)
    Y2=x.^2-(1./10)*sqrt(C-(x-1).^2)
endfunction

for i=-10:100:1000
    Y1=niveauplus(x,i)
    Y2=niveaumoins(x,i)
    plot2d(x,Y1)
    plot2d(x,Y2)
end
```

La deuxième option est de travailler avec la fonction de Scilab *contour2di*.

```
Nt=100;
x=linspace(-1,2,Nt);
y=linspace(-1,2,Nt);
z=zeros(Nt,Nt);
for i=1:Nt
    for j=1:Nt
```

```
       z(i,j)=rose([x(i),y(j)])
    end
end

xset('window',1)
clf()
contour(x,y,z,[0:10])
```

Pour obtenir les 100 premières itérations, c'est suffisant d´appeler la fonction avec les valeurs correspondantes et N=100 qui indiquent combien d'éléments il va nous donner. Avant tout, on écrit les fonctions Rosenbrock et son gradient.

```
function y=rose(x)
    y=100*(x(2)-x(1)^2)^2+(x(1)-1)^2
endfunction

function y=gradrose(x)
    y=[-400*x(1)*(x(2)-x(1)^2)+2*(x(1)-1);
    200*x(1)*(x(1)-x(2)^2)]
endfunction

X0=[0;1]
Beta=0.1
alphainit=1
tau=0.3
N=100;
XN=GradEx1(rose,gradrose,X0,Beta,alphainit,tau,N)
//[XN,YN,F]=GradEx1g3(rose,gradrose,X0,Beta,alphainit,tau,N) pour plusiers dimensions
```

Pour le point (ii), on va utiliser le vecteur X_N obtenu avant.

```
J=[];
for i=1:N+1
J=[J,rose(XN(i))];
end
```

Exercice 2

On souhaite remplacer l´algorithme de backtracking par un nouveau procédé de dichotomie où le second critère (en plus du critère dArmijo) est donné par la condition :

$$J(X_0 + \alpha_d) > J(X_0) + \beta_2 \alpha < d, \nabla J(X_0) >$$

avec $0 < \beta_2 < \beta$. Ce second critère (condition de Goldstein) permet de sélectionner un pas suffisamment grand. Partant dun intervalle $[0, \alpha_{max}]$ où α_{max} ne satisfait pas la condition dArmijo, proposer et implémenter avec Scilab un algorithme permettant de trouver un pas convenable et comparer avec la solution précédente. On pourra prendre $\beta_2 = 0.001$.

Resolution

Cette méthode évite d´évaluer ∇J beaucoup de fois, et nous permet de prendre un α suffisemment grand parce qu´on délimite la zone admissible ci-dessous. Donc on cherche un α qui vérifie :

$$\beta < \frac{J(X_0 + \alpha d) - J(X_0)}{\alpha < d, \nabla J(X_0)} < \beta_2$$

Qui est équivalent à :

$$\beta\alpha < d, \nabla J(X_0) < J(X_0 + \alpha d) - J(X_0) < \beta_2 \alpha < d, \nabla J(X_0)$$

```
function XN=GradEx2(J,nablaJ,X0,Beta,Beta2,alphainit,tau1,tau2,N)
    for i=1:N
        d=-nablaJ(X0)
        alpha=alphainit
        continuer = %T
        n=0;
        while continuer && n<1000
            continuer = %F
            n=n+1
            if(Beta*alpha*d'*gradrose(X0)<rose(X0+alpha*d)-rose(X0))
                alpha=alpha*tau1
                continuer = %T
            end
            if(Beta2*alpha*d'*gradrose(X0)>rose(X0+alpha*d)-rose(X0))
                alpha=alpha*tau2 //pour rester au bord du intervale valide
                continuer = %T
            end
         end
         X0=X0+alpha*d
     end
     XN=X0
endfunction
```

Exercice 3

On remplace la condition (2) par la nouvelle condition (dite de Wolfe) :

$$\varphi'(\alpha) > \beta_3 \varphi'(0)$$

avec $\varphi(\alpha) = J(X_0 + \alpha_d)$ et où $0 < \beta_3 < \beta$. Reprendre l´algorithme de dichotomie construit dans lexercice 2 et comparer les résultats obtenus avec cette nouvelle méthode (avec $\beta_3 = 0.001$).

Resolution

On change le deuxième *if* avec la condition donnée sur l´énnoncé.

