CaMUS **6**, 81 – 98

Analyse fine d'algorithmes de type ARC-TR au cas unidimensionnel

Samuel Goyette

Reçu le 2017-09-21 et accepté le 2018-01-09

Résumé Dans ce document, nous présenterons les algorithmes d'optimisation unidimensionnelle de région de confiance et de régularisation avec adaptation cubique. Puis, nous étudierons le calcul de minimiseurs globaux pour chacun d'entre eux. Nous verrons également des résultats numériques pour les algorithmes de régularisation cubique.

1 Introduction

Le problème d'optimisation que l'on veut résoudre est le suivant

$$\underset{t}{\text{minimiser}}\ \phi(t)$$

où $\phi : \mathbb{R} \to \mathbb{R}$ est une fonction continue deux fois différentiable. S'il existe un point t^* qui résout ce problème, il doit satisfaire aux conditions nécessaires d'optimalité :

$$\phi'(t^*) = 0 \tag{1}$$

$$\phi''(t^*) \geq 0. \tag{2}$$

Ainsi, pour résoudre ce problème, on peut utiliser une méthode analytique : on dérive la fonction, on trouve les points qui annulent la dérivée (appelés des points stationnaires) et, parmi ces points, on regarde les points dont l'évaluation de la dérivée seconde est positive. Cependant, certaines fonctions sont compliquées et le calcul n'est pas trivial. Si on prend les polynômes de degré un ou deux, il existe des formules analytiques connues pour trouver les racines de tels polynômes. Il en existe aussi pour ceux de degrés trois et quatre, mais elles sont complexes et peuvent mener à des erreurs de précision numérique. Pour les polynômes d'ordre supérieur ou égal à cinq, il n'existe pas de formule analytique permettant de trouver les racines. C'est le cas de nombreuses fonctions.

J'aimerais remercier Jean-Pierre Dussault qui m'a supervisé dans toutes mes activités de recherche au cours de mon baccalauréat.

On utilise donc des algorithmes d'optimisation qui nous permettent de trouver les racines de fonctions.

Les algorithmes de *région de confiance* (appelés TR pour *Trust-Regions*) et les algorithmes de *régularisation avec adaptation cubique* (appelés ARC pour *Adaptative Regularization using Cubics algorithms*) ont pour idée générale d'approximer une fonction "complexe" par une fonction plus "simple", que l'on appelle modèle, sur un certain intervalle. Ces fonctions sont dites "simples", car on est facilement capable de calculer leurs points stationnaires (ainsi que leurs minima globaux), ce qui permet de minimiser la fonction originale rapidement.

Dans ce texte, nous allons étudier les algorithmes de région de confiance et les algorithmes de régularisation cubique adaptative dans les sections 2 et 3. Dans ces deux sections, nous observerons le fonctionnement traditionnel des algorithmes puis certaines modifications qui permettent de les simplifier. Dans la section 4, nous observerons les résultats numériques associés aux deux méthodes.

2 Algorithme région de confiance (TR)

2.1 Présentation de l'algorithme

Les algorithmes TR visent à approximer notre fonction initiale ϕ par une fonction quadratique q_t. On veut minimiser cette approximation dans un certain intervalle ($[-\Delta, \Delta]$ où $\Delta > 0$) que l'on appelle région de confiance. Supposons que l'on a ϕ deux fois différentiable (remarquons que certaines variantes n'utilisent qu'une approximation de la dérivée seconde, ce qui permet d'alléger cette hypothèse). On considère l'approximation suivante de ϕ dans un voisinage de t :

$$q_t(d) = \phi(t) + \phi'(t)d + \frac{1}{2}Hd^2 \approx \phi(t+d) \text{ pour } d \in [-\Delta,\Delta] \tag{3}$$

où H est soit $\phi''(t)$ soit une approximation de $\phi''(t)$.

