TP ENSEEIHT
Méthodes numériques pour les problèmes d'optimisation

E. Bergou, Y. Diouane, S. Gratton, C. Royer, D. Ruiz et E. Simon

9 décembre 2013

1 Optimisation sans contraintes

Dans cette partie, nous nous intéressons aux algorithmes pour la résolution de problèmes d'optimisation sans contraintes, et en particulier aux algorithmes de recherche linéaire. L'objectif de ce TP sera de réaliser un code de minimisation locale, et d'évaluer ses performances sur les fonctions de tests suivantes :

- $f_1(x, y) = 2(x + y - 2)^2 + (x - y)^2$.
- $f_2(x, y) = 100(y - x^2)^2 + (1 - x)^2$.

1.1 Algorithmes de descente

Un algorithme de descente est déterminé par les stratégies de choix des directions de descente successives, puis par le pas qui sera effectué dans la direction choisie. Dans la première partie nous allons nous concentrer sur le choix de la direction de descente, et dans la deuxième partie nous allons voir des méthodes pour le calcul du pas, qui garantissent la convergence globale.

Directions de descente

L'idée est de remplacer la fonction f par un modèle local plus simple, dont la minimisation nous donnera une direction de descente.

1.1.1 Algorithmes de gradient à pas fixe ou pas optimal

Soit $x_k \in \mathbb{R}^n$ l'itéré courant, on remplace f au voisinage de x_k par son développement de Taylor au premier ordre :

$$f(x_k + d) \sim f(x_k) + \nabla f(x_k)^T d.$$

Une direction de descente possible est la direction de plus profonde descente définie par :

$$d_k = -\nabla f(x_k).$$

Le choix de la direction de plus profonde descente définit une famille d'algorithmes appelés algorithmes de descente de gradient, dont le schéma est le suivant :

Algorithm 1.1 ALGORITHME DE DESCENTE

Données : f , x_0 première approximation de la solution cherchée, $\epsilon > 0$ précision demandée.

Sortie : une approximation de la solution du problème : $\min_{x \in \mathbb{R}^n} f(x)$

1.Tant que le test de convergence n'est pas satisfait :

a. Direction de descente : $d_k = -\nabla f(x_k)$.

b. Recherche linéaire : Choisir un pas s_k à faire dans cette direction, tel que : $f(x_{k+1}) \leq f(x_k)$.

c. Mise à jour : $x_{k+1} = x_k + s_k d_k$, $k = k + 1$,

2. Retourner x_k.

1.1.2 Méthode de Newton locale

Pour construire les méthodes de gradient, nous avons remplacé f par son approximation linéaire au voisinage de l'itéré courant. Ces méthodes ne sont pas très performantes, parce qu'elles ne prennent pas en compte la courbure de la fonction (convexité, concavité ...) qui est une information au second ordre.

Principe

Supposons maintenant que f est de classe $\mathcal{C}^2$ et remplaçons f au voisinage de l'itéré courant x_k par son développement de Taylor au second ordre :

$$f(y) \sim q(y) = f(x_k) + \nabla f(x_k)^T (y - x_k) + \frac{1}{2}(y - x_k)^T \nabla^2 f(x_k)\,(y - x_k),$$

On choisit alors comme point x_{k+1} le minimum de la quadratique q lorsqu'il existe et est unique, ce qui n'est le cas que si $\nabla^2 f(x_k)$ est définie positive. Or le minimum de q est réalisé par x_{k+1} solution de : $\nabla q(x_{k+1}) = 0$, soit :

$$\nabla f(x_k) + \nabla^2 f(x_k)(x_{k+1} - x_k) = 0,$$

ou encore, en supposant que $\nabla^2 f(x_k)$ est définie positive :

$$x_{k+1} = x_k - \nabla^2 f(x_k)^{-1} \nabla f(x_k).$$

La méthode ne doit cependant jamais être appliquée en utilisant une inversion de la matrice Hessienne (qui peut être de très grande taille et mal conditionnée) mais plutôt en utilisant :

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

où d_k est l'unique solution du système linéaire :

$$\nabla^2 f(x_k) d_k = -\nabla f(x_k)$$

d_k étant appelée direction de Newton.

Cette méthode est bien définie si à chaque itération, la matrice hessienne $\nabla^2 f(x_k)$ est définie positive : ceci est vrai en particulier au voisinage de la solution x^* cherchée si on suppose que $\nabla^2 f(x^*)$ est définie positive (par continuité de $\nabla^2 f$).

