Techniques Algorithmiques pour le Calcul Scientifique (LI364)

Cours n°3 : Introduction à l'optimisation (2/2)

Stef Graillat

Université Pierre et Marie Curie (Paris 6)

Résumé du cours précédent

- Généralités sur les problèmes d'optimisation
  - Notion de point critique
  - Condition nécessaire et suffisante d'optimalité
  - Multiplicateurs de Lagrange
- Optimisation en dimension 1
  - Méthode de la section dorée
  - Méthode de Newton

Plan du cours : optimisation en dimension $n \geq 2$

- Méthode de recherche directe
- Méthode de gradient
- Méthode de Newton
- Méthode de Quasi-Newton
- Méthode du gradient conjugué (non linéaire)

Bibliographie

- **Scientific Computing, An Introductory Survey, Michael .T. Heath, McGraw-Hill, 2002** (ce cours est basé sur le livre et les transparents associés)
- Mathématiques L3 - Mathématiques appliquées, Jacques-Arthur Weil, Alain Yger, Pearson Education, 2009
- Scientific Computing with Case Studies, Dianne P. O'Leary, SIAM, 2009
- Linear and Nonlinear Optimization, Igor Griva, Stephen G. Nash et Ariela Sofer, SIAM, 2009
- Practical Methods of Optimization, Roger Fletcher, Wiley, 1987
- A Mathematical View of Interior-Point Methods in Convex Optimization, James Renegar, SIAM, 2001
- Numerical Optimization, Jorge Nocedal et Stephen J. Wright, Springer, 2006.

Optimisation en dimension $n \geq 2$

Méthode par recherche de motif

Principe :

- On suppose que l'on a une approximation x de la solution et un ensemble d'au moins $n+1$ directions v_i, $i = 1, \ldots, N$ formant une base positive de $\mathbb{R}^n$: chaque vecteur de $\mathbb{R}^n$ peut s'exprimer comme une combinaison linéaire à coefficients positifs des vecteurs de la base
- À chaque étape, on fait une recherche linéaire dans chaque direction pour minimiser $f(x + \alpha_i v_i)$ et on remplace x par le point minimisant la fonction
- Il s'agit d'un algorithme simple et fonctionne bien en pratique
- C'est un algorithme facilement parallèlisable et qui ne nécessite aucun calcul de dérivée

Méthode de gradient

- Soit $f : \mathbb{R}^n \to \mathbb{R}$ une fonction de n variables à valeur réelle
- En chaque point x où le gradient est non nul, $-\nabla f(x)$ correspond *localement* à la direction de descente de plus grande pente : f décroît le plus rapidement dans cette direction que dans une autre !
- Méthode de gradient : on part d'un point initial x_0 et on calcul les itérés successifs par

$$x_{k+1} = x_k - \alpha_k \nabla f(x_k)$$

où α_k est un paramètre de "recherche linéaire" qui détermine la distance à parcourir dans la direction $\nabla f(x_k)$

Méthode de gradient (suite)

- Étant donné une direction de descente (par exemple l'opposé du gradient), on cherche α_k comme solution du problème de minimisation

$$\min_{\alpha_k \geq 0} f(x_k - \alpha_k \nabla f(x_k))$$

que l'on peut résoudre avec un algorithme d'optimisation en dimension 1 (voir le cours précédent)
- La méthode de gradient est fiable : tant que le gradient est non nul, on fait diminuer $f(x)$
- Mais la méthode ne se souvient pas des directions précédentes : on fait souvent une progression en zigzag !
- La convergence est linéaire

