Rapport d'optimisation numérique

Philippe Leleux
Groupe F

Partie 1 : Comparaison des différents calculs de pas avec Newton et quasi-Newton

Nous allons tester les algorithmes avec :

- valeur pour le paramètre de contraction de backtracking : ro = 0.5.
- valeur des paramètres pour les conditions de Wolfe : c1 = 0.4, c2 = 0.5.

Pour chacun des algorithmes de calcul du pas nous ne présenterons que les résultats pour f2, pour f1 le calcul converge toujours vers la valeur exacte très rapidement (presque tout le temps en 1 itération peu importe le point de départ).

I Algorithme de Newton

Les critères d'arrêt pour cet algorithme sont :

- | ||xk|| - ||xk+1|| | < Critere
- | ||(xk)|| - ||f(xk+1)|| | < Critere
- iter > nb_iter

avec Critere = 1e -10
nb_iter = 100000

1 Newton avec pas fixe

Les résultats obtenus avec cette méthode sont résumés dans le tableau suivant :

Pas Fixe – X0	termine	juste	iter	nfev	ngev	nhev	Durée (s)
[0, 0]	oui	oui	3	10	3	3	0
[0, 1]	oui	oui	6	19	6	6	0
[1, 0]	oui	oui	2	7	2	2	0
[1, 3]	oui	oui	2	7	2	2	0,01
[3, 1]	oui	oui	6	19	6	6	0
[10, 10]	oui	oui	6	19	6	6	0

On observe d'abord qu'avec cette méthode pour décider le pas, tous les calculs aboutissent à la bonne valeur avec un nombre d'itérations bas et un nombre de calculs demandés également bas. La durée de l'algorithme est quasi-nulle, en effet ce qui prend le plus de temps c'est le calcul de la hessienne et ici on ne la calcule que très peu puisque le nombre d'itération est bas et que l'on en a pas besoin dans le calcul du pas.

En première hypothèse on peut se dire que ce sont les variations de la première coordonnées du point de départ par rapport à la solution qui influent réellement sur la convergence. Plus cette variable est éloignée de la solution, plus la convergence est longue et demande des calculs mais jusqu'à un certain seuil. Pour illustration, avec [1, 3] et [1, 0] on a une convergence rapide tandis que pour [0, 1] et [3, 1] la convergence est plus longue.

2 Newton avec backtracking

Résultats obtenus avec backtracking :

Backtracking – X0	termine	juste	iter	nfev	ngev	nhev	Durée (s)
[0, 0]	oui	oui	16	95	39	16	0
[0, 1]	oui	[-0,08467, 0,03609]	6	141	67	6	0,01
[1, 0]	oui	oui	2	11	4	2	0
[1, 3]	oui	oui	2	11	4	2	0
[3, 1]	oui	oui	24	137	56	24	0,01
[10, 10]	oui	oui	50	287	118	50	0,02

Pour cet algorithme, il y a un calcul sur un point qui aboutit à un mauvais résultat. Ce point [0, 1] est l'un des points qui posera le plus problème dans la suite. Sinon cette algorithme converge relativement rapidement (moins que pour le pas fixe mais rapidement par rapport à la suite) vers la bonne solution.

Un calcul de la hessienne en plus est demandé à chaque itération ainsi qu'un grand nombre d'évaluations de f et son gradient c'est ce qui explique que l'algorithme est plus long. On remarque également que le nombre d'itération et d'évaluation de fonctions n'est pas du tout stable au contraire de la méthode précédente, par exemple avec [3, 1] et [1, 3] qui équidistant de la solution demande respectivement 24 et 2 itérations, 137 calcul de f contre 11 pour donc un ratio d'aà peu près 10 contre 3 pour la méthode précédente.

On peut faire la même observation que précédemment mais cette fois-ci nous n'observons pas de seuil par rapport à la première coordonné et contrairement à précédemment si on éloigne la seconde de la solution, les itérations augmentent légèrement.

3 Newton avec bisection

Résultats avec bisection :

Bisection – X0	termine	juste	iter	nfev	ngev	nhev	Durée (s)
[0, 0]	oui	oui	16	95	73	16	0,01
[0, 1]	oui	oui	17	100	77	17	0
[1, 0]	oui	oui	2	11	8	2	0,01
[1, 3]	oui	oui	2	11	8	2	0
[3, 1]	oui	oui	22	125	95	22	0,01
[10, 10]	oui	oui	50	291	222	50	0,02

Ces résultats ne changent pas trop par rapport à la méthode avec backtracking, seulement maintenant le calcul avec le point [0, 1] converge cette fois-ci vers un résultat et aussi rapidement qu'avec [0, 0].

Ici la dépendance de la convergence par rapport à la première variable est encore plus flagrante, pour celle-ci donnée, les résultats (itérations, etc...) sont quasi-identiques.

