École Nationale Supérieure
d'Électrotechnique, d'Électronique, d'Informatique,
d'Hydraulique et des Télécommunications

Institut National Polytechnique

RAPPORT DU PROJET D'OPTIMISATION NUMERIQUE

Réalisé par :
Ismail
MOUSSAOUI

Sous la direction de :
M. Joseph GERGAUD

SOMMAIRE

1 Introduction

Dans le cadre du cours d'optimisation numérique dispensé aux étudiants de 2ème année de la filière informatique et mathématique appliquées de l'ENSEEIHT, nous avons été emmenés à programmer sous matlab différents algorithmes classiques d'optimisation. Ainsi nous avons eu à programmer des algorithmes d'optimisation sans contraintes à savoir l'algorithme de newton local et l'algorithme des régions de confiances. Nous avons aussi programmé un algorithme d'optimisation avec contraintes , celui du lagrangien augmenté. Ce rapport présente le travail réalisé, les tests effectués , les résultats de ceux ci ainsi que les intérpretations de certains résultats.

Première partie
Optimisation sans contraintes

2 Algorithme de Newton local

Cet algorithme approche la fonction à minimiser par son developpement de Taylor au second ordre et calcul le min exacte de cette approximation par résolution d'un système linéaire à solution unique.ce qui nous donne le point de départ de nouvelle itération

2.1 Sorties et paramétres du *newton_local.m*

Paramétres :

- f : la fonction f que l'on veut minimiser
- x0 : l'itére intial
- tol : la précision demandée
- max_iter : le nombre maximum d'itérations.
- g_f : la fonction gradiant de f
- h_f : la fonction hessienne de f

Sorties :

- x_sol : itére final
- f_sol : valeur de la fonction objectif en xsol
- nb_itr : nombre d'itérations
- flag : indicateur de deroulement de l'algorithme :
  - 0 : convergence
    $||\nabla f(x_k)|| \leq tol * (\nabla f(x_0) + \sqrt{\epsilon_{machine}})$
  - 1 : Distance entre les itérés trop faible pour poursuivre
    $||x_k - x_{k+1}|| \leq tol * (||x_k|| + \sqrt{\epsilon_{machine}})$
  - 2 : Distance entre les f(itéres) trop faibles pour poursuivre
    $||f(x_{k+1}) - f(x_k)|| \leq tol * (||f(x_k)|| + \sqrt{\epsilon_{machine}})$
  - 3 : nombre max d'itération atteint
  - -1 :Une erreur s'est produite

2.2 Résultats :

Résultat du script *test_newton_local.m* sur les exemples d'annexe A

Constantes : tol = $1e^{-3}$, max_iter =50 .

Fonction	Point initial	itére final xsol	valeur de f en xsol	nombre d'itérations	flag
f_1	x_{011}	$\begin{pmatrix}1\\1\\1\end{pmatrix}$	$1.9722e^{-31}$	1	0
f_1	x_{012}	$\begin{pmatrix}1\\1\\1\end{pmatrix}$	5.0191^{-29}	1	0
f_2	x_{021}	$\begin{pmatrix}1\\0.99999\end{pmatrix}$	$1.8527e^{-11}$	5	0
f_2	x_{022}	$\begin{pmatrix}1.0004\\1.0007\end{pmatrix}$	$1.3279e^{-07}$	3	0
f_2	x_{023}		$2.5e^{+19}$	4	1

2.3 Reponses aux questions

1. La fonction f1 est une fonction quadratique , elle est donc égale à son développement de Taylor à l'odre 2 . Et puisque Newton Local calcule le min exacte de cette développement de Taylor à l'odre 2 de la fonction donc en une itération on résout donc le systeme lineaire associe, avec pour solution le minimum de f_1.

