Optimisation Numerique:
Problemes d'Optimisation avec et sans contraintes

Thibault MEUNIER

9 fevrier 2017

Sommaire

1 Algorithme de Newton local

La resolution d'un probleme lineaire est une operation simple. Ainsi, l'algorithme de Newton local consiste en l'approximation de la fonction a minimiser par son developpement lineaire à l'ordre 2. En resolvant le probleme lineaire, on trouve une solution au probleme initial.

1.1 Pertinence

L'algorithme de Newton presente un interet majeur qui est sa rapidite. En effet, l'approximation faite de la foncion a l'ordre 2 permet une convergence rapide vers un minima.
Il presente cependant un inconvenient majeur qui est une convergence non assuree, et le minima trouve n'est que local. Selon le point de depart choisi, l'algorithme peut tendre vers l'infini ou meme iterer autour d'un minima sans l'atteindre.
Par la suite nous essaierons de l'ameliorer pour tirer partie de ses proprietes de convergence rapide toute en assurant cette convergence.

1.2 Reponses aux questions

1. La fonction f_1 est égale à son développement de Taylor à l'ordre 2. En une iteration on resout donc le systeme lineaire associe, avec pour solution le minimum de f_1.

2. L'algorithme peut ne pas converger pour certains points initiaux de f_2. En effet, comme explique precedemment, l'approximation de la fonction par son developpement a l'ordre 2 conduit a une fonction pouvant conduire dans le sens oppose a un minimum. Le systeme obtenue peut ne pas etre inversible.

1.3 Tests

Fichiers testsNewton.m

Soient f_1 et f_2 comme definie dans le sujet :

$$f_1(x) = 2(x_1 + x_2 + x_3 - 3)^2 + (x_1 - x_2)^2 + (x_2 - x_3)^2$$

$$f_2(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$$

On effectue les tests suivants afin de valider l'algorithme ecrit. Chaque test cherche a minimiser f en partant du point x_0.

- $f = f_1, \quad x_0 = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^T$ 2 tours realises

$$x^* = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}^T, \quad f_1(x^*) = 2.46 * 10^{-31} \approx 0$$

- $f = f_1, \quad x_0 = \left[\begin{array}{ccc} 10 & 3 & -2.2 \end{array} \right]^T$ 2 tours realises

$$x^* = \left[\begin{array}{ccc} 1 & 1 & 1 \end{array} \right]^T, \quad f_1(x^*) = 1.77 * 10^{-30} \approx 0$$

- $f = f_2, \quad x_0 = \left[\begin{array}{cc} -1.2 & 1 \end{array} \right]^T$ 8 tours realises

$$x^* = \left[\begin{array}{cc} 1 & 1 \end{array} \right]^T, \quad f_2(x^*) = 4.94 * 10^{-30} \approx 0$$

- $f = f_2, \quad x_0 = \left[\begin{array}{cc} 10 & 0 \end{array} \right]^T$ 6 tours realises

$$x^* = \left[\begin{array}{cc} 1 & 1 \end{array} \right]^T, \quad f_2(x^*) = 4.94 * 10^{-28} \approx 0$$

- $f = f_2, \quad x_0 = \left[\begin{array}{cc} 0 & \frac{1}{200} + 10^{-12} \end{array} \right]^T$ non convergence
  En effet le point $x_0^* = \left[\begin{array}{cc} 0 & \frac{1}{200} \end{array} \right]^T$ est un point singulier de la hessienne de f_2. Un simple decalage de 10^{-12} ne permet pas un convergence franche du fait de la precision des calculs effectues.

2 Regions de confiance : pas de Cauchy

Le principal defaut de l'algorithme de Newton est sa non convergence. L'algorithme des region de confiance se base donc sur ce dernier, qui fonctionne bien dans la plupart des situation, et s'attache a resoudre ce probleme de convergence. Pour ce faire, il met en place des regions dites "de confiance" permettant a l'algorithme d'evoluer sans difficulte. On diminue ou augmente la taille de cette region au fur et a mesure afin de trouver un minimum. L'algorithme beneficiera ainsi d'une vitesse de convergence quasiment quadratique.

2.1 Pertinence

L'avantage du pas de Cauchy est egalement son principal inconvenient. Il se base sa minimisation sur le calcul du gradient de la fonction en le point courant et choisie la direction de plus forte pente. Ainsi il effectue un calcul simple qui conduira en general vers le minimum, mais il peut aussi s'egarer et augmenter significativement le nombre d'iterations necessaires a la resolution du probleme de minimisation initial.