L'idée est donc de minimiser q_t dans l'intervalle de confiance, afin de s'approcher du minimum de ϕ. Il s'agit d'un procédé itératif, c'est-à-dire qu'après s'être approché du minimum de ϕ, on ajuste q_t pour avoir la meilleure approximation possible de ϕ. On répète cette procédure jusqu'à l'obtention d'un minimum local de ϕ (ou un point qui s'en rapproche suffisamment). Chaque itération consiste à trouver d^* qui résout le problème suivant :

$$\begin{array}{ll} \underset{d}{\text{minimiser}} & q_t(d) \\ \text{sujet à} & |d| \leq \Delta \end{array}$$

Si q_t est une bonne approximation de ϕ dans $[-\Delta,\Delta]$, on peut se permettre d'augmenter l'intervalle de confiance, donc Δ. Si q_t est une mauvaise approximation, alors on réduit Δ. Pour déterminer si on doit ajuster la taille de l'intervalle de confiance, on compare la réduction prévue par le modèle et la réduction actuelle de la fonction que l'on note respectivement par $\text{pred} = q_t(0) - q_t(d_R)$ et $\text{ared} = \phi(t) - \phi(t + d_R)$. À l'aide du ratio de ces deux quantités $r = \frac{\text{ared}}{\text{pred}}$ on a

un aperçu de la qualité de l'approximation. On pose r_1 et r_2 des seuils pour qualifier q_t. Si on a un r trop petit ($r < r_1$), notre fonction q_t n'est pas une bonne approximation de ϕ dans $[-\Delta,\Delta]$, alors on réduit Δ et on passe à la prochaine itération. Sinon on ajuste la valeur de t. Finalement, après avoir ajusté la valeur de t, si r est assez grand ($r > r_2$), on a que q_t est une bonne approximation de ϕ sur $[-\Delta,\Delta]$ et on peut alors se permettre d'augmenter la taille de notre région de confiance. Nous verrons que le choix des paramètres r_1 et r_2 influence la performance de l'algorithme plus tard.

Le pseudo-code est présenté dans l'algorithme 1 [Dus15].

Algorithme 1 : Région de confiance modèle quadratique

Données : $\{\phi$: une fonction une ou deux fois différentiable $\}$;
$\{q_t$ une approximation quadratique telle que $q_t(0) = \phi(t)\}$;
$\{\epsilon$ un critère d'arrêt $\}$;
$\{r_1, r_2$ des critères pour déterminer si on augmente la taille de l'intervalle de confiance $\}$;

$\Delta \leftarrow 2$ $t \leftarrow 1$;
répéter
  $d_R \leftarrow$ minimiseur global de q_t dans l'intervalle $|d| \leq \Delta$;
  ared$\leftarrow \phi(t) - \phi(t + d_R)$;
  pred$\leftarrow q_t(0) - q_t(d_R)$;
  r $\leftarrow \frac{\text{ared}}{\text{pred}}$;
  si *($r < r_1$)* **alors** $\Delta \leftarrow \Delta/2$;
  sinon
    $t \leftarrow t + d_R$;
    si $r > r_2$ **alors** $\Delta \leftarrow 2\Delta$;
jusqu'à *($|\phi'(t)| \leq \epsilon$)*;
Résultat $\leftarrow t$

L'algorithme de région de confiance a plusieurs propriétés intéressantes :

- il est relativement simple ;
- il possède des propriétés de convergence présentées dans le théorème suivant.

Théorème 2.1. *Lorsque $H = \phi''(t)$, l'algorithme de région de confiance converge vers un point t^* qui satisfait aux conditions nécessaires d'optimalité (équations 1 et 2).*

La preuve de convergence de l'algorithme n'est pas triviale, pour plus de détail voir [Dus15]. Notons toutefois que la convergence est obtenue "à l'infini" (de manière asymptotique). Étant donné qu'en réalité on dispose d'un nombre fini d'itérations, ce minimum ne pourra jamais être atteint. Ceci explique pourquoi notre critère d'arrêt est $|\phi'(t)| < \epsilon$ et non $\phi'(t) = 0$.

2.2 Choix de H

Pour déterminer le minimum global de $q_t(d)$, on utilise différentes variantes du modèle 3. Ces variantes consistent simplement en différentes manières d'estimer la dérivée seconde de ϕ, c'est-à-dire, le coefficient quadratique de $q_t(d)$. Afin d'alléger la notation, posons $f = \phi(t)$ et $g = \phi'(t)$. On a

$$q_t(d) = f + gd + \frac{1}{2}Hd^2.$$

On peut utiliser les modèles suivants :

1. Le modèle de Taylor consiste à utiliser $H = \phi''(t)$. Dans ce cas, le polynôme de Taylor de degré 2 de $\phi(t+d)$ est $q_t(d) = \phi(t) + \phi'(t)d + \frac{1}{2}\phi''(t)d^2$. Dans la littérature, ainsi que dans cet article, on le note modèle de Newton.