2. Parmi les limites de l'algorithme de newton local, il y a la non convergence pour certains fonctions dont le développement de Taylor à l'odre 2 est trop loin de la fonction pour certains points .
Donc lorsque le premier itéré est trop loin de la solution ou si la matrice hessienne est singulière l'algorithme ne converge pas .
Nous avons donc essayé d'observer cette limite avec notre implémentation en donnant des valeur initiales assez éloignées de la solution. Nous avons ainsi pu observer qu'avec certaines valeurs initiales, l'algorithme de newton ne converge pas. En effet avec la fonction f_2 par exemple, ne converge pas pour le point initial x_{023}

3 Régions de confiance - Partie 1

L'algotithme de Régions de confiance essaye d'éviter Le principal défaut de l'algorithme de Newton qui est sa non-convergence. L'idée de cette méthode est d'approcher f par une fonction modèle quadratique dans une région de confiance et minimiser cette approximation dans cette région. et mettre à jour dans chaque itération le rayon de cette région pour s'assurer que le modéle quadratique est prés de f dans cette région

3.1 Le pas de Cauchy isolé

l'algorithme du pas de cauchy cherche a résoudre le sous probléme de région de confiance .

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

elle cherche la solution selon moins le gradient de la fonction à minimiser et à l'intérieur de la région de confiance courante c'est-à-dire à l'intérieur du boule de rayon delta .

3.1.1 Sorties et paramétres du *pas_cauchy.m*

En Entrée :

- g : la fonction gradiant de f en x_k
- H : la fonction hessienne de f en x_k
- delta : rayon de la région de confiance à l'itération k

En sortie :

— s : minimum trouvé
— d : la décroissance obtenue avec cette algorithme

3.1.2 Résultats :

Résultat du script *test_pas_cauchy.m* sur les exemples de l'annexe B

Constantes : delta=1 .

Gradient	Hessienne	Pas de Cauchy
$\begin{pmatrix}0\\0\end{pmatrix}$	$\begin{pmatrix}7 & 0\\0 & 2\end{pmatrix}$	$\begin{pmatrix}0\\0\end{pmatrix}$
$\begin{pmatrix}6\\2\end{pmatrix}$	$\begin{pmatrix}7 & 0\\0 & 2\end{pmatrix}$	$\begin{pmatrix}-0.92308\\-0.30769\end{pmatrix}$
$\begin{pmatrix}-2\\1\end{pmatrix}$	$\begin{pmatrix}-2 & 0\\0 & 10\end{pmatrix}$	$\begin{pmatrix}0.89443\\-0.44721\end{pmatrix}$

3.2 Régions de confiance avec pas de Cauchy

3.2.1 Sorties et paramétres du *region_confiance.m*

Entrée :

— f : la fonction objectif
— g_f : la fonction gradiant de f
— h_f : la fonction hessienne de f
— tol : la précision demandée
— max_iter : le nombre maximum d'itérations.
— delta_max : rayon de région de confiance max
— delta0 : rayon de région de confiance initial
— gamma1,gamma2,eta1,eta2 : les constantes des régions de confiance
— type_algo : le type d'algorithme pou calculer le pas 1 : par le pas de cauchy
2 : par More-Sorensen

Sorties :

— x_sol : itére final
— ng_sol : norme du gradient en xsol
— nb_iter : nombre d'itérations
— f_sol : valeur de la fonction objectif en x_sol
— d : la décroisance de fonction objectif
— flag : indicateur de deroulement de l'algorithme :

— 0 : convergence
— 1 : Distance entre les itérés trop faible pour poursuivre
— 2 : Distance entre les f(itéres) trop faibles pour poursuivre
— 3 : nombre max d'itération atteint
— -1 :Une erreur s'est produite

3.2.2 Résultats :

Résultat du script *test_region_confiance.m* sur les exemples de l'annexe A

Constantes : tol = $1e^{-3}$, max_iter =50 $\delta_0 = 2$ pour f2 et $\delta_0 = 1$ pour f1 ; $\gamma_1 = 0.50$; $\gamma_2 = 2$; $\eta_1 = 0.25$; $\eta_2 = 0.75$;.

f	x0	itére final xsol	valeur de f en xsol	norme du gradient en xsol	la décroissance obtenue	nombre d'itérations	flag
f_1	x_{011}	$\begin{pmatrix} 1.003 \\ 1.0004 \\ 0.99781 \end{pmatrix}$	$1.6343e^{-05}$	0.011114	9	15	0
f_1	x_{012}	$\begin{pmatrix} 1.0093 \\ 0.99927 \\ 0.98924 \end{pmatrix}$	0.0002108	0.03216	197.7198	16	0
f_2	x_{021}	$\begin{pmatrix} -0.92738 \\ 0.86447 \end{pmatrix}$	3.7168	2.3813	20.4832	50	3
f_2	x_{022}	$\begin{pmatrix} 1.1601 \\ 0.91327 \end{pmatrix}$	18.7338	218.8509	1000062.26	7	0
f_2	x_{023}	$\begin{pmatrix} 0.8358 \\ 0.69793 \end{pmatrix}$	0.027002	0.17215	0.9755	50	3