```
function XN=GradEx3(J,nablaJ,X0,Beta,Beta3,alphainit,tau1,tau2,N)
    for i=1:N
        d=-nablaJ(X0)
        alpha=alphainit
        continuer = %T
        n=0;
        while continuer && n<1000
            continuer = %F
            n=n+1
            if(Beta*alpha*d'*nablaJ(X0)<J(X0+alpha*d)-J(X0))
                alpha=alpha*tau1
                continuer = %T
            end
            if(nablaJ(X0)>Beta3*nablaJ([0,0]))
                alpha=alpha*tau2
```

```
                continuer = %T
            end
        end
        X0=X0+alpha*d
    end
    XN=X0
endfunction
```

Exercice 4

En suivant la même démarche que dans l´exercice 1, écrire une nouvelle fonction Scilab, utilisant la méthode de Newton pour minimiser une fonction J et la tester sur le même exemple (fonction de Rosenbrock). La recherche linéaire est alors supprimée (prendre $\alpha = 1$) et un nouvel argument apparait, en loccurrence HJ le Hessien de J.

Resolution

On commence avec le Hessien de J

```
function Z=hessrose(X)
    x=X(1);y=X(2);
    Z=[-400*y+1200*x^2+2, -400*x;-400*x,200];
endfunction
```

On appelle $g(x) = \nabla J(x)$ (donc $g'(x) = H_J(x)$) et on obtient l´équation de la droite tangente $Y = g(X_k) + g'(X_k) * (x - X_k)$. Pour la méthode de Newton on veut chercher le point où $y = 0$, c'est à dire

$$0 = g(X_k) + g'(X_k) * (X_{k+1} - X_k) \Rightarrow X_{k+1} = X_k - g'^{-1}(X_k)g(X_k)$$

```
function XN=NewEx4(J,nablaJ,HJ,X0,N)
XN=X0;
for i=1:N
d=-inv(HJ(X0))*nablaJ(X0)
X0=X0+d;
XN=[XN,X0];
end
endfunction
```

Exercice 5

Même exercice que l´exercice 4, avec la méthode BFGS (à noter que l´argument HJ disparait). Conclure sur l´efficacité comparée des méthode de gradient, Newton et BFGS.

Resolution

Avec la méthode BFGS on évite de calculer l'inverse de H_J avec une approximation de celle çi.
La structure du programme c'est comme dans le exercice 4 mais avec la construction de la matrice avant,

$$G_k = G_{k-1} + \frac{y'_{k-1}y_{k-1}}{y'_{k-1}d_{k-1}} - \frac{G_{k-1}d_{k-1}d'_{k-1}G_{k-1}}{d'_{k-1}G_{k-1}d_{k-1}}$$

où

$$\begin{cases} y_{k-1} = g(X_k) - g(X_{k-1}) \\ d_{k-1} = X_k - X_{k-1} \end{cases} \quad (8)$$

Pour la première itération on a $H = I_d$.