2.2 Reponses aux questions

1. La fonction f_1 est égale à son développement de Taylor à l'ordre 2. Sur f_1, l'agorithme de Newton converge en une unique iteration car il minimise directement la quadratique. Avec l'assertion qu'il effectue, le pas de Cauchy effectue plusieurs pas avant de trouver le meme minimum local. Il aurait pu etre plus efficace sur des fonctions complexes.

2. On peut jouer sur differents parametres pour modifier optimiser l'execution de l'algorithme des regions de confiances :

- Δ_{max} : rayon de confiance maximal
- γ_1 et γ_2 : facteurs d'agrandissement et de reduction de la region de confiance
- η_1 et η_2 : criteres d'agrandissement et de reduction de la region de confiance

2.3 Tests

Fichiers testsCauchy.m

On teste le pas de Cauchy sur des fonctions de la forme

$$q(s) = s^T g + \frac{1}{2} s^T H s$$

- $g = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad H = \begin{bmatrix} 7 & 0 \\ 0 & 2 \end{bmatrix}, \Delta = 1$ donne: $s = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$
- $g = \begin{bmatrix} 6 \\ 2 \end{bmatrix}, \quad H = \begin{bmatrix} 7 & 0 \\ 0 & 2 \end{bmatrix}, \Delta = 1$ donne $s = \begin{bmatrix} -0.9231 \\ -0.3077 \end{bmatrix}$
- $g = \begin{bmatrix} -2 \\ 1 \end{bmatrix}, \quad H = \begin{bmatrix} -2 & 0 \\ 0 & 10 \end{bmatrix}, \Delta = 1$ donne $s = \begin{bmatrix} 0.8944 \\ -0.4472 \end{bmatrix}$

3 Algorithme des Regions de Confiance

3.1 Tests

Fichiers testsRegionConfiance.m

On effectue ces tests sur les fonctions f_1 et f_2 avec les paramètres suivants:

$$\eta_1 = 0.25, \eta_2 = 0.75, \gamma_1 = 0.5, \gamma_2 = 2, \Delta_0 = 2, \Delta_{max} = 10 \times |x_0|$$

- $f = f_1, \quad x_0 = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^T$ 35 tours realises

$$x^* = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}^T, \quad f_1(x^*) = 1.97 * 10^{-11} \approx 0$$

- $f = f_1, \quad x_0 = \begin{bmatrix} 10 & 3 & -2.2 \end{bmatrix}^T$ 32 tours realises

$$x^* = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}^T, \quad f_1(x^*) = 1.84 * 10^{-10} \approx 0$$

- $f = f_2, \quad x_0 = \left[\begin{array}{cc} -1.2 & 1 \end{array} \right]^T$ 1000 tours realises

$$x^* = \left[\begin{array}{cc} 1 & 1 \end{array} \right]^T, \quad f_2(x^*) = 8.86 * 10^{-4} \approx 0$$

On atteint ici le nombre maximum d'iterations autorises. La convergence est donc tres lente.

- $f = f_2, \quad x_0 = \left[\begin{array}{cc} -1.2 & 1 \end{array} \right]^T$ 9 tours realises

$$x^* = \left[\begin{array}{cc} 0.99 & 0.98 \end{array} \right]^T \approx \left[\begin{array}{cc} 1 & 1 \end{array} \right]^T, \quad f_2(x^*) = 1.50^{-5} \approx 0$$

La pente etant faible, la solution fournie est approximative.

- $f = f_2, \quad x_0 = \left[\begin{array}{cc} 0 & \frac{1}{200} + 10^{-12} \end{array} \right]^T$ 516 tours realises

$$x^* = \left[\begin{array}{cc} 1 & 1 \end{array} \right]^T, \quad f_2(x^*) = 5.81 * 10^{-12} \approx 0$$

Comme attendu, l'algorithme converge sur un cas qu'il lui etait impossible de resoudre auparavant.

4 Newton pour les equations non lineaires

L'algorithme suivant est un sous-probleme necessaire a la resolution de l'algorithme de More-Sorensen. Nous ne resoudrons donc pas toutes les equations lineaires mais celles de la forme :

$$||s(\lambda)||^2 - \delta^2 = 0, \quad ||s(\lambda)||^2 = \sum \frac{\alpha_i^2}{(\lambda + \beta_i)^2}$$

4.1 Reponses aux questions

1. On s'arrete lorsque λ_{min} ou λ_{max}, bornes de l'intervalle au sein duquel on cherche la solution, verifie a peu pres l'equation (avec une precision de ϵ).