-

3.2.3 Reponses aux questions

1. La fonction f_1 est égale à son développement de Taylor à l'ordre 2. Donc Sur f_1, l'algorithme de Newton converge en une unique itération car il minimise directement f_1,Dans le cas du pas de Cauchy il y a aussi minimisation de la même quadratique mais selon le vecteur moins gradient de la fonction , Pourtant cette direction n'est pas toujours proche du minimum local de la fonction . on ne trouve pas donc pas immédiatement le minimum local de f_1 mais aprés 15 itérations pour x_{011} et 16 itérations pour x_{012}.
Cependant l'algorithme des régions de confiance nous garanti la convergence même

dans les cas ou la hessienne est singulière ce qui n'est pas le cas avec l'algorithme de newton.

2. On peut jouer sur différents paramétres pour modifier optimiser l'éxecution de l'algorithme des régions de confiances :

- Δ_{max} : rayon de confiance maximal
- γ_1 et γ_2 : facteurs d'agrandissement et de réduction de la région de confiance
- η_1 et η_2 : critéres d'agrandissement et de réduction de la région de confiance

4 Régions de confiance - Partie 2

4.1 Newton pour les équations non linéaires

L'algorithme suivant permet de résoudre un sous problème nécessaire à la résolution de l'algorithme de More-Sorensen.

elle permet de résoudre des équations de la forme $\phi(\lambda) = 0$, où la fonction ϕ sera une fonction non linéaire de la variable réelle. Cela sera réalisé (de façon approchée) par l'utilisation de la méthode de Newton 2 , combinée avec une technique de dichotomie pour assurer sa convergence

4.1.1 Sorties et paramétres du *newton_eq_non_lin.m*

Entrée :

- fi : une fonction de variable réelle pour laquelle on cherche un zéro
- d_fi : la fonction dérivée de fi
- tol : la précision demandée
- max_iter : le nombre maximum d'itérations.
- l_min , l_max : des valeurs réelles tel que fi(l_min)*fi(l_max)¡=0

Sortie :

- l : la valeur approché de zéro de fi
- fl : valeur de f en l
- nb_itr : nombre d'itération

4.1.2 Résultats :

Résultat du script *test_region_confiance.m* sur les exemples de l'annexe C

Constantes : tol=1e-6 ; max_itr=100 ; l_min=100 ; l_max=0 ;

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

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

- $b = (2, 14), \quad a = (2, 6) \quad \delta = 0.5$

$$\lambda = 3.4965$$

- $b = (-38, 20), \quad a = (2, 20) \quad \delta = 0.2$

$$\lambda = 82.6112$$

— $b = (-38, 20), \quad a = (2, 20) \quad \delta = 0.7$

$$\lambda = 8.7083$$

On s'interesse d'abord a la fonction ϕ_2 definie comme suit

$$\phi_2(\lambda) = \frac{1}{||s(\lambda)||^2} - \frac{1}{\delta^2}, \quad ||s(\lambda)||^2 = \sum \frac{a_i^2}{(b_i + \lambda)^2}$$

— $b = (2, 14), \quad a = (2, 6) \quad \delta = 0.5$

$$\lambda = 3.4965$$

— $b = (-38, 20), \quad a = (2, 20) \quad \delta = 0.2$

$$\lambda = 82.6112$$

— $b = (-38, 20), \quad a = (2, 20) \quad \delta = 0.7$

$$\lambda = 8.7083$$

4.1.3 Réponses aux questions

1. Le critère d'arrêt pertinent est donc que la solution trouvée a un ϵ prés. On s'arrête donc lorsque λ courant vérifie à ϵ près l'équation ($|\phi(\lambda)| \leq \epsilon$).