```
function XN=BFEx5(J,nablaJ,X0,N)
XN=X0;
d=-nablaJ(X0); //on multiplie par H=Id la première iteration
X1=X0+d;
XN=[XN,X1];
dk=X1-X0;
yk=nablaJ(X1)-nablaJ(X0);
X0=X1;
    for i=2:N
        B=(yk.*yk)/(yk.*dk);
        C=(H*dk*dk'*H)/(dk'*H*dk);
        H=H+B-C;
        d=-H*nablaJ(X0);
        X1=X0+d;
        XN=[XN,X1];
        dk=X1-X0;
        yk=nablaJ(X1)-nablaJ(X0);
    end
endfunction
```

Algorithmes d'optimisation locale avec contraintes

Groupe 5 :
Fabrice FEUTANG
Fatima-Ezzahrae SELAM
Hafsa EL HERICHI
Abraham HAKOBIAN

5 Décembre 2018

Table des matières

1 Méthode du Gradient Projeté ou Méthode de plus forte pente

Soit le problème d'optimisation à ressoudre :

$$\min f(x)$$

$x \in \mathbb{R}$
sous :

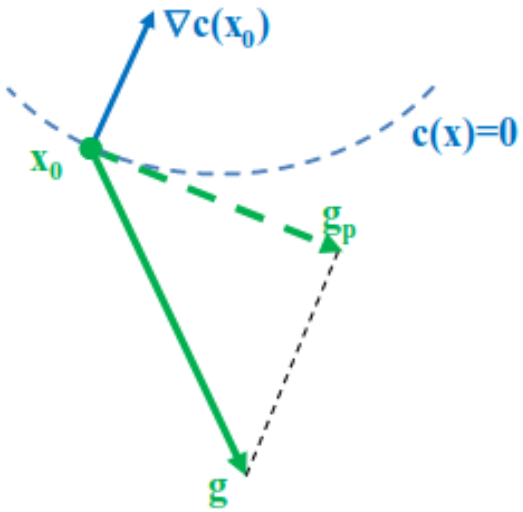
$$C_E(x) = 0$$
$$C_I(x) \leq 0$$

1.1 Definition

Le gradient projeté est la projection orthogonale de f sur l'hyperplan tangent.

1.2 Principe de la méthode

Le principe de la methode consiste à adapter l'algorithme sans contrainte du gradient dans le cas sans contraintes , en le projectant sur les direction admissibles :hyperplan tangent aux contrainte.



une phase de reajustement dans le domaine admissible est necessaire.
La méthode du gradient projeté équivaut à la méthode de plus forte pente appliquée dans l'espace nul des contraintes.

Exemple

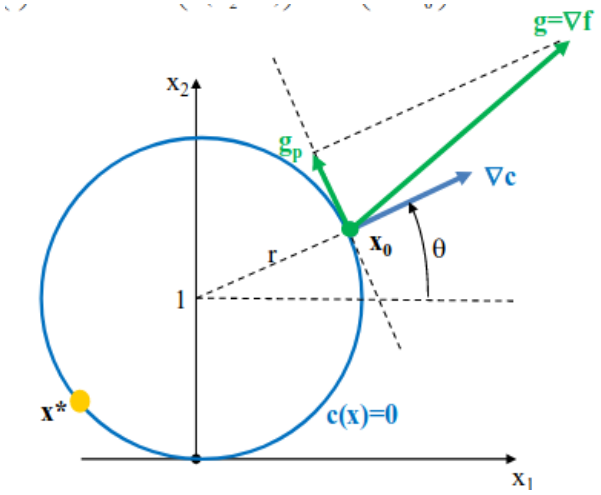
soit à minimiser la fonction :

$f(x) = x_1 + x_2$ sous $x_1^2 + (x_2 - 1)^2 - 1 = 0$

- **point admissible** $x_0 = (1, \theta_0)$.
  $\bigtriangledown f(x_0) = (1, 1)$
  $\bigtriangledown\ c(x_0) = (2x_1, 2(x_2 - 1)) = 2r_0(cos\theta_0, sin\theta_0)$
- **Gradient Projeté** x_0 $g_p = Pg$ avec $A = \bigtriangledown c(x_0)^T$,$g = \bigtriangledown f(x_0)$,
  $P = I - A^T(AA^T)^{-1}A$

$A = 2r_0(cos\theta_0 sin\theta_0)$ $g_p = (cos\theta_0, -sin\theta_0).(cos\theta_0, -sin\theta_0)^T$

- **Direction de descente**
  $d = \frac{g_p}{\|g_p\|}$
  $d = (-sin\theta_0, cos\theta_0)$ou $(sin\theta_0, -cos\theta_0)$



1.3 Algorithme du Gradient projété

Algorithme de gradient projeté

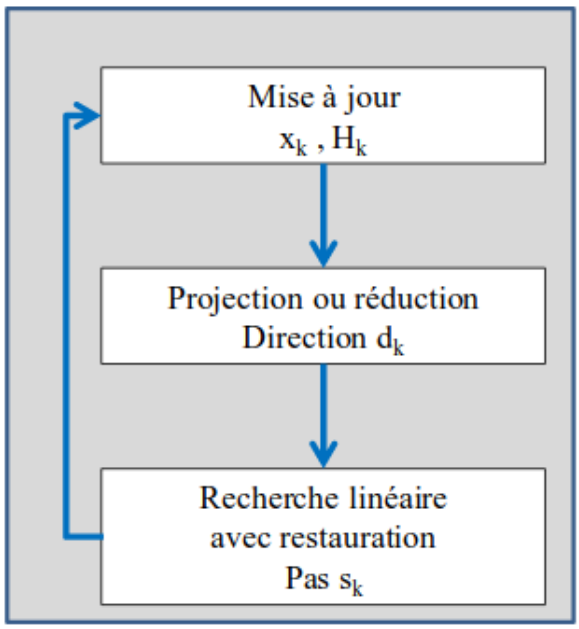
— construire la direction de descente au point courant
— effectuer une recherche linéaire avec restauration

Direction de descente

— Sélection des contraintes actives
— projection ou réduction dans l'hyperplan tangent
— Mise à jour du hessien(quasi-Newton)

Recherche linéaire

— Méthode de dichotomie sur le pas de déplacement
— Restauration avant évaluation du pas
— Règles d'acceptation (Armijo,...)



2 Méthodes de pénalisation

2.1 Introduction

La pénalisation est un concept simple qui permet de transformer un problème d'optimisation avec contraintes en un problème ou en une suite de problèmes d'optimisation sans contraintes.

D'un point de vue théorique, une approche par pénalisation peut être utilisée pour étudier un problème d'optimisation dont les contraintes sont diciles à prendre en compte, alors que le problème pénalisé a des propriétés mieux comprises ou plus simples à mettr en évidence.

D'un point de vue numérique, cette transformation en problèmes sans contrainte permet d'utiliser des algorithmes d'optimisation sans contrainte pour obtenir la solution de problèmes dont l'ensemble admissible peut avoir une structure complexe. Cette approche est très souvent utilisée : elle permet d'obtenir une solution de qualité susante rapidement sans avoir à entrer dans l'algorithmique sophistiquée de l'optimisation avec contraintes. Ce n'est cependant pas une technique universelle, car elle a ses propres inconvénients : non-diérentiabilité ou nécessité de minimiser une suite de fonctions de plus en plus mal conditionnées.

2.1.1 Problème à résoudre :

(P) : $\min f(x)$ sur $C_E(x) = 0\,;\, C_I(x) < 0$

2.1.2 Problème pénalisé :

Critère augmenté :
On construit une suite d'approximations qui ne serait pas forcément dans le domaine admissible, en changeant la fonction à minimiser (par pénalisation).
On considère le nouveau problème (P') :

$\min(f(x) + \rho * \alpha(x))$ où :
$\begin{cases} \rho > 0 : \text{ paramètre à fixer ou faire tendre vers l'infini.} \\ \alpha(x) >= 0 : \text{ pénalisation} \end{cases}$

2.2 Pénalisation en norme 2 (pénalisation quadratique)

$\alpha(x) = \frac{1}{2} \|C(x)\|_2^2$ (avec C(x) : Contraintes avtives du problème.)

- **Problème pénalisé :**

$$\min f(x) + \tfrac{1}{2} \|C(x)\|_2^2 \longrightarrow \text{solution } x_\rho \qquad (x \in \mathbb{R}^n)$$

• **Méthode de résolution :**

+ On résout une suite de problèmes pénalisés avec des valeurs croissantes de la pénalisation ρ.
+ Chaque problème k+1 est initialisé avec la solution précédente x_k.
+ Problème k avec pénalisation ρ_k : $\min f_{\rho_k}(x) \longrightarrow$ solution x_k
+ Il faut vérifier que la suite des solutions x_k converge vers la solution x^* du problème initial.

2.3 Pénalisation en norme 1 (pénalisation exacte)

$\alpha(x) = \|C(x)\|_1$ (avec C(x) : Contraintes avtives du problème .)

• **Problème pénalisé :**

$$\min f(x) + \|C(x)\|_1 \longrightarrow \text{solution } x_\rho \qquad (x \in \mathbb{R}^n)$$

• **Méthode de résolution :**

+ On résout une suite de problèmes pénalisés avec des valeurs croissantes de la pénalisation ρ.
+ Chaque problème k+1 est initialisé avec la solution précédente x_k.
+ Problème k avec pénalisation ρ_k : $\min f_{\rho_k}(x) \longrightarrow$ *solution* x_k