2. Si l'utilisateur ne fournie pas de λ_{min} ou de λ_{max}, on peut demarer avec un intervalle quelconque en augmentant sa taille au fur et a mesure. Tant $f(\lambda_{min}) * f(\lambda_{max}) > 0$, on double la taille de l'intervalle. Ceci presente l'interet de trouver rapidement un intervalle valide. On pourra par la suite affiner cet intervalle si necessaire en reduisant l'une ou l'autre des bornes. Cette operation pourrait etre facilement implementee et surcharger l'implementation existante.

4.2 Tests

Fichiers testsNewtonPlus.m

On s'interesse d'abord a la fonction ϕ_1 definie comme suit

$$\phi_1(\lambda) = ||s(\lambda)||^2 - \delta^2, \quad ||s(\lambda)||^2 = \sum \frac{\beta_i^2}{(\lambda_i + \lambda)^2}$$

- $\lambda = (2, 14), \quad \delta = 0.5$

$$\lambda = 3.5$$

- $\lambda = (-38, 20), \quad \delta = 0.2$

$$\lambda = 20$$

- $\lambda = (-38, 20), \quad \delta = 0.7$

$$\lambda = 1.4360$$

5 Region de confiance : pas de More-Sorensen

5.1 Pertinence

Le pas de Cauchy peut converger lentement vers la solution car il base sa descente sur celle du gradient. Le pas de More-Sorensen pour sa par resout le probleme d'ordre 2 a l'interieur de la regoin de confiance, et considere donc d'autres directions que celle du gradient.

5.2 Reponses aux questions

1. La decroissance obtenue a l'aide du pas de More-Sorensen est bien plus importante que celle du pas de Cauchy du fait de l'approximation consideree. En effet, le pas de Cauchy se base uniquement sur la pente du gradient, approximant donc grossierement la fonction a l'ordre 1. Le pas de More-Sorensen approche la fonction plus finement et resout un probleme d'ordre 2. La convergence est ainsi plus rapide.

2. L'algorithme de More-Sorensen cherche a minimiser une quadratique sur la region de confiance, et approche donc mieux la fonction pour trouver une descente appropriee. Cela se fait au detriement d'un calcul de valeurs propres qui constitue une operation couteuse.
Apres implantation et test, bien que son calcul soit plus couteux, le pas de More-Sorensen fait converger plus rapidement l'algorithme.

5.3 Tests

Fichiers testsRegionConfiance.m

On s'interesse ici uniquement a la resolution du probleme initial. On prendra les parametres suivants:

$$\eta_1 = 0.25, \eta_2 = 0.75, \gamma_1 = 0.2, \gamma_2 = 1.5, \Delta_0 = 2, \Delta_{max} = 10 \times (|x_0|)$$

- $f = f_1, \quad x_0 = \left[\begin{array}{ccc} 1 & 0 & 0 \end{array} \right]^T$ 1 tours realises

$$x^* = \left[\begin{array}{ccc} 1 & 1 & 1 \end{array} \right]^T, \quad f_1(x^*) = 3.08 * 10^{-31} \approx 0$$

- $f = f_1, \quad x_0 = \left[\begin{array}{ccc} 10 & 3 & -2.2 \end{array} \right]^T$ 3 tours realises

$$x^* = \left[\begin{array}{ccc} 1 & 1 & 1 \end{array} \right]^T, \quad f_1(x^*) = 4.93 * 10^{-31} \approx 0$$

- $f = f_2, \quad x_0 = \left[\begin{array}{cc} -1.2 & 1 \end{array} \right]^T$ 27 tours realises

$$x^* = \left[\begin{array}{cc} 1 & 1 \end{array} \right]^T, \quad f_2(x^*) = 5.78 * 10^{-17} \approx 0$$

- $f = f_2, \quad x_0 = \left[\begin{array}{cc} 10 & 0 \end{array} \right]^T$ 47 tours realises

$$x^* = \left[\begin{array}{cc} 1 & 1 \end{array} \right]^T, \quad f_2(x^*) = 7.45^{-4} \approx 0$$

- $f = f_2, \quad x_0 = \left[\begin{array}{cc} 0 & \frac{1}{200} + 10^{-12} \end{array} \right]^T$ 40 tours realises

$$x^* = \left[\begin{array}{cc} 1 & 1 \end{array} \right]^T, \quad f_2(x^*) = 3.08 * 10^{-17} \approx 0$$

6 Algorithme du Lagrangien augmente

6.1 Réponses aux questions

1. On a fixe le nombre maximal d'iteration a 50. On prend egalement les parametres d'execution suivants:

$$f = f_1 \quad \tau = 1.5 \quad \eta_0 = 0.1258925 \quad \alpha = 0.1 \quad \beta = 0.9 \quad x_0 = \left[\begin{array}{ccc} 0 & 1 & 1 \end{array} \right]^T$$

