Université du Sud - Toulon - Var
ISITV - 2ème année Marine

Optimisation Non Linéaire
Travaux Pratiques I

Mehmet Ersoy

Optimisation sans contraintes : algorithmes

Le but de ce TP est de tester différentes méthodes d'optimisation sans contraintes. La programmation se fera dans un premier temps avec Matlab. Pour avoir un aperçu des fonctions, on pourra utiliser la fonction `contour` ou `surf` de matlab.

On veut résoudre numériquement le problème suivant :

$$\inf\{J_0(x) = \frac{1}{2}(Ax, x) - (b, x);\ x \in \mathbb{R}^3\} \text{ où } A = \begin{pmatrix} 1 & 4 & 3 \\ -3 & 6 & 3 \\ -1 & 0 & 7 \end{pmatrix} \quad \text{et} \quad b = \begin{pmatrix} -1 \\ 1 \\ -1 \end{pmatrix}.$$

1. Montrer que J_0 est strictement convexe. En déduire l'existence et unicité du minimum.
2. Programmer la méthode du gradient à pas constant `gradcst` .
   (a) Tester diverses valeurs pour le pas choisi et le point de départ pour une précision $\varepsilon = 1.0e^{-06}$.
   (b) On comparera la vitesse de convergence (i.e. le nombre d'itérations nécessaires avant l'arrêt de l'algorithme) et le temps de calcul (nous utiliserons les commandes `tic gradcst(...)tic`).
   (c) Faire le lien avec le théorème de convergence du cours.
   (d) Calculer le produit scalaire du gradient de J en x avec un vecteur v. Comparer avec le résultat obtenu en faisant la différence finie :
   $$\nabla J(x) \cdot v \approx \frac{J(x + tv) - J(x - tv)}{2t}.$$
   En déduire une manière de calculer numériquement le gradient de J sans faire le calcul théorique puis réécrire la méthode du gradient à pas constant en intégrant ce calcul approché.
3. Reprendre la question 3 en utilisant l'algorithme de Newton `newton`.
4. Faire un tableau comparatif des méthodes à pas constant, Newton, à pas constant et Newton avec calcul approché du gradient pour une précision $\varepsilon = 1.0e^{-06}$ en terme de nombres d'itérations et temps de calcul. Conclure.
5. Appliquer la méthode à pas constant, Newton, à pas constant et Newton avec calcul approché du gradient à la fonction de Rosenbrock :
   $$J_1(x, y) = (1 - x)^2 + 100(y - x^2)^2.$$
   On testera divers points initiaux dont $(-2, 0)$, $(-2, 2)$, $(0, 2)$ et $(2, 2)$ et nous utiliserons la fonction `contdir` pour visualiser le trajet vers le minimum.

1 Travail à faire

Vous rédigerez un court rapport contenant les réponses aux questions théoriques et le tableau de performance mentionné ci-dessus accompagné de commentaires, ainsi que le code numérique. Ce compte-rendu est à remettre à la prochaine séance. La qualité de la rédaction, la clarté et la précision des arguments sera prise en compte dans l'évaluation.

Méthode de la plus grande pente

Méthode de la plus grande pente : pas constant

On veut résoudre le problème

$$(P) \qquad \inf\{J(x) \,:\, x \in \mathbb{R}^d\},$$

on emploie l'algorithme suivant :

1. $x^0 \in \mathbb{R}^d$ fixé, pas $\rho > 0$ fixé, précision $\varepsilon > 0$ fixée,
2. $x \leftarrow x^0$,
3. TANT QUE $\|\nabla J(x)\| > \varepsilon$ FAIRE
   - $\rho \leftarrow \rho$
   - $x \leftarrow x - \rho \nabla J(x)$
4. afficher x.

Méthode de Newton

On veut résoudre le problème

$$(P) \qquad \inf\{J(x) \,:\, x \in \mathbb{R}^d\},$$

on emploie l'algorithme suivant :

1. $x^0 \in \mathbb{R}^d$ fixé, pas $\rho > 0$ fixé, précision $\varepsilon > 0$ fixée,
2. $x \leftarrow x^0$,
3. Tant que $\|\nabla J(x)\| > \varepsilon$ faire
   - $x = x - \delta$ avec $DF(x)\delta = F(x)$ où $F = \nabla J$.
   - $x \leftarrow y$
4. afficher x.