Algorithme

Algorithm 1.2 MÉTHODE DE NEWTON LOCALE

Données : f , x_0 première approximation de la solution cherchée, $\epsilon > 0$ précision demandée.

Sortie : une approximation de la solution du problème : $\min_{x \in \mathbb{R}^n} f(x)$

1.Tant que le test de convergence est non satisfait :

a. Calculer d_k solution du système : $\nabla^2 f(x_k) dk = -\nabla f(x_k)$,

b. Choisir un pas s_k, (e.g $s_k = 1$),

c. Mise à jour : $x_{k+1} = x_k + s_k d_k$, $k = k + 1$,

2. Retourner x_k.

Travail à réaliser

Résolution mathématique

Pour chacun des problèmes tests,

- Donner les points critiques des fonctions proposées.
- Les fonctions f_i admettent-elles des minima sur $\mathbb{R}^2$?

Programmation en Matlab

Résoudre les problèmes tests de minimisation :

$$\min_{(x,y)\in\mathbb{R}^2} f_i(x)\ i \in 1,2$$

par l'algorithme de Newton (pour le pas, prendre $s_k = 1$ à chaque itération).

- Préciser le ou les critère(s) d'arrêt utilisé(s).
- Commenter les résultats obtenus avec différents points de départ x_0.

1.1.3 Méthodes de Quasi-Newton

Principe

Pour éviter le calcul de la matrice hessienne, les méthodes de Quasi-Newton utilisent une approximation de cette matrice comme suit :

Pour une fonction quadratique, il est aisé de démontrer que $\nabla f(x_1) - \nabla f(x_2) = \nabla^2 f(x_1)(x_1 - x_2)$. Cela indique que la connaissance de deux vecteurs distincts x_1 et x_2 et de la différence de gradient associée permet d'obtenir, au voisinage de la solution, une approximation sur $\nabla^2 f(x)$. Plus généralement, on suppose connus $s = x_1 - x_2$ et $y = \nabla f(x_1) - \nabla f(x_2)$, ainsi qu'une approximation courante B de la Hessienne. On cherche une nouvelle approximation $\hat{\mathrm{B}}$, telle que $\hat{\mathrm{B}}$ soit symétrique et que $\hat{\mathrm{B}}\ s = y$.
Cela ne suffit pas pour définir de manière unique $\hat{\mathrm{B}}$, et on recherche des $\hat{\mathrm{B}}$ de norme minimale (pour certaines normes) pour forcer l'unicité. Un exemple d'algorithme de Quasi-Newton est l'Algorithme BFGS qui approxime directement l'inverse de la matrice hessienne.

Algorithme(BFGS)

Algorithm 1.3 BFGS

Données : f , x_0 première approximation de la solution cherchée, $\epsilon > 0$ précision demandée, approximation initiale de l'inverse de la matrice hessienne en x_0 B_0

Sortie : une approximation de la solution du problème : $\min_{x\in\mathbb{R}^n} f(x)$

1.Tant que le test de convergence est non satisfait :

a. Calculer d_k : $d_k = -B_k \nabla f(x_k)$.

b. Effectuer une recherche linéaire pour trouver le pas optimal s_k dans la direction trouvée.

c. Mise à jour de x_{k+1} : $x_k = x_k + s_k d_k$.

d. Mise à jour de y_k : $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$.

e. Mise à jour de B_k : $B_{k+1} = B_k + \frac{(s_k d_k - B_k y_k) d_k^T + d_k (s_k d_k - B_k y_k)^T}{d_k^T y_k} - \frac{(s_k d_k - B_k y_k)^T y_k}{(d_k^T y_k)^2} d_k d_k^T$.

2. Retourner x_k.

Travail à réaliser

Résoudre les problèmes tests de minimisation :

$$\min_{(x,y)\in\mathbb{R}^2} f_i(x)\ i \in 1,2$$

par l'algorithme de Quasi-Newton (BFGS).

- Commenter les résultats obtenus avec différents points initiaux.
- Comparer cet algorithme avec celui de Newton.

Il reste maintenant à définir une stratégie de recherche linéaire pour le calcul du pas.

Pas de Descente

1.1.4 Pas fixe

On choisit un pas fixe pour toutes les directions de descente calculées à chaque itération.
Comment choisir un pas qui garantisse la convergence ?

1.1.5 Pas optimal

Une idée consiste à choisir un pas qui rende la fonction à minimiser la plus petite possible dans la direction choisie.

s_k solution de : $\min_{s>0} f(x_k + s\, d_k)$

En pratique, à part pour quelques problèmes, le calcul du pas optimal est coûteux, et on calcule alors un pas qui satisfait simplement les conditions de Wolfe (qui garantissent au passage la convergence globale).