Table 1: Evolution du nombres d'iterations en fonction de μ_k (lignes) et de λ_k (colonnes)

	0.1	0.3	0.5	0.7	0.9	1.2	1.4	1.6	1.8	2
0.1	41	41	41	41	41	41	50	50	50	50
0.3	39	39	38	38	38	38	38	38	37	37
0.5	37	37	37	37	37	37	37	36	36	36
0.7	37	36	36	36	36	36	36	36	35	35
0.9	36	36	36	36	36	36	36	35	35	35
1.2	35	35	35	35	35	35	35	34	34	34
1.4	35	35	35	35	35	34	34	34	34	34
1.6	35	35	34	34	34	34	34	34	33	33
1.8	34	34	34	34	34	34	34	33	33	33
2	34	34	34	34	34	33	33	33	33	33

On constate que le λ initial a peut d'impact sur l'evolution de l'algorithme. Son augmentation accelere peut le deroulement de celui-ci. On remarque egalement que le lambda en sortie est important dans tous les cas, ce qui suggere que la solution du probleme sans contrainte ne satisfait pas aux contraintes appliquees. Ceci est verifie en se basant sur les resolutions effectues precedement.
μ pour sa part a un impact significatif sur le nombre d'iterations realisees. Comme le montre le tableau ci-dessus, son augmentation tend a diminuer le nombre d'iterations realises, et permet egalement de rendre soluble des resolutions qui prenaient auparavant trop de temps, la partie haute du tableau le confirmant. On note egalement que le μ final a augmente a chaque fois, ce qui montre que les contraintes n'etaient pas respectees et qu'il a fallu penaliser plusieurs fois afin de les respecter.
De maniere plus general, l'algorithme est beaucoup plus lent que les precedents. En effet, on effectue a chaque iteration des algorithmes deja lourds du fait de la modification du facteur des contraintes.

Table 2: Evolution du nombre d'iterations en fonction de τ

τ	0.1	0.3	0.5	0.7	0.9	1.2	1.4	1.6	1.8	2
nombres d'iterations	50	50	50	50	50	50	41	29	24	20

2. On remarque que les valeurs grande de τ ameliore la converge de l'algorithme. On pourrait en effet penser qu'une valeur faible ameliorerait la convergence du fait d'un plus faible penalisation. En ayant choisi un critere d'arret sur la precision de la fonction, on s'attache plus a la justesse du resultat.
La valeur de τ la plus importante est celle qui fournie donc les meilleures resultats. Il faut neanmoins faire attention a ne pas trop l'augmente, celle-ci pouvant deteriorer la convergence sur d'autres fonctions.

3. Cette partie concernant l'adaptation de l'algorithme precedent au probleme d'inegalite n'a pas ete traite. Il faudrait modifier l'algorithme pour determiner le signe du gradient des contraintes. On incluerait egalement les contraintes d'egalites en les traduisant par 2 inegalites.

6.2 Tests

Fichiers testsLagrangien.m

On reprend les tests donnes par le sujet en notant f la fonction a minimiser, x le point de depart de la minimisation et c la contrainte.

- $f = f_1 \quad x_0 = x_{c11} \quad c = c_1 \quad$ 33 tours realises

$$x^* = \begin{bmatrix} 0.5 & 1.25 & 0.5 \end{bmatrix}^T, \quad f_1(x^*) = 2.25$$

- $f = f_1 \quad x_0 = x_{c12} \quad c = c_1 \quad$ 33 tours realises

$$x^* = \begin{bmatrix} 0.5 & 1.25 & 0.5 \end{bmatrix}^T, \quad f_1(x^*) = 2.25$$

- $f = f_2 \quad x_0 = x_{c21} \quad c = c_2 \quad$ 5 tours realises

$$x^* = \begin{bmatrix} 0.9072 & 0.8228 \end{bmatrix}^T, \quad f_1(x^*) = 0.0086$$

- $f = f_2 \quad x_0 = x_{c22} \quad c = c_2 \quad$ 5 tours realises

$$x^* = \begin{bmatrix} 0.9072 & 0.8228 \end{bmatrix}^T, \quad f_1(x^*) = 0.0086$$

Conclusion

Ce projet fut tres interessant car il apporte un aspect concret a des problemes etudies en cours. Pouvoir confronter les connaissances et acquis a une implantation a permis de mieux les structurer.
Le faire par l'intermediaire de seance de TP dedies est un vrai plus, bien que les explications fournies etaient parfois difficiles a decryptees.

7 Annexe

L'ensemble des fichiers a ete modifiee dans un but de clarte et de remise en forme. Les fichiers fournis precedemment etaient a usage personnel et ont ete rendus en l'etat tel que demande.