Méthode des différences finies

Dans les algorithmes précédents, on peut remplacer le calcul de $\nabla J(x)$ par l'algorithme qui calcul un vecteur g qui est une approximation par différences finies de $\nabla J(x)$:

1. $x \in \mathbb{R}^d$ fixé, paramètre $0 < \delta << 1$ fixé (plus petit que la précision ε),
2. Pour $i = 1$ à d faire
$$g_i \;:=\; \frac{J(x + \delta\, e_i) - J(x - \delta\, e_i)}{2\,\delta}$$
où e_i est le i-ième vecteur de la base canonique de $\mathbb{R}^d$,
3. renvoyer g.

.

Université du Sud - Toulon - Var
ISITV - 2ème année Marine

Optimisation Non Linéaire
Travaux Pratiques II

Mehmet Ersoy

Optimisation sans contraintes : résolution d'une EDO

L'objectif de ce TP est d'utiliser deux nouvelles méthodes d'optimisations (essentiellement des variantes de la méthode du gradient à pas constant et de la méthode de Newton) afin d'approcher les solutions de l'équation de Laplace :

$$-u''(x) + c(x)u(x) = f(x) \quad x \in (0,1) \tag{1}$$

satisfaisant les conditions de Dirichlet

$$u(0) = 0 \text{ et } u(1) = 0.$$

Ce problème modélise le fléchissement d'une poutre fixé en ses extrémités sous la contrainte de force f. c est une fonction à la valeurs dans $\mathbb{R}^+$.

1 Quelques algorithmes supplémentaires

En complément aux TP I, nous allons aussi programmer les deux méthodes suivantes. Dans aucuns cas, nous calculerons les gradients. Nous utiliserons la fonction `Grad.m` que nous avons programmé au TP I.

1.1 Méthode du gradient à pas optimal

Cette algorithme permet de choisir le pas de manière à minimiser la fonction $\rho \in \mathbb{R} \to J(x+\rho d) \in \mathbb{R}$ de sorte que $J(x+\rho d) < J(x)$ soit satisfait à chaque itérations. Ainsi, le pas ρ est choisit de manière optimale. En pratique on ne résout pas ce problème de minimisation. On utilise des algorithmes de recherches (la plus célèbre étant la recherche linéaire de Wolfe). Cependant, dans le cadre de ce tp, nous utiliserons la fonction `fminbnd(@fun,rho1,rho2)` qui permet de rechercher le minimum d'une fonction réelle `fun(x)` (voir l'aide Matlab pour plus de précision) dans un intervalle (ρ_1, ρ_2). Bien sûr cette façon de procédé n'est pas la plus économe!!!

L'algorithme général pour la résolution du problème

$$\inf\{J(x) \,:\, x \in \mathbb{R}^d\}, \tag{P}$$

s'écrit donc

1. $x^0 \in \mathbb{R}^d$ fixé, pas $\rho > 0$, précision $\varepsilon > 0$ fixée,
2. $x \leftarrow x^0$,
3. Tant que $\|\nabla J(x)\| > \varepsilon$ faire
   - $d \leftarrow -\nabla J(x)$
   - $\rho \leftarrow \rho > 0$ minimum de $J(x+\rho d)$. (Nous utiliserons en pratique la fonction `fminbnd(fun,rho1,rho2)`) avec `rho1` petit et `rho2` suffisamment "grand".
   - $x \leftarrow x + \rho d$
4. afficher x.

Programmer cette méthode (`optim(J,x0,h,eps,iter)`) et la comparer avec l'algorithme du gradient à pas constant sur la fonctionnelle $J(x,y) = x^2 + y^2 - x - y$. Le minimum est bien sur $(1,1)$. Que constatez vous.