Exemple de la méthode de gradient

- Méthode de gradient pour minimiser

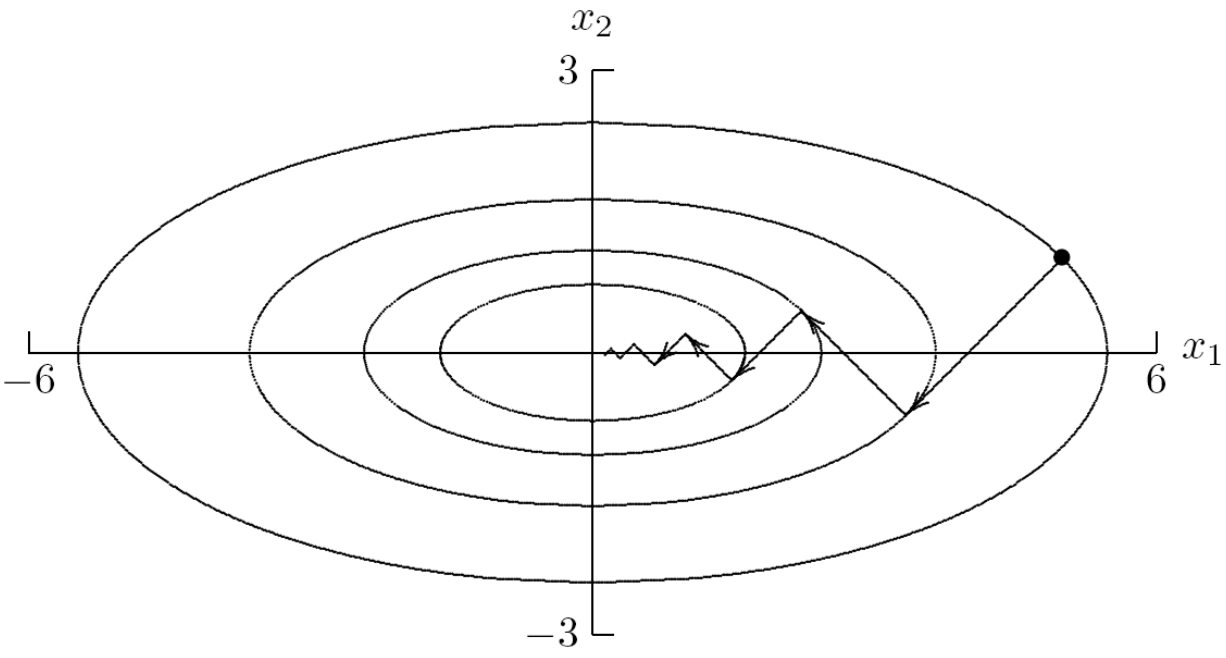
$$f(x) = 0.5x_1^2 + 2.5x_2^2$$

- Le gradient est donné par $\nabla f(x) = \begin{pmatrix} x_1 \\ 5x_2 \end{pmatrix}$
- En prenant $x_0 = \begin{pmatrix} 5 \\ 1 \end{pmatrix}$, on a $\nabla f(x_0) = \begin{pmatrix} 5 \\ 5 \end{pmatrix}$
- On fait ensuite une recherche linéaire dans la direction opposée au gradient

$$\min_{\alpha_0} f(x_0 - \alpha_0 \nabla f(x_0))$$

La solution est donnée par $\alpha_0 = 1/3$ et donc l'itéré suivant est $x_1 = \begin{pmatrix} 3.333 \\ -0.667 \end{pmatrix}$

Exemple de la méthode de gradient (suite)



Méthode par différence finie

Principe :

- On chercher à approximer le gradient par différence finie

$$(\nabla f(x))_i \approx \frac{f(x + te_i) - f(x)}{t}$$

  pour t petit et e_i le i-ième vecteur unité
- Peu de précision sur le gradient
- Mieux vaut utiliser des méthodes de différentiation automatique : méthode permettant d'évaluer numériquement les dérivées d'une fonction spécifiée par un programme informatique

Méthode de Newton

- On sait qu'un minimum x^* vérifie $\nabla f(x^*) = 0$. On peut donc chercher à résoudre l'équation $\nabla f(x) = 0$ par la méthode de Newton.
- On peut aussi approcher f par

$$f(x + h) \approx f(x) + \nabla f(x)h + \frac{1}{2}h^T H_f(x)h$$

  et minimiser l'approximation quadratique en fonction de h
- Dans les 2 cas, on obtient l'itération

$$x_{k+1} = x_k - H_f^{-1}(x_k)\nabla f(x_k)$$

Méthode de Newton (suite)

- On ne calcule pas explicitement l'inverse de la hessienne. On résout plutôt le système linéaire

$$H_f(x_k)s_k = -\nabla f(x_k)$$

et on calcule

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

- La convergence de la méthode de Newton est quadratique à condition de commencer l'itération assez près du résultat.