Algorithme de Backtracking

Algorithm 1.4 ALGORITHME DE BACKTRACKING

Données : $x, d, 0 < \rho < 1, s = 1$

Sortie : un pas qui satisfait la première condition faible de Wolfe

1. répéter :

a. si la première condition de Wolfe n'est pas satisfaite :
$s = \rho s$

b. sinon : stop et retourner s

2. fin rèpètition .

Travail à réaliser

Implémenter l'algorithme de backtracking. Résoudre les problèmes tests de minimisation :

$$\min_{(x,y)\in\mathbb{R}^2} f(x)\ i \in 1, 2$$

par les algorithmes de Descente précedents et avec un pas calculé par l'algorithme de backtracking.

Commenter les résultats obtenus.

Algorithme de Bissection

Algorithm 1.5 ALGORITHME DE BISSECTION

Données : $x, d, \alpha = 0, s = 1, \beta = +\infty$

Sortie : Un pas qui satisfait les conditions faibles de Wolfe

1. Répéter :

a. Si la première condition de Wolfe n'est pas satisfaite :
$\beta = s, s = 0.5(\alpha + \beta)$

b. Sinon : si la deuxième condition de Wolfe n'est pas satisfaite :
On pose $\alpha = s$,, et $s = 2\alpha$ si $\beta = +\infty$, ou $s = 0.5(\alpha + \beta)$ sinon.

c. Sinon : stop et retourner s.

2. Fin répétition .

Travail à réaliser

Implémenter l'algorithme de bissection. Résoudre les problèmes tests de minimisation :

$$\min_{(x,y)\in\mathbb{R}^2} f(x)\ i \in 1,2$$

par les algorithmes de Descente précedents et avec un pas calculé par l'algorithme de bissection.

Commenter les résultats obtenus.

Pour la suite on définit la fonction ϕ :

$\phi(s) = f(x_k + sd_k)$

Algorithme d'Interpolation

Algorithm 1.6 ALGORITHME D'INTERPOLATION QUADRATIQUE COMBINÉE AVEC INTERPOLATION CUBIQUE

Données : $x, d, s_0 > 0$

Sortie : un pas qui satisfait la première condition faible de Wolfe.

1. Si $\phi(s_0) \leq \phi(0) + c_1 s_0 \phi^{'}(0)$, $s = s_0$, stop
2. Sinon :

   a. On considère le polynôme ϕ_q de degré 2 interpolant ϕ aux points $\phi(0)$, $\phi^{'}(0)$ et $\phi(s_0)$

   $$\phi_q(s) = \frac{\phi(s_0) - \phi(0) - s_0\phi'(0)}{s_0^2} s^2 + \phi'(0)s + \phi(0)$$

   b. $s_1 = argmin\ \phi_q(s)$

   Si $\phi(s_1) \leq \phi(0) + c_1 s_1 \phi^{'}(0)$, $s = s_1$, stop

   Sinon : on considère le polynôme ϕ_c de degré 3 interpolant ϕ aux points $\phi(0)$, $\phi^{'}(0)$, $\phi(s_0)$ et $\phi(s_1)$

   $$\phi_c(s) = as^3 + bs^2 + \phi'(0)s + \phi(0),$$

   avec :

   $$\begin{bmatrix} a \\ b \end{bmatrix} = \frac{1}{s_0^2 s_1^2 (s_1 - s_0)} \begin{bmatrix} s_0^2 & -s_1^2 \\ -s_0^3 & s_1^3 \end{bmatrix} \begin{bmatrix} \phi(s_1) - \phi(0) - \phi'(0)s_1 \\ \phi(s_0) - \phi(0) - \phi'(0)s_0 \end{bmatrix}$$

   $s_2 = argmin\ \phi_c(s)$

   Si $\phi(s_2) \leq \phi(0) + c_1 s_2 \phi^{'}(0)$, $s = s_2$, stop

   Sinon : $s_0 = s_1, s_1 = s_2$ et répéter l'interpolation cubique jusqu'à satisfaction de la première condition de Wolfe.

Travail à réaliser

Implémenter l'algorithme de recherche de pas par interpolation. Résoudre les problèmes tests de minimisation :

$$\min_{(x,y)\in\mathbb{R}^2} f(x)\ i \in 1,2$$

par les algorithmes de Descente précedents et avec un pas calculé par l'algorithme d'interpolation.