1.2 Une méthode Quasi-Newton (BFGS)

Au cours du tp précédent, nous avons vu que l'algorithme de Newton nécessite l'inversion de la matrice Hessienne de la fonctionnelle J. En pratique, si le système est trop grand l'inversion de la matrice Hessienne est couteuse. Les méthodes Quasi-Newton consiste donc à remplacer la hessienne $D^2J(x_k) := \nabla F(x_k)$ par une matrice B_k plus facile à calculer qui se traduit par

$$x_{k+1} - x_k = B_k(F(x_{k+1}) - F(x_k))$$

avec $F(x) := \nabla J(x)$. Cette expression n'est rien d'autre que la généralisation de la méthode de la sécante pour un problème de dimension 1.

L'algorithme général pour la résolution du problème

$$(P) \qquad \inf\{J(x) \,:\, x \in \mathbb{R}^d\},$$

est

1. $x^0 \in \mathbb{R}^d$ fixé, pas $\rho > 0$, précision $\varepsilon > 0$ fixée, $B^0 = I_d$
2. $B \leftarrow B^0$,
3. Tant que $\|\nabla J(x)\| > \varepsilon$ faire
   - $d \leftarrow -B\nabla J(x_0)$
   - $\rho \leftarrow \rho > 0$ minimum de $J(x_0 + \rho d)$.
   - $x \leftarrow x_0 + \rho d$
   - $s \leftarrow x - x_0$
   - $y \leftarrow \nabla J(x) - \nabla J(x_0)$
   - $B \leftarrow B - \dfrac{sy^tB + Bys^t}{y^ts} + \left(1 + \dfrac{y^tBy}{y^ts}\right)\dfrac{ss^t}{y^ts}$
   - $x_0 \leftarrow x$
4. afficher x.

La manière dont B est défini est due à Broyden-Fletcher-Goldfarb-Shanno et est communément appelé BFGS. C'est une des méthodes les plus performantes en optimisation non linéaire.

Programmer la méthode BFGS (`bfgs(J,x0,h,eps,iter)`. Appliquer cette méthode à la fonctionnelle de de Rosenbrock en partant du point $(-0.9, 1)$.

2 Résolution d'une équation différentielle

Pour résoudre l'équation (1), on se donne une subdivision régulière de $[0, 1]$ en N intervalles de longueur $h = \dfrac{1}{N+1}$ et on considère les nœuds $x_i = ih$, $i = 0, \ldots, N+1$. On cherche alors une solution approchée aux nœuds $u_i \approx u(x_i)$ vérifiant l'équation (discrétisée par une méthode différences finies) :

$$(2) \qquad \begin{cases} \dfrac{-u_{i-1} + 2u_i - u_{i+1}}{h^2} + c(x_i)u_i = f(x_i) \quad \forall 1 \leqslant i \leqslant N \\ u_0 = u_{N+1} = 0 \end{cases}$$

1. Montrer que l'équation (2) peut s'écrire sous la forme

   $$(3) \qquad A_N U_N = f_N$$

   où $U_N = (u_i)_{1\leqslant i\leqslant N}$, $U_N = (f(x_i))_{1\leqslant i\leqslant N}$ et

   $$\frac{1}{h^2}\begin{pmatrix} 2 + h^2c_1 & -1 & & 0 \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & -1 \\ 0 & & -1 & 2 + h^2c_N \end{pmatrix}$$

2. Montrer que : U_n est solution de (3) ssi U_n minimise $J_N(U) = \frac{1}{2}(A_N U, U) - (f_N, U)$.

Dans toute la suite, on prend $c(x) = 1$ et $f(x) = 1$.

(a) Vérifier que la fonction u ci dessous est solution de l'équation (1) :

$$u(x) = -e^x \frac{-1 + e^{-1}}{-e + e^{-1}} + e^{-x} \frac{e - 1}{-e + e^{-1}} + 1.$$

Pour la fonction `fminbnd`, nous fixons désormais `rho1` à $1.0\ 10^{-12}$ et `rho2` à 100. Nous choisirons x_0 un vecteur unité, le nombre d'itérations maximales à 10000 et $\rho = 10$ initial (pour ne pas être embêté).

