Compte rendu optimisation numérique

Dans ce projet, nous avons programmé trois méthodes de résolution de problème sans contraintes : méthode de descente, de Newton et de quasi Newton (ou bfgs). Dans chacun de ces méthodes, on utilise des algorithmes de calcul du pas :

- Le pas constant : on utilise un pas qui vaut toujours la même valeur, généralement 1 ou -d'd/(d*Hess(x)*d)
- Backtracking prenant en paramètre un scalaire rho. Le pas résultat satisfait la première condition Wolfe
- La bissection : le pas satisfait les deux conditions faibles de Wolfe
- L'interpolation : le pas satisfait la première condition faible de Wolfe
- L'approche : le pas satisfait lesdeux condition fortes de Wolfe

Le fichier test permet de calculer le minimum de f1 ou f2, de calculer la valeur de la fonction en ce point, de calculer le temps d'exécution de l'algorithme et le nombre d'itérations pour les algorithmes de Newton, de descente (avec les différents pas) et de quasi Newton (avec les différents pas). A la fin du test s'affiche le nombre d'évaluation de f1, f2, leur gradient et hessienne.

1 – Résultat des tests pour l'optimisation sans contrainte

Comme la fonction f1 est très simple, le test montre que le minimum est 1;1 et que la valeur de f en ce point est 0, ceci avec un précision moyenne en 10^{-8} , allant jusqu'à 10^{-32}, avec un nombre d'itération ne dépassant jamais 10, et un nombre d'évaluation allant jusqu'à 250 au maximum.

Pour la fonction f2 :

- [10;10] : Les résultats obtenus pour la méthode de descente sont tous mauvais. Pour la méthode de quasi Newton, ce point semble problématique pour les algorithmes de calcul du pas de backtracking et de pas constant (égal à 1). Pour le reste, le résultat est bon avec un moyenne de 40 itérations et une précision en 10^{-6}
- [0;0] : solution correcte avec un précision moyenne en 10^{-4} . Le nombre d'itération dans les algorithmes est faible (environ 40) , mais le nombre d'évaluation des fonctions est important (a peu prés 2500).
- [3;1], [1;3], [0;1], [1;0] : les résultats sont très bons, sauf pour la descente par interpolation.

De manière générale, il semble que plus l'on s'éloigne du minimum de f2, et plus la méthode de descente est mauvaise. La méthode de quasi Newton avec pas constant semble aussi avoir des faiblesses. Les meilleurs méthodes sont donc celle de Newton locale, de quasi Newton avec bissection, backtracking, interpolation et approche.

2 – Comparaison avec la fonction fminunc

Cette fonction utilise l'algorithme de quasi Newton pour calculer le minimum de la fonction.

- [10;10] est problématique pour cette fonction aussi : le résultat reste bon puisque l'on arrive au point [1,04;1,08] et la valeur de f est 0,0015 pour un gradient de 0,15.
- Aux autres points, les résultats sont comparables aux résultats obtenus précédemment, avec toutefois une meilleur précision pour les résultats précédents.