2. On peut également utiliser le modèle de la sécante : $H = \frac{g-\tilde{g}}{t-\tilde{t}} \approx \phi''(t)$ où $\tilde{t}$ est le t de l'itération précédente et $\tilde{g} = g(\tilde{t})$.

3. Un autre modèle possible est celui de la sécante améliorée. On pose : $s = t - \tilde{t}$, $y = g - \tilde{g}$, $\Gamma = 3(g-\tilde{g})s + 6(f-\tilde{f})$ et $\bar{y} = y + \frac{\Gamma}{s}$ où $\tilde{f} = f(\tilde{t})$. On a donc $H = \frac{\bar{y}}{s} \approx \phi''(t)$. Il s'agit d'une variante de la méthode de la sécante. Cette méthode est un cas particulier d'un algorithme en grande dimension, voir [ZX01] et [XZ01] pour plus de détail.

Dans tous les cas, le sommet de la parabole est approché par la direction de Newton $d_N = \frac{-g}{H}$. En effet,

$$q_t'(d) = g + Hd = 0 \Leftrightarrow d = \frac{-g}{H}.$$

On note que la variable par rapport à laquelle on dérive est d.

2.3 Choix de la direction de descente

Dans les méthodes de type région de confiance, afin de déterminer d (algorithme 1-ligne 4), on procède traditionnellement comme dans l'algorithme 2 en fonction du modèle considéré. Le d^* qui minimise $q_t(d)$ dans l'intervalle de confiance $[-\Delta,\Delta]$ représente la direction de descente.

Algorithme 2 : Choix traditionnel de la direction de descente pour les méthodes de région de confiance

```
1 Calcul de la direction de Newton : d_N = −g/H;
2 si q_t(Δ) < q_t(−Δ) alors
3 |  d_R = Δ;
4 sinon
5 |  d = −Δ;
6 si |d_N| < Δ et q_t(d) > q_t(d_N) alors d = d_N;
7 d_R ← d
```

Le procédé est simple, on évalue la fonction q_t aux bornes de notre intervalle ainsi qu'à son sommet et la plus petite valeur représente la direction de descente. Or, on peut déterminer d_R sans jamais évaluer la fonction q_t grâce à une analyse simple d'un polynôme de degré deux. Le théorème suivant établit la direction de descente en fonction du signe de H et de g.

Théorème 2.2. *Si on a $H > 0$ (donc une parabole orientée vers le haut), alors la direction de descente d^* est soit d_N ou la borne de l'intervalle de confiance de signe opposé à g. Sinon, si $H < 0$ (donc une parabole orientée vers le bas), alors la direction de descente est la borne de l'intervalle de confiance de signe opposé à g.*

On peut visualiser les différentes possibilités décrites dans le théorème précédent dans la table 1. On peut également s'intéresser aux cas où g ou H sont nuls. Il s'agit de cas particuliers, car obtenir zéro en calcul numérique est un phénomène plutôt rare.

- **Cas 1 :** $g = 0$. Si $H > 0$, alors $d = 0$ est un minimum global de $q_t(d)$ donc l'algorithme arrête. Sinon si $H < 0$, on se déplace vers l'extrémité de l'intervalle qui se rapproche le plus du minimum local et on arrête, donc $d = \pm\Delta$.

- **Cas 2 :** $H = 0$. Dans ce cas $q_t(d)$ est une droite (qui a un minimum à l'intérieur de l'intervalle de confiance). Donc, il s'agit de la même procédure que lorsque $H < 0$.