+ Il faut vérifier que la suite des solutions x_k converge vers la solution x^* du problème initial.

2.4 Mise en oeuvre (Méthodes avec critère augmenté)

2.4.1 Types de pénalisation

• **Pénalisation quadratique :** différentiable, mais nécessite une pénalisation forte pour approcher x^*.
• **Pénalisation exacte :** exacte, mais non différentiable.

2.4.2 Réglage de la pénalisation

• **Trop faible :** risque de divergence (pas de minimum du problème pénalisé).
• **Trop forte :** mauvais conditionnement, difficultés numériques.

2.4.3 Utilisation du critère augmenté

• Difficultés pratiques si l'on veut obtenir une bonne précision sur la solution x^*.
• Le critère augmenté peut servir de fonction mérite dans le cadre d'autres algorithmes pour évaluer la progression vers l'optimum.

2.4.4 Méthodes avec lagrangien augmenté

On cherche à se ramener à une suite de problèmes sans contraintes :
- En onservant un critère différentiable.
- En évitant le mauvais conditionnement dû à une pénalisation trop forte : utilisation des multiplicateurs de Lagrange pour réduire la pénalisation : méthode duale ou lagrangienne.

• **Principe de la méthode du Lagrangien augmenté**

La méthode du Lagrangien augmenté consiste à résoudre une suite de problèmes :
$\min L_{\rho_k}(x;\lambda_k) = L(x;\lambda_k) + \frac{1}{2}\rho_k \left\|C(x)\right\|_2^2 = f(x) + \lambda_k^T C(x) + \frac{1}{2}\rho_k \left\|C(x)\right\|_2^2$
Avec ρ_k=valeur de pénalisation du problème k
Et λ_k=estimation des multiplicateurs pour le problème k.
La solution x_k du problème $\min L_{\rho_k}(x;\lambda_k)$ converge vers la solution x^* du problème initial.
Elle vérifie également : $\nabla_x L_{\rho_k}(x_k;\lambda_k) = 0$

3 Méthode duale

3.1 Rappels

3.1.1 Lagrangien

Nous rappelons que le Lagrangien du problème est défini comme qui suit :
$\mathcal{L}(x, \lambda, \mu) = f(x) + \lambda C_E(x) + \mu C_I(x)$.

3.1.2 Point selle

Définition 1 *On appelle un point selle de $\mathcal{L}$ défini sur $\mathbb{R}^n \times \mathbb{R}^p \times \mathbb{R}^q_+$ tout triplet $(x^*, \lambda^*, \mu^*) \in \mathbb{R}^n \times \mathbb{R}^p \times \mathbb{R}^q_+$ véritifiant l'équation :*
$\forall (x, \lambda, \mu) \in \mathbb{R}^n \times \mathbb{R}^p \times \mathbb{R}^q_+ : \mathcal{L}(x^*, \lambda, \mu) \leq \mathcal{L}(x^*, \lambda^*, \mu^*) \leq \mathcal{L}(x, \lambda^*, \mu^*)$

3.2 Le problème dual

3.2.1 Problème primal et problème dual

Problème Primal (P) : minimiser $f(x)$ sous les contraintes d'égalité $C_E(x)$ ainsi que les contraintes d'inègalité $C_I(x)$, $x \in X \subseteq \mathbb{R}^n$.

X est une partie quelconque de $\mathbb{R}^n$. $\mathbb{R}^n$ peut être un ensemble discret (un ensemble de nombres entiers par exemple), ou encore un ensemble défini par des contraintes d'égalité ou d'inégalité autres que ceux définis plus haut.

À partir du problème (P), nous pouvons définissons un autre problème que nous appelons : problème dual (D).

(D) a pour but de maximiser $\omega(\lambda, \mu) = \inf\{f(x) + \lambda C_E + \mu C_I : x \in X\}$ sous la contrainte $\mu \geq 0$.

On rappelle que $\mathcal{L}(x, \lambda, \mu) = f(x) + \lambda C_E(x) + \mu C_I(x)$ est la fonction Lagrangienne.

Et on appelle $\omega(\lambda, \mu) = inf_{x \in X} \mathcal{L}(x, \lambda, \mu)$ la fonction duale.

Le problème dual (D) est sans contraintes mais la fonction ω est coûteuse à évaluer.

3.2.2 Théorème de dualité

Theorem 1 *Si (x^*, λ^*, μ^*) est un point-selle, alors x^* est une solution (optimale) de (P) et (λ^*, μ^*) est une solution (optimale) de (D).*

3.3 Maximisation duale

Le but de cette méthode est de résoudre le problème dual, c'est-à-dire de maximiser la fonction duale sous la contrainte que certaines des variables doivent être positives ou nulles.

La simplicité de ces contraintes fait que l'on peut adapter l'algorithme de plus forte pente utilisé dans le cas de l'optimisation sans contraintes.
La différence est que l'on utilise un sous-gradient à la place du gradient et que l'on interdit des déplacements qui rendraient négatives des variables astreintes à être positives ou nulles.

Pour ce faire, nous aurons besoin de $X_{\mathcal{L}}(x, \lambda, \mu)$ qui représente la valeur de x qui minimise $\mathcal{L}(x, \lambda, \mu)$ à λ et μ fixés.

Nous aurons également besoin de la condition d'ordre 1 qui dit que si $min_{x\in\mathbb{R}^n}$ en $x = x_{\mathcal{L}}(\lambda, \mu)$, alors $\nabla_x \mathcal{L}(x_L(\lambda, \mu), \lambda, \mu) = 0$.
Nous avons également besoin de connaître la fonction duale : $\omega(\lambda, \mu) = min_{x\in\mathbb{R}^n}\mathcal{L}(x, \lambda, \mu)$, son gradient ainsi que son Hessien.

Une fois que nous avons toutes ces informations, nous pouvons procéder à la maximisation duale.

3.3.1 Méthode de maximisation

L'évaluation de la fonction duale $\omega(\lambda, \mu) = \min_{x\in\mathbb{R}^n} \mathcal{L}(x, \lambda) = \mathcal{L}(x_{\mathcal{L}}(\lambda, \mu), \lambda, \mu)$ est très coûteuse donc nous cherchons à faire une minimisation en chaque valeur (λ, μ) pour obtenir $x_{\mathcal{L}}(\lambda, \mu)$ puisque l'on ne peut pas directement maximiser ω.

On réalise donc à chaque itération :
• une minimisation par rapport à x : $(x_k, \lambda_k, \mu_k) \longrightarrow (x_{k+1}, \lambda_k, \mu_k)$
• une maximisation par rapport à λ et μ : $(x_{k+1}, \lambda_k, \mu_k) \longrightarrow (x_{k+1}, \lambda_{k+1}, \mu_{k+1})$