2. Pour des fonctions quelconque c'est difficile de chercher le couple (λ_{min} , λ_{max}) qui vérifie la condition souhaitée.
Dans notre cas les fonctions ont une forme bien définie que l'on peut utiliser .
Si par exemple, on a λ_{min}, on peut chercher λ_{max} en se déplaçant dans l'intervalle $]\lambda_{min}$, $+\infty[$ de 1 en 1 jusqu'à tomber sur une valeur rendant f négative ou en multipluant λ_{min} par une valeur positive à chaque itération .

4.2 Algorithme de Moré-Sorensen isolé

Le pas de Cauchy converge lentement vers la solution car il se restreint aux déplacements selon moins le gradient

Le pas de More-Sorensen recherche à chaque itération une solution de la quadratique approchant la fonction dans toute la boule (la région de confiance)

4.2.1 Sorties et paramétres du *more_sorensen.m*

En Entrée :

- g : la fonction gradiant de f en x_k
- H : la fonction hessienne de f en x_k
- delta : rayon de la région de confiance à l'itération k

En sortie :

- s : minimum trouvé (pas de Moré Sorensen)
- d : la décroissance obtenue avec cette algorithme
- l : valeur du multiplicateur de Lagrange obtenue
- flag : indicateur de deroulement de l'algorithme :
  - 1 : solution intérieure
  - 2 : solution sur la frontière et q1 * g' != 0
  - 3 : solution sur la frontière et q1 * g' = 0 et $norm$(s_(-λ1)) > $delta$
  - 4 : solution sur la frontière cas difficile ($q1 * g' = 0 et norm$(s_(-λ1)) < $delta$)
  - -1 : Une erreur s'est produite

4.2.2 Résultats :

Résultat du script *test_more_sorense.m*

Constantes : tol = 1e-6 ; delta = 1 ;

Quadratique	Pas de Moré Sorensen	λ	décroissance	flag
1	$\begin{pmatrix} 0 \\ 0 \end{pmatrix}$	0	0	1
2	$\begin{pmatrix} -0.7485 \\ -0.66313 \end{pmatrix}$	1.016	3.4166	2
3	$\begin{pmatrix} 0.99745 \\ -0.071402 \end{pmatrix}$	4.0051	3.0357	2
4	$\begin{pmatrix} 1 \\ 0 \end{pmatrix}$	2	1	4
5	$\begin{pmatrix} 0.51495 \\ -0.85722 \end{pmatrix}$	2.1041	1.8229	2
6	$\begin{pmatrix} -0.10526 \\ 0.99444 \end{pmatrix}$	15	7.6053	4

4.3 Régions de confiance avec Moré-Sorensen

4.3.1 Sorties et paramétres du *region_confiance.m*

Ils sont indiqués dans la partie Régions de confiance avec Pas de cauchy

4.3.2 Résultats :

Résultat du script *test_region_confiance.m* sur les exemples de l'annexe A

Constantes : tol = $1e^{-3}$, max_iter =50 $\delta_0 = 2$ pour f2 et $\delta_0 = 1$ pour f1 ; $\gamma_1 = 0.50$; $\gamma_2 = 2$; $\eta_1 = 0.25$; $\eta_2 = 0.75$;.

f	x0	itére final xsol	norme du gradient en xsol	valeur de f en xsol	nombre d'itérations	la décroissance obtenue	flag
f_1	x_{011}	$\begin{pmatrix}1\\1\\1\end{pmatrix}$	$2.5121e^{-15}$	$1.9722e^{-31}$	1	9	0
f_1	x_{012}	$\begin{pmatrix}1\\1\\1\end{pmatrix}$	$2.5121e^{-15}$	$1.9722e^{-31}$	3	197.72	0
f_2	x_{021}	$\begin{pmatrix}0.9985\\0.99696\end{pmatrix}$	0.019521	$2.4819e^{-06}$	34	24.2	0
f_2	x_{022}	$\begin{pmatrix}9.99955\\99.991\end{pmatrix}$	17.9999	80.9919	1	10^6	0
f_2	x_{023}	$\begin{pmatrix}0.99999\\0.99999\end{pmatrix}$	0.00015319	$6.1504e^{-11}$	34	1.0025	0

4.3.3 Reponses aux questions

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

2. L'algorithme RC trouve la solution en moins d'itération avec l'algorithme de Moré-Sorensen, et donc la décroissance avec celui-ci est plus rapide que celle obtenue avec le pas de Cauchy. L'algorithme de Moré à l'avantage d'être une

méthode de résolution éxacte. Mais,une itération de ce dernier est très couteuse car elle fait le calcul des vecteurs propres et fait appel à l'algorithme de Newton non Linéaire.