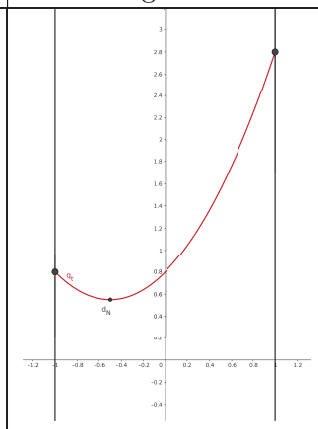
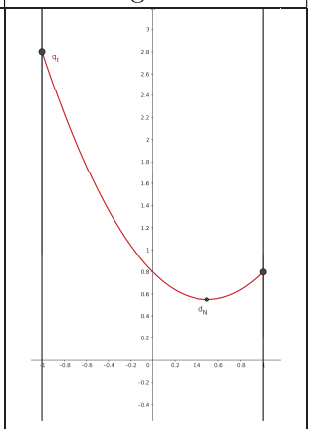
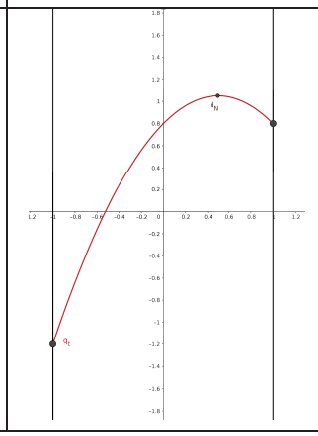
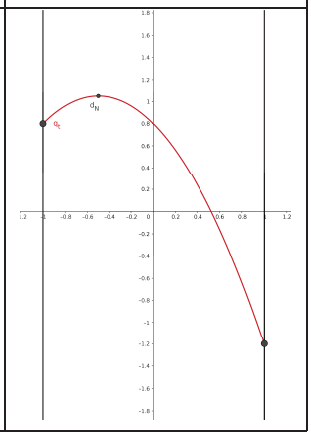
On peut donc représenter ce théorème dans l'algorithme 3. On rappelle que cette procédure vise à trouver d tel que nécessaire à la ligne 4 de l'algorithme 1.

Algorithme 3 : Choix de la direction de descente pour TR

```
1 Calcul de la direction de Newton : d_N = −g/H ;
2 si H > 0 alors
3 |   si g > 0 alors d_R =max(−Δ,d_N);
4 |   sinon d =min(d_N,Δ);
5 sinon
6 |   si g > 0 alors d = −Δ;
7 |   sinon d = Δ;
8 d_R ← d (minimiseur global de q_t dans [−Δ,Δ])
```

Changer l'algorithme 2 par l'algorithme 3 dans l'algorithme de région de confiance n'affecte en rien les performances de ce dernier. Il permet simplement des appels de moins à la fonction q_t que la version traditionnelle.

Table 1 – Cas possibles de région de confiance

	g>0	g<0
H>0		
H<0		

Remarque 2.3. L'idée derrière la preuve mathématique de cet algorithme est très simple. On a que $q_t'(0) = g$, le signe de g détermine donc la variation de q_t à l'origine. On sait que q_t est représenté par une parabole donc le signe de H détermine si elle est ouverte vers le bas ($H < 0$) ou si elle est ouverte vers le haut ($H > 0$).

- **Cas 1 :** Si $H > 0$ et $g > 0$, alors q_t est croissante en 0. Donc l'abscisse de son sommet est négative, car q_t est ouverte vers le haut. Ainsi, on compare simplement $-\Delta$ et d_N. Si $-\Delta \leq d_N$, alors on prend d_N (qui correspond au sommet) sinon, d_N n'est pas dans l'intervalle de confiance donc on prend $-\Delta$. Si $g < 0$, il s'agit du même cas, mais avec une symétrie par rapport à la droite $x = 0$.

- **Cas 2 :** De la même manière, si $H < 0$, q_t est ouverte vers le bas. Donc le sommet n'est jamais un minimum local (c'est un maximum local). Ainsi le minimum dans l'intervalle correspond simplement au Δ le plus loin de d_N.

3 Algorithme de régularisation cubique adaptative (ARC)

3.1 Présentation de l'algorithme

L'algorithme ARC suit la même idée que l'algorithme de TR, mais en ajoutant un terme de régularisation cubique. Donc, on ne cherche plus à minimiser $q_t(d)$ (équation 3), mais bien :

$$c_t(d) = q_t(d) + \frac{1}{3\alpha} |d|^3 \text{ où } \alpha > 0. \tag{4}$$

Dans cet algorithme, la région de confiance disparaît (il n'y a plus de Δ, il n'y a plus de contrainte). Le terme α a le même rôle que Δ. Plus α est grand, plus on se rapproche de $q_t(d)$ et inversement, si α est petit, on a une fonction dont les valeurs augmentent rapidement lorsque d tend vers l'infini (négatif ou positif). Le pseudo-code est présenté dans l'algorithme 4.