Commenter les résultats obtenus.

Algorithmes d'Approche et de Finition

Algorithm 1.7 ALGORITHME APPROCHE

Données : $x, d, s_0 = 0, s_1 > 0, i = 1, s_{max}$

Sortie : un pas s qui satisfait les conditions fortes de Wolfe

1.Répéter :

a. Si $\phi(s_i) > \phi(0) + c_1 s_i \phi^{'}(0)$ ou $(\phi(s_i) \geq \phi(s_{i-1})$ and $i > 1)$
$s =$ finition(s_{i-1}, s_i), stop

b. Sinon : si $|\phi^{'}(s_i)| \leq -c_2 \phi^{'}(0)$
$s = s_i$, stop

c. Sinon : si $\phi^{'}(s_i) \geq 0$
$s =$ finition(s_i, s_{i-1}), stop

d. Sinon : choisir $s_{i+1} \in [s_i s_{max}], i = i + 1$

2. Fin répétition .

Pour la fonction finition, l'ordre des arguments est important, à savoir que dans finition(s_{min}, s_{max}) ces paramètres vérifient les 3 propriétés suivantes :

1. l'intervalle (s_{min}, s_{max}) contient un pas satisfaisant les conditions fortes de Wolfe ;
2. s_{min} est choisi tel que la condition de décroissance suffisante (première condition de Wolfe) est satisfaite ;
3. s_{max} est choisi tel que $\phi^{'}(s_{min})(s_{max} - s_{min}) < 0$.

Algorithm 1.8 ALGORITHME FINITION

Données : x, d, s_{min}, s_{max}.

Sortie : un pas s qui satisfait les conditions fortes de Wolfe et qui est compris entre s_{min} et s_{max}.

1. Répéter :

Choisir un $s_j \in (s_{min}, s_{max})$ (en utilisant l'interpolation ou la bissection)

a. Si $\phi(s_j) > \phi(0) + c_1 s_j \phi^{'}(0)$ ou $\phi(s_j) \geq \phi(s_{min})$
$s_{max} = s_j$

b. Sinon :
si $|\phi^{'}(s_j)| \leq -c_2 \phi^{'}(0)$, $s = s_j$, stop
sinonsi $\phi^{'}(s_j)(s_{max} - s_{min}) \geq 0$, $s_{max} = s_{min}$,$s_{min} = s_j$
sinon $s_{min} = s_j$
fin si

c. Fin Si

2. Fin Répétition .

Explication de l'algorithme de finition :

Si s_j satisfait les conditions fortes de Wolfe, on arrête et on choisi $s = s_j$.

Sinon, si s_j satisfait la décroissance suffisante, on met à jour $s_{min} = s_j$, pour maintenir la propriété 2, et si la propriété 3 n'est pas satisfaite, on remédie à la situation avec la mise à jour : $s_{max} = s_{min}$

Faites un graphe pour mieux voir les choses.

Travail à réaliser

Implémenter les algorithmes approche + finition. Résoudre les problèmes tests de minimisation :

$$\min_{(x,y)\in\mathbb{R}^2} f(x)\ i \in 1,2$$

par les algorithmes de descente précedents et avec un pas calculé par l'algorithme d'approche + finition.

Faites une comparaison générale entre les différents algorithmes précédents.

1.2 Régions de confiance

(traiter cette partie à la fin si le temps le permet)

Principe

L'idée de la méthode des régions de confiance est d'approcher $f(x_k)$ par une fonction modèle plus simple m_k dans une région $R_k = (x_k + p; ||p|| < \Delta_k)$ pour un Δ_k fixé.

Cette région dite "de confiance" doit être suffisament petite pour que

$$m_k(x_k + p) \sim f(x_k + p).$$

Le principe est que, au lieu de résoudre l'équation : $f(x_{k+1}) = min_{||p||<\Delta} f(x_k + p)$,n on résout :

$$m_k(x_{k+1}) = min_{||p||<\Delta} m_k(x_k + p) \quad (1)$$

Si la différence entre $f(x_{k+1})$ et $m_k(x_{k+1})$ est trop grande, on diminue le Δ_k (et donc la région de confiance) et on résout le modèle (1) à nouveau. Un avantage de cette méthode est que toutes les directions sont prises en compte. Par contre, il faut faire attention à ne pas trop s'éloigner de x_k, car la fonction m_k n'approche proprement f que sur une région proche de x_k.