(b) Résoudre l'équation (1) en utilisant les algorithmes à pas optimal et BFGS avec $N = 10$ pour $\varepsilon = 0.1$, 0.01 puis 0.001. Tracer les solutions approchées et la solution exacte sur un même graphique pour chaque ε. Que constatez vous ?

(c) Résoudre le problème pour $N = 2, 5, 10, 20$ et 40. Pour toutes les simulations reporter le nombre d'itérations, le temps de calcul nécessaire pour trouver U_N et tracer la courbe qui à N associe le temps de calcul. Faire de même pour la courbe qui à N associe le nombre d'itérations. Conclure.

Nous utiliserons le petit script suivant :

```
clear all,clc,clf

rho = 10;
eps= 1e-01;
iter=10000;

N=[2 5 10 20 40];
l = 1;
for n=N
 h = 1/(n+1);
 x0=ones(1,n);
 %BFGS
 tic
 [u0,k] = bfgs(@J,x0',rho,eps,iter);
 T0(l) = toc;
 k0(l)=k;
 %Pas optimal
 tic
 [u1,k] = optim(@J,x0',rho,eps,iter);
 T1(l) = toc;
 k1(l)=k;
 l = l+1;
 xx = [0:h:1];
 plot(xx,[ 0 u0' 0],xx,[ 0 u1' 0],xx,u(xx))
 legend('bfgs','optim','uex')
 pause(0.5)
end
subplot(211)
plot(N,T0,N,T1)
legend('bfgs', 'optim')
subplot(212)
plot(N,k0,N,k1)
legend('bfgs', 'optim')
```

3 Travail à faire

Vous rédigerez un court rapport contenant les réponses aux questions théoriques, le tableau de performance mentionné ci-dessus accompagné de commentaires, des figures correspondantes et de votre code numérique commenté. Ce compte-rendu (format NomsBinome.zip) est à envoyer pour la prochaine séance à l'adresse électronique `Mehmet.Ersoy@univ-tln.fr`. La qualité de la rédaction, la clarté et la précision des arguments sera prise en compte dans l'évaluation.

.

Université du Sud - Toulon - Var
ISITV - 2ème année Marine

Optimisation Non Linéaire
Projet (TP III)
Un problème de calcul des variations

Mehmet Ersoy

Optimisation sans contraintes : la courbe brachistochrone
Le toboggan le plus rapide

On souhaite fabriquer un toboggan partant d'un point $A = (x_A, y_A)$ et arrivant à un point $B = (x_B, y_B)$. L'objectif est de minimiser le temps mis par un petit bonhomme lâché (soumis à la seule force de gravité) sans vitesse depuis A pour qu'il atteigne le point B.

Un peu de modélisation

Soient $A = (x_A, y_A)$ et $B = (x_B, y_B)$ deux points de l'espace. Le temps de parcours T pour aller d'un point A au point B le long d'un chemin γ est donnée par

$$T(\gamma) = \int_0^T dt = \int_\gamma \frac{ds}{v}$$

où s est l'abscisse curviligne du point matériel (le petit bonhomme) et v sa vitesse. Sous la seule influence de la force gravitationnelle et $y_A > y_B$, l'énergie cinétique est donnée par

$$E_c = \frac{1}{2}mv^2$$

et d'énergie potentielle

$$E_p = mgy$$

où m est la masse du point matériel et $y = f(x)$ (une paramétrisation de la courbe γ). f modélisera donc la forme du toboggan. D'après le principe de la conservation de l'énergie total du système, on en déduit la relation

$$\frac{1}{2}mv^2 + mgy = \frac{1}{2}mv_A^2 m + mgy_A = mgy_A.$$

Par conséquent, on obtient

$$v = \sqrt{2g(y_A - y)} = \sqrt{2g(y_A - f(x))}.$$

D'autre part, on sait que

$$\frac{ds}{dx} = \sqrt{1 + (f'(x))^2}.$$

La fonctionnelle ...