4 Newton avec Interpolation

Résultats avec interpolation :

Interpolation – X0	termine	juste	iter	nfev	ngev	nhev	Durée (s)
[0, 0]	oui	oui	1964	9823	3928	1964	0,000001
[0, 1]	oui	oui	1957	9788	3914	1957	0,62
[1, 0]	oui	oui	2	11	4	2	0
[1, 3]	oui	oui	2	11	4	2	0
[3, 1]	oui	oui	7530	37653	15060	7530	2,52
[10, 10]	oui	[1,000075, 1,00015]	100000	500004	200000	100000	31,2

Avec cette méthode, le nombre d'itérations et d'évaluations des fonctions explosent, il n'y a que deux points pour lesquels la convergence est rapide (et est même resté constant pour toutes les méthodes). Le temps d'exécution se retrouve bien évidemment largement augmenté lui aussi puisque l'on effectue énormément de calculs de la hessienne.

Pour le calcul avec pour point de départ [10, 10], le critère d'arrêt est le nombre d'itérations (100000) cependant en essayant avec un nombre d'itérations plus grand (x10) le résultat ne devient pas pour autant meilleur. Cet algorithme est très gourmant en calcul et très long.

5 Newton avec Approche et Finition

Résultats pour cette méthode :

approche et finition – X0	termine	juste	iter	nfev	ngev	nhev	Durée (s)
[0, 0]	oui	oui	16	131	77	16	0,01
[0, 1]	oui	oui	17	137	80	17	0,01
[1, 0]	oui	oui	2	13	6	2	0
[1, 3]	oui	oui	2	13	6	2	0
[3, 1]	non						
[10, 10]	non						

Avec cette méthode, il aut réellement être près de la condition pour obtenir le bon résultat (au moins pour la première coordonnée). Cependant si c'est le cas, Les résultats sont comparables avec ceux de la bisection (ce qui est normal puisque la méthode est utilisée dans l'algorithme).

On a donc une méthode qui converge rapidement et avec peu de calculs mais très sensible aux variations du point de départ.

II Algorithme de quasi-Newton (BFGS)

Pour cet algorithme, les critères d'arrêt sont :

- | ||graf|| | < Critere
- iter > nb_iter

1 BFGS avec pas fixe

Résultats avec pas fixe :

Pas Fixe – X0	termine	juste	iter	nfev	ngev	nhev	Durée (s)
[0, 0]	oui	oui	50	101	201	0	0,02
[0, 1]	oui	oui	38	77	153	0	0,01
[1, 0]	oui	oui	21	43	85	0	0
[1, 3]	oui	oui	31	63	125	0	0,01
[3, 1]	oui	oui	171	343	685	0	0,04
[10, 10]	non						

On remarque dès le départ que cet algorithme est plus long que l'algorithme de Newton, au niveau du temps d'une part mais surtout au niveau du nombre d'évaluations de la hessienne. Le calcul de la hessienne est très gourmand en temps et ressource, le fait qu'on ne le calcule plus est une bonne chose, cependant ici le temps de calcul a augmenté et on a même aucune convergence pour le point de départ [10, 10].

On remarque ici que le nombre d'itérations est bien moins dépendant de la première coordonnée du point de départ que précédemment.

2 BFGS avec backtracking

Résultats avec backtracking :

Backtracking – X0	termine	juste	iter	nfev	ngev	nhev	Durée (s)
[0, 0]	oui	oui	24	145	145	0	0,01
[0, 1]	oui	oui	32	379	286	0	0,02
[1, 0]	oui	oui	22	139	136	0	0,01
[1, 3]	oui	oui	37	229	226	0	0,02
[3, 1]	oui	[NaN, NaN]	10	185	123	0	0,01
[10, 10]	oui	oui	85	573	542	0	0,04

Ici également on remarque que le temps augmente et surtout les calculs passant pour nfev de entre 11 et 287 à entre 139 et 573 tandis que pour ngev on passe de entre 4 et 118 à entre 123 et 542 mais toujours aucun calcul de la hessienne, particularité de cette méthode.

Pour le point [3, 1], une erreur de calcul se produit pendant l'exécution (probablement une division par 0) et le résultat est donc faux mais je n'ai pas pu déterminé quel en était la raison.

3 BFGS avec bisection

Résultats avec cette méthode :

Bisection – X0	termine	juste	iter	nfev	ngev	nhev	Durée (s)
[0, 0]	oui	oui	21	125	174	0	0,01
[0, 1]	oui	oui	20	137	185	0	0,01
[1, 0]	oui	oui	15	107	141	0	0,01
[1, 3]	oui	oui	20	129	175	0	0,02
[3, 1]	oui	oui	26	189	253	0	0,01
[10, 10]	oui	oui	47	317	442	0	0,03

Sur toutes les méthodes de calcul de pas avec BFGS, c'est celle-là qui paraît être la plus satisfaisante. Avec cette méthode, peu importe le point de départ, elle converge rapidement (quelques centièmes c'est plutôt rapide et nous verrons à quel poitn dans les prochaines méthodes) et le nombre de calculs est le moins important, si ce n'est avec le pas fixe.