3.3.2 Déplacement

Le déplacement sur x est construit en minimisant $\mathcal{L}(x, \lambda_k, \mu_k)$.
Nous obtenons ainsi une minimisation exacte ou approchée grâce à

$$\min_{x\in\mathbb{R}^n} \mathcal{L}(x, \lambda_k, \mu k) \longrightarrow x_{k+1} = x_{\mathcal{L}}(\lambda_k, \mu_k)$$

Le déplacement sur (λ, μ) est construit à partir du gradient et éventuellement du Hessien de ω.
Il existe plusieurs méthodes de déplacement sur (λ, μ) telles que : Uzawa, le Lagrangien augmenté ou encore la méthode de Newton.

3.4 Algorithme d'Uzawa

3.4.1 Idée de l'algorithme :

L'idée à travers cet algorithme est de considérer le Lagrangien $\mathcal{L}$ au lieu de la fonction f.

Deux raisons principales justifient ce choix :
• La fonction Lagrangienne englobe la fonction f et représente bien le problème.

• Nous avons également vu qu'une condition nécessaire du premier ordre pour que x^* soit un minimum de f avec contraintes est que x^* soit un point critique de $\mathcal{L}$.

Nous avons le résultat qui suit d'après la définition d'un point selle :

Theorem 2 *Supposons que f , C_E et C_I soient des fonctions de classe C^1 et que le triplet $(x, \lambda, \mu) \in \forall (x, \lambda, \mu) \in \mathbb{R}^n \times \mathbb{R}^p \times \mathbb{R}^q_+$ soit un point-selle de $\mathcal{L}$ sur $\forall (x, \lambda, \mu) \in \mathbb{R}^n \times \mathbb{R}^p \times \mathbb{R}^q_+$. Alors ce point-selle vérifie les conditions de Kuhn-Tucker.*

Dans le cas important convexe, nous avons une caractérisation des points-selles grâce aux conditions KKT.

Theorem 3 *Supposons que f, C_E et C_I soient C^1 et convexes. Alors le triplet $(x, \lambda, \mu) \in \mathbb{R}^n \times \mathbb{R}^p \times \mathbb{R}^q_+$ est point-selle de $\mathcal{L}$ sur $\mathbb{R}^n \times \mathbb{R}^p \times \mathbb{R}^q_+$ si et seulement si il vérifie les conditions de Kuhn-Tucker.*

Le théorème précédent indique que nous allons chercher un triplet $(x, \lambda, \mu) \in \mathbb{R}^n \times \mathbb{R}^p \times \mathbb{R}^q_+$ vérifiant les conditions de $Kuhn - Tucker$ de la manière suivante :

1. Pour (λ^*, μ^*) fixés dans $\mathbb{R}^n \times \mathbb{R}^q_+$, nous allons chercher le minimum sans contraintes de la fonction $x \mapsto \mathcal{L}(x, \lambda^*, \mu^*)$.
2. Pour x^* fixé dans $\mathbb{R}^n$, on cherche le maximum sur $\mathbb{R}^p \times \mathbb{R}^q_+$ de la fonction $(\lambda, \mu) \mapsto \mathcal{L}(x^*, \lambda, \mu)$

On fait donc ces deux calculs simultanément et on obtient ainsi l'algorithme d'Uzawa.

3.4.2 Algorithme théorique d'Uzawa

Algorithme Théorique :

Entrée : $x_0, \lambda_0, \mu_0)$;
Conditions : $\alpha_1 > 0, \alpha_2 > 0$;
$k \longleftarrow k + 1$
$\mathbf{x_{k+1}} = \mathbf{x_k} - \alpha_\mathbf{1} \nabla_\mathbf{x} \mathcal{L}(\mathbf{x_k}, \lambda_\mathbf{k}, \mu_\mathbf{k})$
$\lambda_{k+1} = \lambda_k + \alpha_2 C_E(x_{k+1})$
$\mu_{\mathbf{k+1}} = \mathbf{max}(\mathbf{0}, \mu_\mathbf{k} + \alpha_\mathbf{2} \mathbf{C_I}(\mathbf{x_{k+1}}))$

4 TP 2 Optimisation : optimisation locale avec contraintes

L'objectif de cette séance est d'utiliser le logiciel Scilab (ou Matlab, ou Python) afin de comparer différents algorithmes de recherche du minimum d'une fonction f : $\mathbb{R}^n \to \mathbb{R}$ sous la contrainte égalité $c_E(x) = 0$

1 Exercice préliminaire

On considère le problème de la canette : étant donné une canette cylindrique de rayon r et de hauteur h, quelle est la surface minimale de cette canette pour un volume de 33cl ? Représenter graphiquement sur une même figure dans un repère d'abscisse r et d'ordonnée h les lignes de niveau de la fonction à minimiser ainsi que la courbe d'équation V=33cl.Retrouver graphiquement la condition de KKT sur cet exemple.

2 Exercice Méthode Uzawa

1. On cherche à programmer tout d'abord la méthode d'Uzawa pour résoudre le problème général de minimisation d'une fonction J sous la contrainte
cE (x)= 0.
Ecrire une fonction Scilab ayant pour arguments J, ∇J,c_E,∇c_E, X_0, λ_0 (point initial) α_1 et α_2 (pas) et N et renvoyant la valeur de X_N et de λ_N après N itérations.

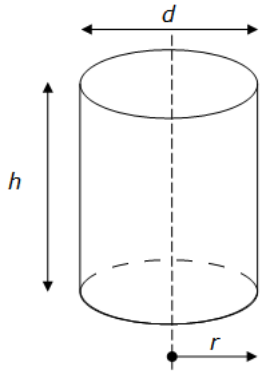
2.On suppose que J est la fonction de Rosenbrock et que $c_E(x_1, x_2) = 2x_1 + x_2 - 4$. Appliquer l'algorithme d'Uzawa sur cet exemple.

3. Que donne l'algorithme d'Uzawa sur le problème de la canette ?

Correction du TP

1 Préliminaire

La canette peut être représenter par un cylindre de rayon r et de hauteur h.



La fonction de la surface du cylindre est définie par $S(r,h) = 2\pi r^2 + 2\pi rh$.
Il faut qu'on minimise cette fonction sous la contrainte :
$C_E(r,h) = \pi r^2 h - V = 0$
On peut exprimer la hauteur en fonction du rayon à l'aide de la fonction de contrainte et ainsi obtenir une fonction de surface dépendant que d'une seule varible.On a $h = \frac{V}{\pi r^2}$
En injectant cette relation dans la fonction de surface S , on obtient :
$S(r) = 2\pi r^2 + \frac{2V}{r}$
Pour minimiser cette nouvelle fonction, on doit chercher la valeur de r pour laquelle sa dérivée est nulle.
$S'(r) = 4\pi r - \frac{2V}{r^2} = 0 \Leftrightarrow r_m =^3 \sqrt{\frac{V}{2\pi}}$
On a aussi $h_m = \frac{V}{\pi r_m^2}$
On a donc obtenu les valeurs de r et h pour laquelle la surface est minimale sous la contrainte V=330ml.

2 Exercice Méthode Uzawa