Algorithme 4 : Algorithme ARC

Données : {ϕ : une fonction une ou deux fois dérivable };
{c_t une approximation de ϕ };
{ϵ un critère d'arrêt };
{r_1, r_2 des critères pour déterminer si on augmente la taille de l'intervalle de confiance };

1 $t \leftarrow 0$;
2 $\tilde{t} \leftarrow 1$;
3 **répéter**
4 $\quad d \leftarrow$ minimiseur global de $c_t(d)$;
5 $\quad$ ared$\leftarrow \phi(t) - \phi(t+d)$;
6 $\quad$ pred$\leftarrow c_t(0) - c_t(d)$;
7 $\quad$ r $\leftarrow \frac{\text{ared}}{\text{pred}}$;
8 $\quad$ **si** $(r < r_1)$ **alors** $\alpha \leftarrow \alpha/2$;
9 $\quad$ **sinon**
10 $\quad\quad \tilde{t} \leftarrow t$;
11 $\quad\quad t \leftarrow t + d$;
12 $\quad$ **si** $r > r_2$ **alors** $\alpha \leftarrow 2\alpha$;
13 **jusqu'à** $(|\phi'(t)| \leq \epsilon)$;
14 Résultat $\leftarrow t$

La forme de l'algorithme est très similaire à celle de l'algorithme de région de confiance. La différence entre les deux est que l'approximation de ϕ est maintenant c_t et non q_t. Étant donné qu'on a une autre approximation, la stratégie pour déterminer le minimiseur global de c_t n'est plus la même que celle utilisée pour la région de confiance. C'est ce que nous allons explorer dans la section qui suit.

3.2 Calcul du minimiseur global simple et précis

Le problème qui nous intéresse est de trouver la valeur de d qui minimise $c_t(d)$ (ligne 4 de l'algorithme 4). La difficulté supplémentaire qui vient avec la recherche d'un minimiseur de $c_t(d)$ réside dans le terme cubique avec une valeur absolue absent de la méthode TR. Il y a potentiellement deux minima locaux à cette fonction et il n'y a pas de formule directe pour nous trouver le minimum global. Cependant, il est tout de même possible d'énoncer un théorème similaire au théorème 2.1 pour l'algorithme ARC.

Théorème 3.1. *Soit* $c_t(d) = f + gd + \frac{1}{2}Hd^2 + \frac{1}{3\alpha}|d|^3$. *Posons* $\delta_1 = H^2 - 4\frac{g}{\alpha}$ *et* $\delta_2 = H^2 + 4\frac{g}{\alpha}$. *Si* $H > 0$ *alors le minimum global de* c_t *est :*

$$d^* = \frac{-2g}{H + \sqrt{\delta_i}}.$$

Sinon, si $H \leq 0$ *alors :*

$$d^* = \frac{-H + \sqrt{\delta_i}}{\frac{(-2\,\text{signe}(g))}{\alpha}}.$$

Où $i = 1$ *si* $g < 0$ *et* $i = 2$ *si* $g > 0$.

Démonstration. On a

$$c_t(d) = f + gd + \frac{1}{2}Hd^2 + \frac{1}{3\alpha}\left|d\right|^3.$$

On dérive c_t pour trouver les points stationnaires.

$$c_t'(d) = g + Hd + \frac{1}{\alpha}d\left|d\right|.$$

Maintenant, on cherche les racines de $c_t'(d)$. Étant donné qu'il y a une valeur absolue ainsi que plusieurs cas possibles en vertu des signes de g et H, on représente les racines possibles (ainsi que le nombre total de racines) dans la table 2.

Table 2 – Racines de $c_t'(d)$

	$d > 0$	$d < 0$	nombre de racines
$H > 0$ et $g > 0$	-	$\frac{-H+\sqrt{H^2+4\frac{g}{\alpha}}}{\frac{-2}{\alpha}}$	1
$H < 0$ et $g > 0$	$\frac{-H\pm\sqrt{H^2-4\frac{g}{\alpha}}}{\frac{2}{\alpha}}$	$\frac{-H+\sqrt{H^2+4\frac{g}{\alpha}}}{\frac{-2}{\alpha}}$	3
$H > 0$ et $g < 0$	$\frac{-H+\sqrt{H^2-4\frac{g}{\alpha}}}{\frac{2}{\alpha}}$	-	1
$H < 0$ et $g < 0$	$\frac{-H+\sqrt{H^2-4\frac{g}{\alpha}}}{\frac{2}{\alpha}}$	$\frac{-H\pm\sqrt{H^2+4\frac{g}{\alpha}}}{\frac{-2}{\alpha}}$	3

On peut aussi s'intéresser aux cas particuliers où $H = 0$ et $g = 0$. Les racines associées à ces cas sont dans la table 3.