Le temps de parcours s'écrit donc :

$$T(f) = \int_{x_A}^{x_B} \sqrt{\frac{1 + (f'(x))^2}{2g(y_A - f(x))}}\, dx.$$

$T(f)$ désigne donc une fonctionnelle sur l'espace des fonctions $C^1([x_A, x_B], \mathbb{R})$ qui satisfont les contraintes $f(x_A) = y_A$ et $f(x_B) = y_B$. Le problème de minimisation s'écrit alors

$$\min\{T(f); f : [x_A, x_B] \to \mathbb{R}, f(x_A) = y_A, f(x_B) = y_B\}. \tag{1}$$

Comment s'y prendre

Dans la suite nous fixons les valeurs suivantes : $A = (0, y_A)$ et $B = (1, 0)$ avec $y_A > 0$. Soit $N \geqslant 1$ fixé. On note $h = \dfrac{1}{N+1}$ le pas de discrétisation et $x_i = ih$, $i = 0, \ldots, N+1$ les noeuds du maillage. On considère l'ensemble $\mathcal{A}_N$ des fonctions affines par morceaux tel que $f \in \mathcal{A}_N$ si et seulement si

$$f : [0,1] \to \mathbb{R},\ f(0) = y_A,\ f(1) = 0$$

et

$$\forall n \in \{0, \ldots, N\}, \quad f(x) = y_n w_n(x),\ \forall x \in [x_n, x_{n+1}],\ y_n = f(x_n) \in \mathbb{R}$$

où $w_n(x)$ est la traditionnelle fonction chapeaux ($w_n(x_m) = \delta_{nm}$). Trouver f dans cet espace discret, revient donc à chercher un vecteur $y \in \mathbb{R}^{N+2}$ tel que $y_0 = y_A$ et $y_{N+1} = y_B$. Notons

$$\mathcal{B}_N := \{y \in \mathbb{R}^{N+2}, y_0 = y_A, y_{N+1} = 0\}$$

cet ensemble.

1. Montrer que

$$\forall n = 1 \ldots N, \quad w_n(x) = \begin{cases} w_1\left(\dfrac{x - x_{n-1}}{h}\right) & \text{si} \quad x \in [x_{n-1}, x_n], \\ w_0\left(\dfrac{x - x_n}{h}\right) & \text{si} \quad x \in [x_n, x_{n+1}], \\ 0 & \text{sinon.} \end{cases}$$

2. En posant

$$F(x, y, z) = \sqrt{\frac{1 + z^2}{2g(y_A - y)}},$$

montrer que

$$\int_{x_n}^{x_{n+1}} F(x, f(x), f'(x))\, dx = \frac{2}{\sqrt{2g}} \frac{\sqrt{h^2 + (y_{n+1} - y_n)^2}}{\sqrt{y_A - y_{n+1}} + \sqrt{y_A - y_n}}$$

3. En déduire que le problème initiale (1) s'écrit sous la forme

$$\min\{T_N(y); y \in \mathcal{B}_\mathcal{N}\} \tag{2}$$

où

$$T_N(y) = \sum_{n=0}^{N} \int_{x_n}^{x_{n+1}} F(x, f(x), f'(x))\, dx.$$

Expliciter la forme de T_N.

4. Nous initialiserons l'algorithme d'optimisation de notre choix de façon judicieuse. Pour $N = 1, 2, 5, 10, 20$ et $y_A = 1$, tracer la courbe optimale sur un même graphique et la vitesse.
5. Faire de même pour d'autres valeurs de y_A à N fixé.
6. A (N, y_A) fixé, construire un polynôme d'ordre 1 et 2 (nous choisirons $p(1/2) = 1/2$) qui relie A à B. Tracer ces courbes et leurs vitesses respectives. Comparer les temps de calculs.

1 Travail à faire

Vous rédigerez un court rapport contenant les réponses aux questions théoriques, le tableau de performance mentionné ci-dessus accompagné de commentaires, des figures correspondantes et de votre code numérique commenté. Ce compte-rendu (format NomsBinome.zip) est à envoyer pour la prochaine séance à l'adresse électronique `Mehmet.Ersoy@univ-tln.fr`. La qualité de la rédaction, la clarté et la précision des arguments sera prise en compte dans l'évaluation.

.