```
1
2
3  //Algorithme d'Uzawa
4  // on va utiliser la méthode de plus forte pente , direction de descente : l'opposé du gradient de
       la fonction J ( Lagragienne avec le multiplicateur de Lagrange et la contrainte)
5
6  function [XN,lambdaN]=Uzawa(J,gradJ,cE,gradcE,X0,lambda0,alpha1,alpha2,N)//on crée la fonction
       Uzawa en introduisant les variables d'entrées et de sorties
7    X=X0;lambda=lambda0; // On conserve les valeurs initiales {\'a}
8    XN=X;lambdaN=lambda;
9    for i=1:N
10     alphainit=alpha1; beta=0.1;tau=0.3; // on affecte la valeur de alpha1 , et on introduit les
           coefficients
11     alpha=alphainit;d=-(gradJ(X)+lambda*gradcE(X)); // on initialise la valeur de alpha et on
           introduit la direction de descente d
12
```

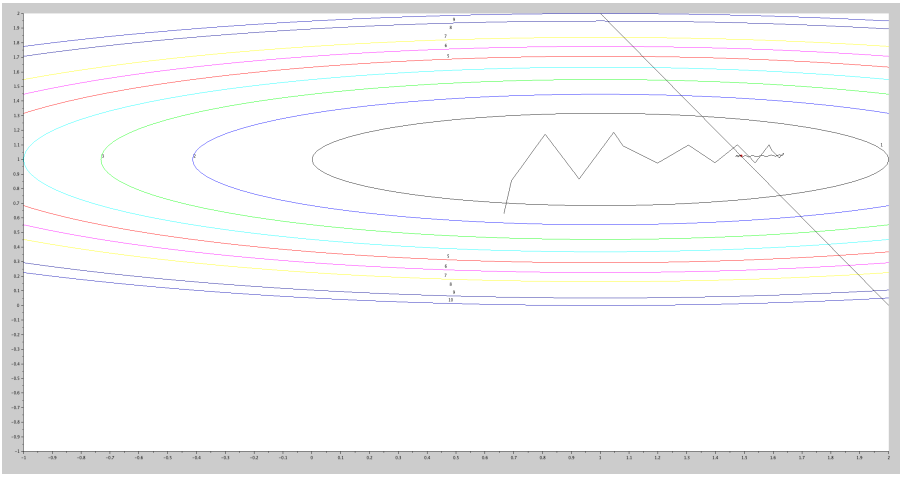
```
13      while (J(X+alpha*d)+lambda*cE(X+alpha*d)>J(X)+lambda*cE(X)-beta*alpha*d'*d) // La
            condition qu'on doit vérifier
14          alpha=alpha*tau // si la condition n'est pas vé rifier on affecte a alpha une nouvelle valeur
                en le multipliant par la valeur tau et on recommence tant que la condition n'est pas vé
                rifier
15      end
16
17      X=X+alpha*d; // apres avoir verifier la condition , le vecteur X prend une nouvelle valeur
18      lambda=lambda+alpha2*cE(X)// Pareil pour le multiplicateur de Lagrange
19      XN=[XN,X];lambdaN=[lambdaN,lambda]; // et on ajoute chaque nouvelle valeur de X et lambda
            dans les vecteurs XN et LambdaN initialises au debut
20    end
21  endfunction
22
23
24
25  //fonction quadratique
26  //on teste le programme sur la fonction de Rosenbrock sous une condition de contrainte
27
28  function y=Rose(x)
29      y=10*(x(2)-1)^2+(x(1)-1)^2 // la fonction de Rosenbrock
30  endfunction
31
32  function y=gradrose(x) // Le gradient de la fonction de Rosenbrock
33      y=[2*(x(1)-1);20*(x(2)-1)]
34  endfunction
35
36  function y=CE(x) // La fonction de contrainte CE
37      y=2*x(1)+x(2)-4;
38  endfunction
39  function y=gradCE(x) // le gradient de la fonction de contrainte
40      y=[2;1];
41  endfunction
42
43  // solution exacte
44  A=[2,0;0,20];b=[2;20];C=[2,1];d=[4];
45  M=[A,C';C,zeros(1,1)];z=[b;d];Xex=inv(M)*z;// calcule de la solution exacte
46
47  X0=rand(2,1);lambda0=-0.1;alpha1=0.1;alpha2=0.1;N=100;
48  [XN2,lambdaN2]=Uzawa(Rose,gradrose,CE,gradCE,X0,lambda0,alpha1,alpha2,N)
49
50  //tracé
51
52  figure(1)
53  Nt=100;
54  x=linspace(-1,2,Nt);
55  y=linspace(-1,2,Nt);
56  z=zeros(Nt,Nt);
57  for i=1:Nt
58      for j=1:Nt
59          z(i,j)=Rose([x(i),y(j)]) // on calcule les lignes de niveau
60  end
```