Table 3 – Racines de $c'(d)$ (cas particuliers)

	$d > 0$	$d < 0$	nombre de racines
$H > 0$ et $g = 0$	0	0	1
$H < 0$ et $g = 0$	$\frac{-H}{\frac{1}{\alpha}}$,0	$\frac{H}{\frac{1}{\alpha}}$,0	2
$H = 0$ et $g > 0$	-	$\frac{\sqrt{4(\frac{g}{\alpha})}}{\frac{-2}{\alpha}}$	1
$H = 0$ et $g < 0$	$\frac{\sqrt{-4(\frac{g}{\alpha})}}{\frac{2}{\alpha}}$	-	1
$H = 0$ et$g = 0$	0	0	1

Maintenant que l'on a les racines de c_t', on cherche à savoir lesquelles de celles-ci correspondent à des minima de c_t. Par construction de la méthode, lorsqu'on a une seule racine de $c_t'(d)$, il s'agit du minimum de c_t, car $\frac{1}{3\alpha}\left|d\right|^3 > 0$.

Lorsque $c'_t(d)$ a trois racines, les deux racines aux extrémités sont nécessairement des minima locaux de c_t et la racine au centre doit être un maximum local de c_t, tel qu'illustré à la figure 1.

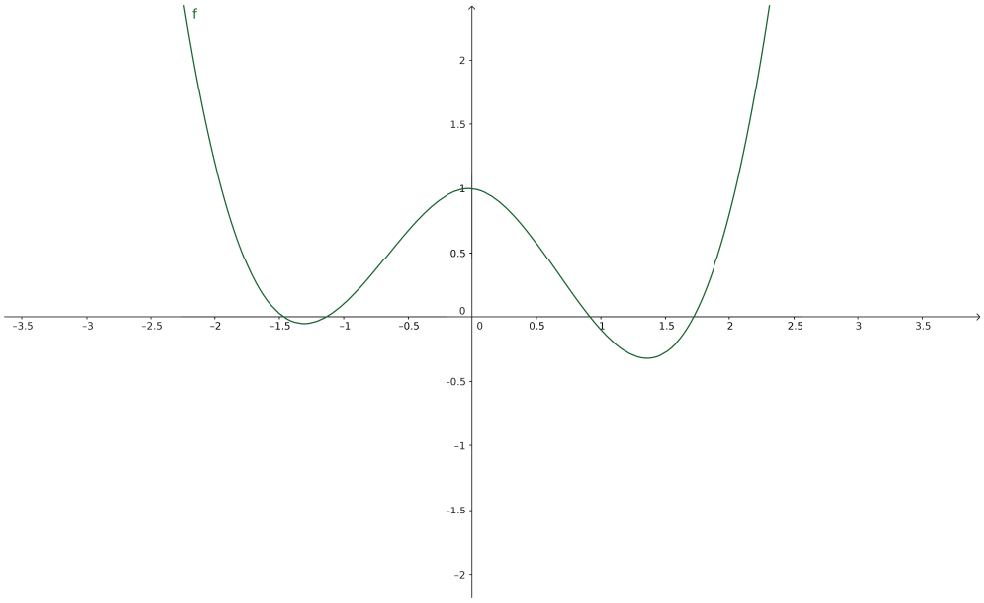


Figure 1 – Forme typique de c lorsque c'_t a trois racines

Pour s'assurer qu'on a des minima locaux, on utilise la dérivée seconde :

$$c''_t(d) = H + \frac{2}{\alpha}\,|d|\,.$$

On voit évidemment que dans les cas où $H > 0$ (les cas avec une seule racine), la racine est un minimum. En effet, $H > 0$, $\alpha > 0$ et $|d| > 0$ pour tout d. On a $c''_t(d) > 0$, donc on a un minimum local en vertu des conditions suffisantes d'optimalité (c'est-à-dire $c'_t(d) = 0$ et $c''_t(d) > 0$). Donc on a une procédure de sélection très simple lorsque $H > 0$. Si $g > 0$, alors on prend $d = \frac{-H+\sqrt{\delta_2}}{\frac{-2}{\alpha}}$ et sinon $d = \frac{-H+\sqrt{\delta_1}}{\frac{2}{\alpha}}$. Il nous reste à montrer que si $H < 0$, on a deux minima locaux.