Deuxième partie
Optimisation avec contraintes

5 Lagrangien Augmenté

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

5.1 Sorties et paramétres du *lagrangien_augmente.m*

En Entrée :

- f : la fonction objectif
- g_f : la fonction gradiant de f
- h_f : la fonction hessienne de f
- c : la fonction condition (c(x)=0)
- g_c : la fonction gradiant de c
- h_c : la fonction hessienne de c
- mu0 : valeur de muk intial
- x0 : itére initial
- l0 : multiplicateur de lagrange
- tol : la précision demandée
- max_iter : le nombre maximum d'itérations.
- to , eta0_ch , alpha , beta , eps0 , eta0 : des constantes d'algorithme
- type_algo : type d'algorithme pour la résolution de la suite de problémes sans contraintes
  - 1 newton local
  - 2 région de confinace (pas de cauchy)
  - 3 région de confinace (Moré Sorensen)

En sortie :

— x_sol : itere final
— l_k : multiplicateur de lagrange final
— mu_k : mu final
— f_sol : valeur de la fonction objectif en x_sol
— flag : indicateur de deroulement de l'algorithme :
— 0 : convergence
— 1 : nombre max d'itération atteint
— -1 : une erreur est produit

5.2 Résultats :

Résultat du script *test_lagrangien.m* sur les exemples de l'annexe E avec résolution de la suite de problémes sans contraintes avec Région de confiance (Moré Sorensen) .
Il suffit de changer le paramétre type_algo pour changer l'algorithme de résolution de la suite de problémes sans contraintes

Constantes : max_iter=100 ; tol=1e-3 ; to = 1.5 ; mu0=10 ;

f	x0	itére final xsol	valeur de f en xsol	nombre d'itérations	λ_k	μ_k	flag
f_1	x_{c11}	$\begin{pmatrix}0.50013\\1.2499\\0.50013\end{pmatrix}$	2.2489	8	4.4794	75.9375	0
f_1	x_{c12}	$\begin{pmatrix}0.50013\\1.2499\\0.50013\end{pmatrix}$	2.2489	8	4.4794	75.9375	0
f_2	x_{c21}	$\begin{pmatrix}0.90724\\0.82276\end{pmatrix}$	0.0086152	3	0.038463	15	0
f_2	x_{c22}	$\begin{pmatrix}0.90718\\0.8226\end{pmatrix}$	0.0086266	4	0.042926	15	0

5.3 Réponses aux questions

1. 0n remarque que λ_k finale , n'est pas nul dans les 4 cas . on peut justufier ça par le fait que la solution du problème sans contrainte n'est pas dans le domaine défini par la contrainte. Il est cependant plus petit dans le cas de la fonction f2 car la solution est proche de satisfaire la contrainte.
μ_k finale , a une valeur élevée dans la pluspart des cas ce qui montre que l'algo-

rithme est passé plusieurs fois par la phase de pénalisation car les contraintes n'étaient pas assez respectées au cours des itérations.
0n remarque aussi que si on donne une valeur initiale λ_0 proche de la vraie valeur du multiplicateur de Lagrange, le résultat est trouvé rapidement.

2. Il faut choisir une valeur de τ supérieur strictement à 1 pour que μ_k tende vers l'infini.
On remarque que si la valeur de τ est inférieur à 1 l'algorithme conduit à des résultats erroné.
On remarque aussi que la précision de la solution et la vitesse de convergence dépend de τ . (La valeur de τ =1.5 est la plus efficace , en terme de nombre d'itérations).

6 Conclusion

Ce projet est très intéressant car il permet de confronter les connaissances et acquis à une implantation concrete à des problèmes étudiés en cours. pour une meilleur compréhension des algorithmes étudiés en cours .