```
61
62  clf()
63  contour(x,y,z,[0:10]) // on trace les lignes de niveau
64  plot(Xex(1),Xex(2),'r*') // on place la solution exacte
65  plot2d(x,4*ones(x)-2*x,rect=[-1,-1,2,2])
66  plot2d(XN2(1,:),XN2(2,:))// on trace la convergence de la méthode
67
68
69  //Résolution avec la canette
70
71  // Solution exacte calculée dans le préliminaire
72  rmin=(165/%pi)^(1/3)
73  hmin=330/(%pi*rmin^2)
74
75  // on va introduire les fonctions de la canette
76
77  function s=surfc(X) // fonction qui permet de calculer la surface de canette
78      r=X(1);h=X(2); // un vecteur contenant le rayon et la hauteur de la canette
79      s=2*%pi*r^2+2*%pi*r*h
80  endfunction
81
82  function s=gradsurfc(X) // le gradient de la surface de la canette
83      r=X(1);h=X(2);
84      s=[4*%pi*r+2*%pi*h;2*%pi*r]
85  endfunction
86
87  function v=cE(X) // la fonction de la contrainte de la canette cE(x)=V-330=0)
88      r=X(1);h=X(2);
89      v=%pi*r^2*h-330
90  endfunction
91
92  function v=gradcE(X) // le gradient de la contrainte introduite précédement
93      r=X(1);h=X(2);
94      v=[2*%pi*r*h;%pi*r^2]
95  endfunction
96
97  X0=[3.7;7.5];lambda0=0.1;alpha1=0.01;alpha2=0.1;N=10;
98  [XN1,lambdaN1]=Uzawa(surfc,gradsurfc,cE,gradcE,X0,lambda0,alpha1,alpha2,N)
99  figure(2)
100 r=linspace(2,4,100)
101 h=linspace(6,9,100)
102 z=zeros(100,100);hvol=zeros(100,1)
103 for i=1:100
104     hvol(i)=330/(%pi*r(i)^2);
105     for j=1:100
106       z(i,j)=surfcan([r(i),h(j)]) // on calcule les lignes de niveau
107     end
108 end
109  clf()
110 contour(r,h,z,20) // on trace les lignes de niveau
111 plot2d([rmin,rmin],[hmin,hmin],-1) // on place la solution exacte
112 plot2d(r,hvol,1,rect =[2,6,4,9])
```