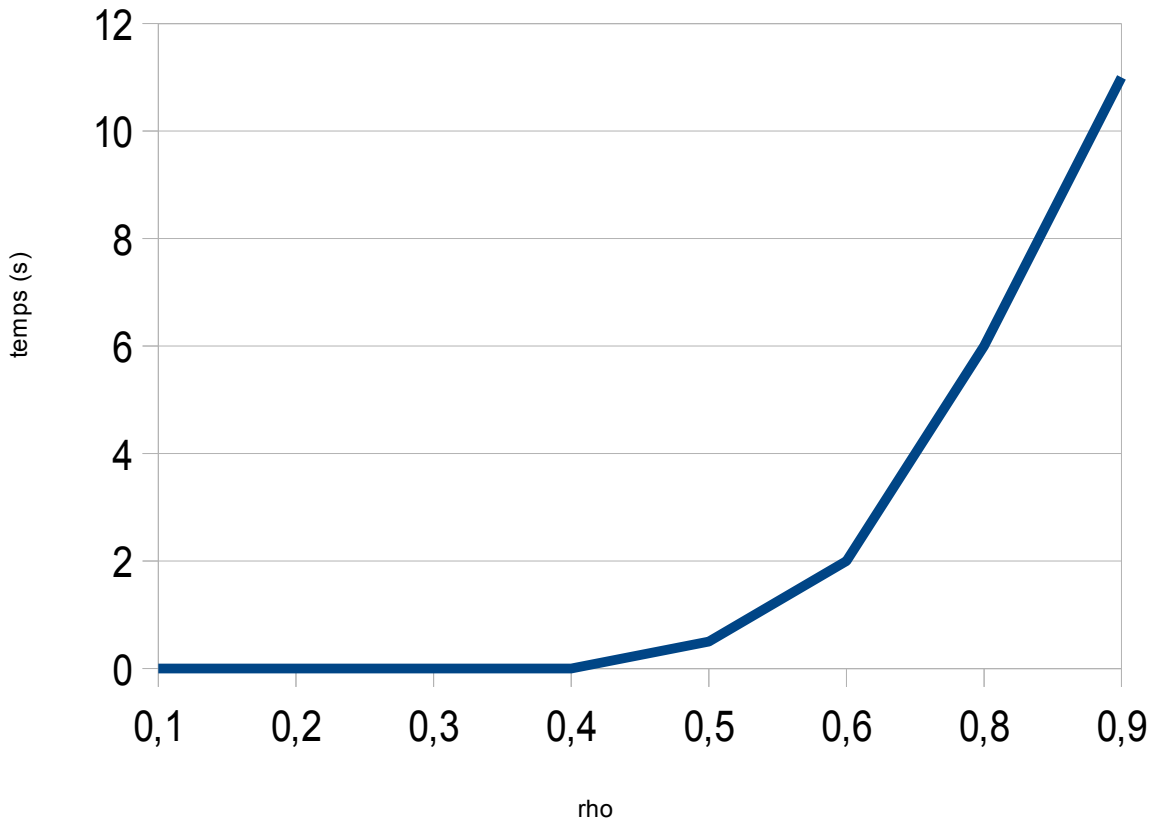
3 – Étude de l'influence du paramètre ρ dans le backtracking

On fait varier ρ dans [0;1], on lance le test, avec le point initial [0,5;0,5], une précision en 0,001, c1 qui vaut 0,4 et c2 0,8, et on compare les résultats dans le tableau suivant :

rho	f1(x_min)	x_min	f2(x_min)	x_min	temps
0,1	0,0000005	0.999;0.998	0,15	0.61;0.37	0,001
0,2	0,00000115		7,91E-010	0.99997;0.9999	0,001
0,3	0,00000114		6,55E-011		0,001
0,4	0,00000104		1,58E-008		0,001
0,5	0,00000111		2E-012		0,5
0,6	0,000001		1E-012		2
0,8	0,0000048		1E-011		6
0,9					11

Les résultats pour x_min sont ensuite toujours les mêmes.

Courbe du temps d'exécution en fonction de ρ :



On remarque les résultats obtenus sont mauvais pour ρ = 0,1, et que ensuite les résulats sont bons, mais le temps d'exécution augmente exponentiellement. On choisit donc ρ=0,3 pour le reste des tests.

4 – Influence du paramètre c1 et c2 sur les tests

Le paramètre c1 est utile dans la première condition de Wolfe qui assure une descente suffisante : plus c1 est petit, plus la descente sera petite (le pas calculé sera petit) et donc on aura plus d'itération. Voici différents tests suivant la valeur de c1 pour c2 fixe :

c1	temps execution moyen	tps execution maximal	nbIterMax	nfev	ngev	
	0,1	1,1185	5	5000	298000	373000
	0,3	0,4587	1,4	4760	139000	146000
	0,4	0,3926	1,28	4500	124000	125000
	0,45	0,3891	1,31	4500	119000	119000

On remarque bien que pour c1 petit, le nombre d'itération est plus important (nbIterMax, représentant le nombre d'itération maximal d'un des algorithmes) et le nombre d'évaluation des fonctions est aussi presque triplé entre c1=0,1 et c1=0,45. Il est donc judicieux de prendre c1=0,45 pour le reste. On ne dépasse pas 0,45 car au delà, l'algorithme de calcul du pas d'interpolation ne fonctionne plus.
Pour ce qui est de la valeur de c2, qui doit être entre c1 et 1, après plusieurs tests on s'aperçoit que si c2 est proche de c1 alors beaucoup d'algorithme boucle infiniment. On prend donc c2 suffisamment éloigné de c1 (pour c1=0,45, c>0,7 est correct).

5 – Test du problème avec contraintes

Le fichier test_prob_cont permet de tester le problème avec contrainte proposé dans l'énoncé en modifiant les paramètres de précision, c1, c2, le point initial, et les paramètres lambda et mu. On obtient par exemple pour une précision en 0,0 une solution xmin= 0,90;0,82 avec une contrainte presque nulle (10^{-16}).

On étudie ensuite l'influence du paramètre mu sur le problème avec contrainte : l'algorithme ne fonctionne plus si mu s'éloigne de 1. De même pour lambda.