Exemple de méthode de Newton

- On veut minimiser

$$f(x) = 0.5x_1^2 + 2.5x_2^2$$

- Le gradient et la hessienne sont donnés par

$$\nabla f(x) = \begin{pmatrix} x_1 \\ 5x_2 \end{pmatrix} \quad \text{et} \quad H_f(x) = \begin{pmatrix} 1 & 0 \\ 0 & 5 \end{pmatrix}$$

- En partant de $x_0 = \begin{pmatrix} 5 \\ 1 \end{pmatrix}$, on obtient $\nabla f(x_0) = \begin{pmatrix} 5 \\ 5 \end{pmatrix}$
- Le système linéaire à résoudre devient

$$\begin{pmatrix} 1 & 0 \\ 0 & 5 \end{pmatrix} s_0 = \begin{pmatrix} -5 \\ -5 \end{pmatrix}$$

et donc $x_1 = x_0 + s_0 = \begin{pmatrix} 5 \\ 1 \end{pmatrix} + \begin{pmatrix} -5 \\ -1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$ qui est la solution exacte du problème (ce qui est normal pour une fonction quadratique)

Méthode de Newton (suite)

- Il n'est pas nécessaire de faire une "recherche linéaire" car la méthode de Newton donne la direction et la longueur : on ne cherche pas le paramètre α_k !
- Si le premier itéré est loin de la solution, on peut quand même faire une étape de "recherche linéaire" pour rendre la méthode plus robuste
- Une fois arrivé près de la solution, $\alpha_k = 1$ est suffisant pour les itérations restantes

Méthode de Newton (suite)

- Si la fonction f est de classe C^2, alors la hessienne H_f est symétrique et même défini positif près du minimum
- Pour résoudre le système linéaire, on peut donc utiliser une décomposition de Choleski $H_f = LL^T$ avec L triangulaire inférieur (deux fois moins coûteux qu'une décomposition LU)
- Loin du minimum, il se peut que $H_f(x_k)$ ne soit pas défini positif et donc que s_k ne soit pas une direction de descente, i.e.

$$\nabla f(x_k)^T s_k < 0$$

- Dans ce cas, on peut prendre comme direction de descente l'opposé du gradient et faire une recherche linéaire

Méthode de région de confiance

- On approche f par

$$f(x+h) \approx q(h) := f(x) + \nabla f(x)h + \frac{1}{2}h^T H_f(x)h$$

- On cherche à minimiser $q(h)$ mais on restreint la recherche à $\|h\| < r$ avec r assez petit
- On choisit $x_{\text{nouveau}} = x + h^*$ avec h^* solution de

$$\min_{\|h\| \leq r} q(h)$$

- Cela permet de ne rechercher une solution que dans un domaine où l'approximation quadratique est suffisamment précise

Méthode de Quasi-Newton

- Le calcul de la hessienne est très coûteux !
- Peut-on s'en passer, i.e. résoudre

$$\min_x f(x)$$

en ayant $\nabla f(x)$ mais pas $H_f(x)$

Deux exemples possibles :

- Estimer $H_f(x)$ par différence finie : méthode de Newton discrète
- Approcher $H_f(x)$ en utilisant des méthodes de Quasi-Newton

Méthode de Quasi-Newton (suite)

- Méthode de Newton : $x_{k+1} = x_k + s_k$ avec $s_k = H_f(x_k)^{-1}\nabla f(x_k)$
- Méthode de Quasi-Newton : utiliser $B_k \approx H_f(x_k)$ à l'aide d'informations déjà disponibles
- À l'étape k, on connaît $\nabla f(x_k)$ et on calcule $\nabla f(x_{k+1})$ avec $x_{k+1} = x_k + s_k$
- $H_f(x_k)$ satisfait

$$H_f(x_k)s_k = \lim_{t\to 0} \frac{\nabla f(x_k + ts_k) - \nabla f(x_k)}{t}$$

Notons $g_k = \nabla f(x_k)$. Si f est quadratique alors

$$H_f(x_k)s_k = g_{k+1} - g_k$$

- On va donc cherche une approximation B_k vérifiant l'équation

$$B_{k+1}s_k = g_{k+1} - g_k$$

dite équation de la sécante

Méthode de Quasi-Newton (suite)