Exemple de modèle : l'approximation de Taylor à l'ordre 2 (modèle quadratique) :

$$m_k = f(x_k) + g_k^T p + \frac{1}{2} p^T B_k p \quad (2)$$

Algorithme

Algorithm 1.9 MÉTHODE DES RÉGIONS DE CONFIANCE

Données : $\Delta_{max} > 0$, $\Delta_0 \in (0, \Delta_{max})$ et $\nu \in [0, \frac{1}{4})$:

Sortie : une approximation de la solution du problème : $\min_{x\in\mathbb{R}^n} f(x)$

1.Tant que le test de convergence n'est pas satisfait :

a. Calculer approximativement p_k la solution du modèle m_k (2) dans la région R_k ;

b. Calculer $\rho_k = \frac{f(x_k)-f(x_k+p_k)}{m_k(0)-m_k(p_k)}$

c. Si $\rho_k < \frac{1}{4}$

$\Delta_{k+1} = \frac{1}{4}\Delta_k$

d. Sinon Si ($\rho_k > \frac{3}{4}$ et $||p_k|| == \Delta_k$)

$\Delta_{k+1} = min(2\Delta_k, \Delta_{max})$

e. Sinon

$\Delta_{k+1} = \Delta_k$.

f. Si $\rho_k > \nu$

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

e. Sinon

$x_{k+1} = x_k$;

2. Retourner x_k.

Travail à réaliser

1. Implémenter l'algorithme des régions de confiance.
2. Le tester sur les problèmes de tests fournis.

2 Optimisation avec contraintes

Dans cette partie, nous nous intéressons à la résolution des problèmes sous contraintes. Le problème se présente donc sous la forme suivante :

$$\min_{x \in \mathbb{R}^n} f(x) \quad sous\ la\ contrainte\ : x \in C,$$

où C est un sous-ensemble non vide de $\mathbb{R}^n$.

2.1 Lagrangien Augmenté

Principe

La méthode du lagrangien augmenté appartient à une classe d'algorithmes qui permettent la résolution des problèmes avec contraintes. Elle ressemble aux méthodes de pénalisation, dans lesquelles on résout le probleme avec contraintes à travers une suite de problèmes sans contraintes.

Algorithme

Algorithm 2.1 MÉTHODE DU LAGRANGIEN AUGMENTÉ POUR LA RÉSOLUTION D'UN PROBLÈME AVEC CONTRAINTES D'ÉGALITÉ

Le problème à résoudre est de type $argmin_x f(x)$, *sous les contraintes* $c_i(x) = 0, i \in I$

Données : $\mu_0 > 0$, $\tau > 0$, λ^0, et le point de départ x_0^s.

Sortie : une approximation de la solution du problème avec contraintes.

1. Tant que pas de convergence, répéter

a. Calculer approximativement un minimiseur x_k du problème sans contraintes suivant : $argmin_x\ L_A(x, \lambda^k, \mu_k) = f(x) - \lambda_k^T c(x) + \frac{\mu_k}{2}\|c(x)\|^2$ avec x_k^s comme point de départ,

Finir lorsque $||L_A(., \lambda^k, \mu_k)|| \leq \tau$.

b. Si convergence, on s'arrête ici.

c. Mettre à jour les multiplicateurs de Lagrange : $\lambda_i^k = \lambda_i^k - \mu_k c_i(x_k) \forall i \in I$;

d. $\mu_{k+1} \geq \mu_k$;

e. $x_{k+1}^s = x_k$;, $k = k + 1$,

Travail à réaliser

- Choisir des critères d'arrêt pour la convergence de l'algorithme.
- Implémenter l'algorithme de lagrangien augmenté, en utilisant les différentes méthodes qui ont été vues en première partie pour la resolution de la suite de problémes sans contrainte.
- Résoudre le problème d'optimisation suivant :

$$\begin{cases} \min & f_2(x,y) \\ (x,y) \in \mathbb{R}^2 & \\ x^2 + y^2 & = \quad 1.5 \end{cases}$$

- Commenter les résultats obtenus.

– Que proposez-vous comme méthode pour la résolution des problèmes avec des contraintes à la fois d'égalité et d'inégalité ?
– Résoudre le problème d'optimisation suivant avec votre nouvelle méthode :

$$\begin{cases} \min & q(x,y) = & (x-1)^2 + (y-2.5)^2 \\ (x,y) \in \mathbb{R}^2 & & \\ x - 2y + 2 & \geq & 0 \\ -x - 2y + 6 & \geq & 0 \\ -x + 2y + 2 & \geq & 0 \\ x & \geq & 0 \\ y & \geq & 0 \end{cases}$$