```
113 plot2d(XN1(1,:),XN1(2,:),rect=[2,6,4,9]) // on trace la convergence de la methode d'Uzawa sur le
      probleme de la canette
```

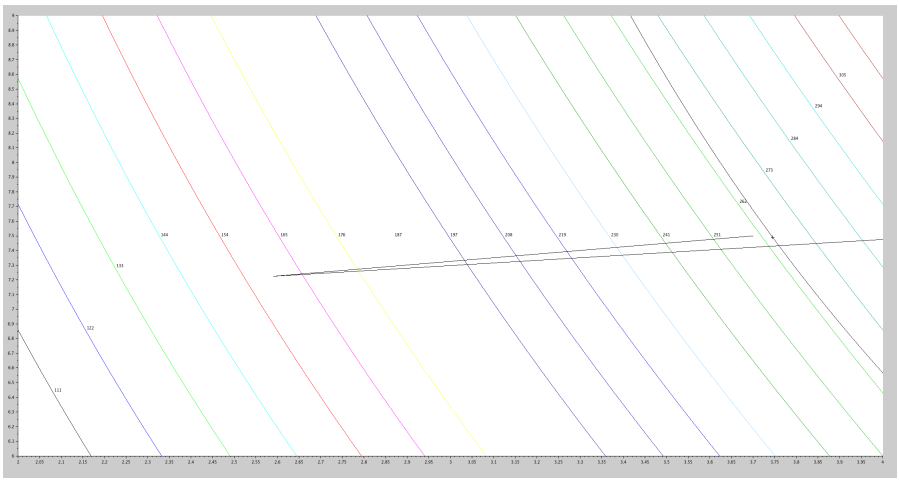
Résultats du programme :

La fonction de Rosenbrock



On peut constater que la méthode d'Uzawa converge après oscillations vers la solution exacte.Plus le pas est petit et plus la convergence sera lente.Les oscillations importantes sont dues à des pas élevés.

Le problème de la canette :



L'algorithme d'Uzawa diverge pour le problème de la canette.