- On connaît le comportement de B_{k+1} dans la direction $g_{k+1} - g_k$ mais on a aucune information pour les autres directions. On peut demander à ce que

$$B_{k+1}v = B_k v \text{ si } v^T s_k = 0$$

- Il y a une unique matrice vérifiant ces conditions

$$B_{k+1} = B_k - (B_k s_y - y_k)\frac{s_k^T}{s_k^T s_k}$$

où $s_k = x_{k+1} - x_k$ et $y_k = g_{k+1} - g_k$

- B_{k+1} est formé à partir de B_k en ajoutant une matrice de rang 1
- Il s'agit de la méthode de Broyden

Algorithme BFGS (Broyden-Fletcher-Goldfarb-Shanno)

- B_{k+1} n'est pas nécessairement symétrique même si B_k l'est
- On peut aussi chercher B_{k+1} sous la forme

$$\min_{B_{k+1}} \|B_{k+1} - B_k\|$$

  vérifiant B_{k+1} symétrique et $B_{k+1}s_k = y_k$
- Le choix de B_{k+1} dépend donc de la norme choisie
- Pour un choix de norme donnée (non décrit ici), on obtient la formule BFGS

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k S_k} + \frac{y_k y_k^T}{y_k^T s_k}$$

Algorithme BFGS (suite)

$x_0 =$ `approximation initiale`
$B_0 =$ `approximation de la hessienne`
pour $k = 0, 1, 2, \ldots$
    `resoudre` $B_k s_k = -\nabla f(x_k)$
    $x_{k+1} = x_k + s_k$
    $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$
    $B_{k+1} = B_k - (B_k s_k s_k^T B_k)/(s_k^T B_k S_k) + (y_k y_k^T)/(y_k^T s_k)$
fin

Algorithme BFGS (suite)

- En pratique, c'est souvent la factorisation de B_k qui est mise à jour et non B_k elle-même, ce qui permet de résoudre le système linéaire pour s_k en $\mathcal{O}(n^2)$ plutôt que $\mathcal{O}(n^3)$
- Contrairement à méthode de Newton, on ne calcule pas la hessienne
- On peut commencer avec $B_0 = I$: on prend alors une direction de descente opposée au gradient
- L'algorithme BFGS a normalement un taux de convergence super-linéaire même si B_k ne converge pas vers la vraie hessienne

Méthode du gradient conjugué non linéaire

- Il s'agit d'une méthode qui ne nécessite pas le calcul de la hessienne et ne nécessite pas non plus le stockage d'une matrice approximant la hessienne
- L'algorithme génère une suite de directions conjuguées accumulant de manière implicite de l'information sur la hessienne
- Pour certaines fonctions quadratiques, l'algorithme du gradient conjugué converge après au plus n itérations, où n est la dimension du problème

Méthode du gradient conjugué non linéaire (suite)

$x_0 =$ `approximation initiale`
$g_0 = \nabla f(x_0)$
$s_0 = -g_0$
pour $k = 0, 1, 2, \ldots$
    `choisir` α_k `minimisant` $f(x_k + \alpha_k s_k)$
    $x_{k+1} = x_k + \alpha_k s_k$
    $g_{k+1} = \nabla f(x_{k+1})$
    $\beta_{k+1} = (g_{k+1}^T g_{k+1})/(g_k^T g_k)$
    $s_{k+1} = -g_{k+1} + \beta_{k+1} s_k$
fin

$\rightarrow$ l'algorithme sera étudié en détail dans le cours sur les méthode itérative de résolution de systèmes linéaires