4 BFGS avec Interpolation et Approche-Finition

Résultats avec ces méthodes :

Interpolation – X0	termine	juste	iter	nfev	ngev	nhev	Durée (s)
[0, 0]	oui	oui	18782	75131	93911	0	6,88
[0, 1]	non						
[1, 0]	oui	[0,99985, 0,003957]	100000	400003	500001	0	36,3
[1, 3]	oui	[1,000017, 2,99806]	100000	400003	500001	0	35,57
[3, 1]	oui	[2,99999, 1,000045]	100000	400003	500001	0	35,62
[10, 10]	oui	oui	100000	400003	500001	0	35,63
approche et finition – X0	termine	juste	iter	nfev	ngev	nhev	Durée (s)
[0, 0]	oui	oui	18782	75131	93911	0	6,7
[0, 1]	non						
[1, 0]	oui	[0,99985, 0,003957]	100000	400003	500001	0	35,56
[1, 3]	oui	[1,000017, 2,99806]	100000	400003	500001	0	35,6
[3, 1]	non						
[10, 10]	non						

Avec ces méthodes, on arrive aux limites, il ne reste que quelques points où elles donnent un résultat juste mais pas forcément ceux auxquels on pourrait s'attendre. Le seul point où on a un résultat juste avec les deux méthodes est [0, 0] alors qu'il est plus éloigné de la solution que [1, 0] ou encore [0, 1], mais ce dernier est un cas spécial avec BFGS.

En plus de ne pas donner des résultats justes, les deux méthodes atteignent nb_iter, nombre d'itération maximum. Telles quelles, ces méthodes ne sont pas applicables pour la résolution de ce genre de problèmes d'optimisation.

Partie 2 : Paramètres ro, c1 et c2

I Influence du paramètre de contraction pour le backtracking

Nous allons tester l'influence de ce paramètre en lançant l'algorithme de Newton avec calcul du pas par backtracking et avec pour point de départ [10, 10] et en utilisant pour ro les valeurs : 0,001, 0,01, 0,5, 0,9, 0,95. Voici les résultats de ces tests :

ro	juste	iter	nfev	ngev	nhev	durée (s)
1,00E-003	oui	6617	46306	19844	6617	2,83
1,00E-001	oui	167	1144	488	167	0,07
0,5	oui	50	287	118	50	0,01
0,9	oui	245	8812	4283	245	0,47
0,95	oui	439	41442	20501	439	2,21

On observe clairement une courbe se dessiner qui trouve son minimum vers 0,5. Le paramètre de contraction joue à tous les niveau, des itérations à la rapidité de convergence et plus on s'approche des bornes 0 et 1, plus la vitesse diminue et le nombre de calculs nécessaires augmente.

Il faudrait faire une calibration pour obtenir le minimum réel pour l'algorithme de Newton mais d'après mes tests, il est trouvé avec un ro proche de 0,5.

Le choix de ce ro est très important au vu des résultats avec 0,001 ou 0,95 par exemple.

Dans les tests des deux algorithmes, le ro choisi était 0,5 pour la raison que cette valeur donne de très bon résultats.

II Influence des paramètres c1 et c2 pour les conditions de Wolfe

Nous allons tester l'influence de ces paramètres dans les même condition que précédemment mais cette fois-ci en calculant le pas avec la méthode bisection qui paraît la plus fiable.

c1 / c2	juste	iter	nfev	ngev	nhev	durée (s)
0,1 / 0,2	non	13	212	143	13	0,02
0,1 / 0,9	oui	48	279	211	48	0,02
0,5 / 0,9	oui	50	287	218	50	0,02
0,8 / 0,9	oui	195	1754	1169	195	0,12
0,000001 / 0,000002	non	8	155	99	8	0,01
0,4 / 0,5	oui	50	291	222	50	0,02
0,98 / 0,99	oui	1242	18631	11178	1242	1,14

L'influence des paramètres c1 et c2 est évidente une fois encore, sauf sur le temps d'exécution où il faut s'approcher de la borne 1 avec les deux paramètres pour voir une vértable influence.Autant en s'approchant de la borne 1, la convergence devient longue autant si on s'approche de 0, c'est le résultat de l'algorithme qui devient faux.

On observe également que plus l'écart entre les deux variables est important, plus la convergence est longue et le nombre de calculs important.

Je n'ai malheureusement pas eu le temps de finir de coder la méthode du lagrangien augmenté.