- **Cas 1 :** $H < 0$ **et** $g > 0$**.** Les racines de c'_t sont respectivement

$$d_1 = \frac{-H-\sqrt{H^2-4\frac{g}{\alpha}}}{\frac{2}{\alpha}},\ d_2 = \frac{-H+\sqrt{H^2-4\frac{g}{\alpha}}}{\frac{2}{\alpha}} \text{ et } d_3 = \frac{-H+\sqrt{H^2+4\frac{g}{\alpha}}}{\frac{-2}{\alpha}}.$$

$$c''_t(d_1) = H + \frac{2}{\alpha}\left|\frac{-H-\sqrt{H^2-4\frac{g}{\alpha}}}{\frac{2}{\alpha}}\right| = \underbrace{H}_{<0} + \underbrace{\left|-H-\sqrt{H^2-4\frac{g}{\alpha}}\right|}_{<\,|\mathrm{H}|} < 0,$$

donc d_1 est un maximum local.

$$c_t''(d_2) = H + \frac{2}{\alpha}\left|\frac{-H+\sqrt{H^2-4\frac{g}{\alpha}}}{\frac{2}{\alpha}}\right| = \underbrace{H}_{<0} + \underbrace{\left|-H+\sqrt{H^2-4\frac{g}{\alpha}}\right|}_{>|\mathrm{H}|} > 0,$$

donc d_2 et un minimum local.

$$c_t''(d_3) = H + \frac{2}{\alpha}\left|\frac{-H+\sqrt{H^2+4\frac{g}{\alpha}}}{\frac{-2}{\alpha}}\right| = \underbrace{H}_{<0} + \underbrace{\left|-H+\sqrt{H^2+4\frac{g}{\alpha}}\right|}_{>|\mathrm{H}|} > 0,$$

donc d_3 et un minimum local.

Comme $d_1 - d_2 = \frac{-H-\sqrt{H^2-4\frac{g}{\alpha}}}{\frac{2}{\alpha}} - \frac{-H+\sqrt{H^2-4\frac{g}{\alpha}}}{\frac{2}{\alpha}} = \frac{-2\sqrt{H^2-4\frac{g}{\alpha}}}{\frac{2}{\alpha}} < 0$ et que $d_3 < 0$. Ainsi, on a bel et bien $d_3 < d_1 < d_2$. Le maximum local est encadré par les deux minimas locaux.

En conséquence, lorsque $H < 0$ et $g > 0$, la direction de descente est $\frac{-H+\sqrt{H^2-4\frac{g}{\alpha}}}{\frac{2}{\alpha}}$ ou $\frac{-H+\sqrt{H^2+4\frac{g}{\alpha}}}{\frac{-2}{\alpha}}$.

- **Cas 2 :** $H < 0$ **et** $g < 0$**.** Les racines de c_t' sont

$$d_1 = \frac{-H+\sqrt{H^2-4\frac{g}{\alpha}}}{\frac{2}{\alpha}}, d_2 = \frac{-H-\sqrt{H^2+4\frac{g}{\alpha}}}{\frac{-2}{\alpha}} \text{ et } d_3 = \frac{-H+\sqrt{H^2+4\frac{g}{\alpha}}}{\frac{-2}{\alpha}}.$$

$$c_t''(d_1) = H + \left|-H+\sqrt{H^2-4\frac{g}{\alpha}}\right| > 0,$$

donc d_1 est un minimum local.

$$c_t''(d_2) = H + \left|-H-\sqrt{H^2+4\frac{g}{\alpha}}\right| < 0,$$

donc d_2 est un maximum local.

$$c_t''(d_3) = H + \left|-H+\sqrt{H^2+4\frac{g}{\alpha}}\right| > 0,$$

donc d_3 est un minimum local. De plus, les racines s'ordonnent de la même manière que précédemment ($d_3 < d_2 < d_1$). Ainsi, si $H < 0$ et $g < 0$, la direction de descente est $\frac{-H+\sqrt{H^2-4\frac{g}{\alpha}}}{\frac{2}{\alpha}}$ ou $\frac{-H+\sqrt{H^2+4\frac{g}{\alpha}}}{\frac{-2}{\alpha}}$.

Le choix de la direction de descente se fait donc de la manière suivante : on pose $\delta_1 = H^2 - 4\frac{g}{\alpha}$ et $\delta_2 = H^2 + 4\frac{g}{\alpha}$. On observe qu'il est possible

que $H^2 - 4\frac{g}{\alpha}$ ou $H^2 + 4\frac{g}{\alpha}$ soit négatif. Toutefois, si c'est le cas, l'autre sera positif et nous donnera certainement un minimum global de c_t. On utilisera donc la direction issue de cette racine.