Méthode de Newton tronquée

- On peut réduire la quantité de calcul d'une méthode de type Newton en résolvant le système linéaire par une méthode itérative
- Un petit nombre d'itération permet alors d'obtenir en général une solution
- La méthode itérative utilisée pour résoudre le système est souvent la méthode du gradient conjugué
- Ces algorithmes nécessitent seulement des produits matrice-vecteur. Il est alors possible d'éviter la construction de la hessienne en utilisant une méthode de différence finie sur le gradient

Moindres carrés non linéaire

- Étant donné (t_i, y_i), on cherche un vecteur x de paramètres qui s'approche "au mieux" au sens des moindres carrés du modèle $f(t, x)$ où f est une fonction non linéaire de x
- On définit le résidu par

$$r_i(x) = y_i - f(t_i, x) \text{ pour } i = 1, \ldots, m$$

  On veut donc minimiser $\phi(x) = \frac{1}{2} r(x)^T r(x)$
- Le gradient est $\nabla\phi(x) = J(x)^T r(x)$ et la hessienne est

$$H_\phi(x) = J(x)^T J(x) + \sum_{i=1}^{m} r_i(x) H_i(x)$$

  où $J(x)$ est la matrice jacobienne de $r(x)$ et $H_i(x)$ est la hessienne de $r_i(x)$

Méthode de Gauss-Newton

- Le système linéaire à résoudre dans la méthode de Newton est

$$(J(x_k)^T J(x_k) + \sum_{i=1}^{m} r_i(x_k) H_i(x_k)) s_k = -J(x_k)^T r(x_k)$$

- Calculer les m hessiennes H_i est coûteux
- De plus, les H_i sont multipliées par les résidus r_i qui sont en général petit si l'on s'approche de la solution et si le modèle correspond bien aux données
- Par conséquent, on peut négliger les termes du second ordre et on résout le système

$$J(x_k)^T J(x_k) s_k = -J(x_k)^T r(x_k)$$

- Il s'agit en fait de l'équation normale associée au problème de moindres carrés linéaires

$$J(x_k) s_k \approx -r(x_k)$$

Méthode de Gauss-Newton (suite)

- On remplace un problème de moindres carrés non linéaires par une suite de problèmes de moindres carrés linéaires
- Si le résidu de la solution est grand alors les termes de second ordre sont importants et la méthode peut ne pas converger
- Si le résidu est grand, il vaut alors mieux utiliser une méthode de minimisation non linéaire qui prenne en compte la hessienne

Méthode de Levenberg-Marquardt

- Il se peut que la matrice $J(x_k)^T J(x_k)$ soit singulière (mais elle est symétrique)
- La méthode de Levenberg-Marquardt consiste à perturber un peu la matrice pour la rendre définie positive
- On résout à chaque étape le système

$$(J(x_k)^T J(x_k) + \mu_k I)s_k = -J(x_k)^T r(x_k)$$

- Le paramètre μ_k est choisi de façon à ce que $(J(x_k)^T J(x_k) + \mu_k I)$ soit symétrique défini positive

Optimisation sous contrainte

- On cherche à résoudre le problème

$$\min_x f(x) \text{ sous } g(x) = 0$$

avec $f : \mathbb{R}^n \to \mathbb{R}$, $g : \mathbb{R}^n \to \mathbb{R}^m$ avec $m \leq n$

- On cherche un point critique du lagrangien $\mathcal{L}(x, \lambda) = f(x) + \lambda^T g(x)$
- En appliquant la méthode de Newton au système non linéaire

$$\nabla \mathcal{L}(x, \lambda) = \begin{pmatrix} \nabla f(x) + J_g(x)^T \lambda \\ g(x) \end{pmatrix} = 0$$

on obtient le système linéaire

$$\begin{pmatrix} B(x, \lambda) & J_g(x)^T \\ J_g(x) & 0 \end{pmatrix} \begin{pmatrix} s \\ \lambda \end{pmatrix} = - \begin{pmatrix} \nabla f(x) + J_g(x)^T \lambda \\ g(x) \end{pmatrix}$$

Cette technique s'appelle la programmation quadratique séquentielle

Programmation linéaire

- Il s'agit d'un problème de minimiser d'une fonction linéaire sous contraintes linéaires
- On peut l'écrire sous la forme

$$\min f(x) = c^T x \quad \text{sous} \quad Ax = b \text{ et } x \geq 0$$

avec $m < b$, $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$ et $c, x \in \mathbb{R}^n$

- L'ensemble des contraintes forme un polyèdres de $\mathbb{R}^n$ et le minimum est atteint à un sommet de ce polyèdres
- La méthode du simplexe consiste à aller de sommets en sommets jusqu'à ce que l'on trouve le minimum

Programmation linéaire (suite)

- La méthode du simplexe est fiable et efficace en général mais est exponentielle en la taille du problème de les pires cas
- Des méthodes de point intérieur développées récemment permettent de résoudre des problèmes de programmation linéaire avec une complexité polynomiale dans le pire cas
- Les méthodes de point intérieur naviguent à l'intérieur des régions admissibles en ne se restreignant pas seulement aux sommets
- Bien que les méthodes de point intérieur soient de meilleurs complexités, la méthode du simplexe reste la méthode la plus utilisée en pratique

Exemple

- Considérons le problème

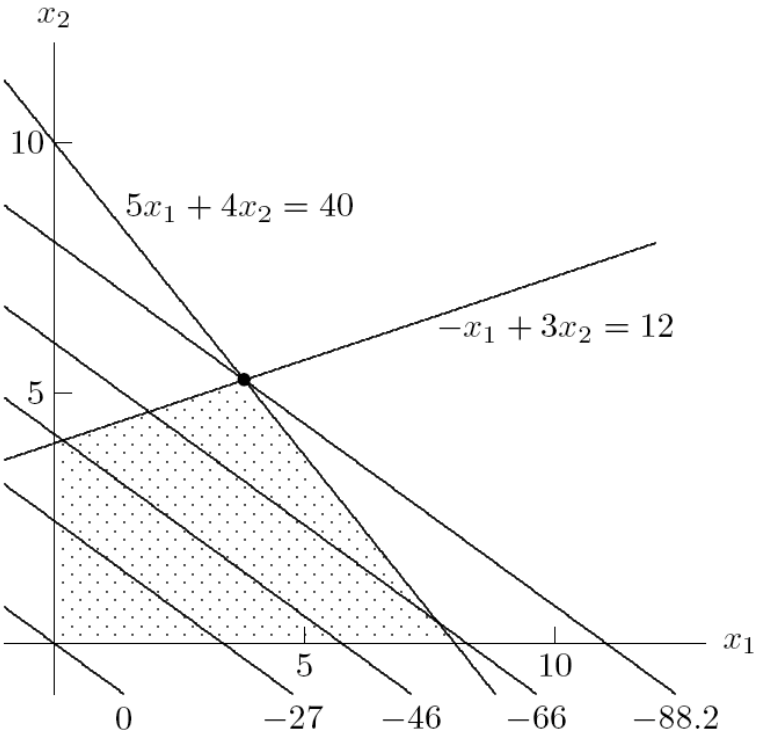
$$\min_x -8x_1 - 11x_2$$

sous les conditions

$$5x_2 + 4x_2 \leq 40, \quad -x_1 + 3x_2 \leq 12, \quad x_1 \geq 0, \quad x_2 \geq 0$$

- Le minimum est obtenu en $x_1 = 3,79$ et $x_2 = 5,26$ où la fonction objectif vaut -88.2

Exemple (suite)



Conclusion

- Méthode de Newton
- Méthodes alternatives ne nécessitant pas le calcul de la hessienne
  - Méthode de Quasi-Newton
  - Méthode de Newton par différence finie
- Méthodes alternatives ne nécessitant pas le calcul de la hessienne ni le stockage de matrice
  - Méthode de gradient
  - Méthode de gradient conjugué non linéaire
- Méthodes alternatives ne nécessitant pas le calcul de dérivées
  - Méthode de différence finie
  - Méthode de Nelder-Meade
  - Méthode de recherche par motif