Nous avons donc une procédure de sélection lorsque $H < 0$ on choisit $d \in \{\frac{-H+\sqrt{\delta_2}}{\frac{-2}{\alpha}}, \frac{-H+\sqrt{\delta_1}}{\frac{2}{\alpha}}\}$ tel que $c_t(d)$ a la plus petite valeur possible. Or, comme nous l'avons vu avec les méthodes de région de confiance, il est possible de choisir la meilleure direction de descente sans évaluer $c_t(d)$.

Le signe de g nous indique la racine qui minimise $c_t(d)$ globalement.

- **Cas 1 :** $g < 0$, illustré à figure 2.

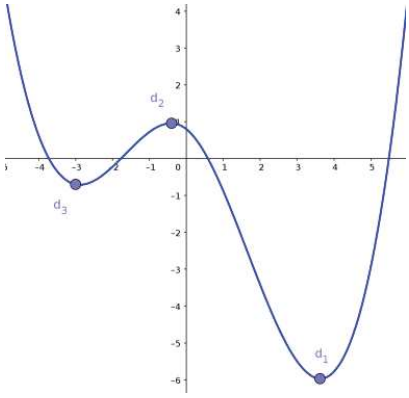


Figure 2 – Forme de c lorsque $H < 0$ et $g < 0$

Donc, si $g < 0$ et $H < 0$, alors le minimum global (donc la direction) est $\frac{-H+\sqrt{\delta_1}}{\frac{2}{\alpha}}$.

- **Cas 2 :** $g > 0$, illustré à la figure 3.

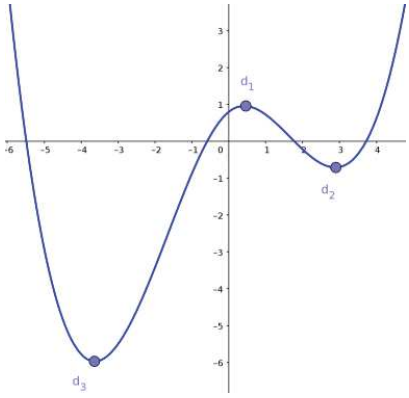


Figure 3 – Forme de c lorsque $H < 0$ et $g > 0$

Donc, si $g > 0$ et $H < 0$, alors le minimum global (donc la direction) est $\frac{-H+\sqrt{\delta_2}}{\frac{-2}{\alpha}}$.

□

3.3 Précision numérique

Un polynôme de la forme $ax^2 + bx + c$ admet ses racines en

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \Leftrightarrow x = \frac{-2c}{b \pm \sqrt{b^2 - 4ac}}.$$

Cette formule permet d'éviter les erreurs numériques lors de la programmation. En effet, on cherche à éviter l'addition de nombres de signes différents dans le code et on privilégie l'addition de nombres de mêmes signes. Par exemple, dans le choix de d, si $H > 0$ alors $-H + \sqrt{\delta_2}$ représente une addition de nombres de signes différents. En utilisant la formule alternative (celle à droite), on a $\frac{-2g}{H+\sqrt{\delta_2}}$ où l'on additionne seulement des nombres de mêmes signes.

Pour résumer, on énonce une procédure de sélection de direction qui traite tous les cas possibles (algorithme 5).

Algorithme 5 : Choix de la direction de descente pour ARC

```
si H > 0 alors
    si g > 0 alors d = -2g / (H + √δ2);
    sinon d = -2g / (H + √δ1);
sinon
    si g < 0 alors d = (-H + √δ1) / (2/α);
    sinon d = (-H + √δ2) / (-2/α);
d_R ← d
```

On remarque que l'analyse préalable rend le choix de la direction de descente très simple.

4 Tests avec le code

4.1 Algorithme de région de confiance

Les changements apportés à la procédure, c'est-à-dire trouver le minimiseur global de q_t sans utiliser d'évaluation de q_t) sont plutôt de nature esthétique, ils n'affectent pas les performances de l'algorithme, mais facilite l'écriture du code.

4.2 Algorithme de régularisation cubique adaptative

La procédure de choix de d a été incorporée dans un code existant en Julia [BKSE12] où le choix de d était plus arbitraire ("ancienne version" présentée

en annexe B) pour comparer les deux versions et ainsi valider l'implémentation de l'algorithme 5. Cet algorithme a été testé avec les problèmes d'optimisation unidimensionnelle de la bibliothèque *OptimizationProblems* voir [AMP], [A.S95] et [Dus15]. Notre outil de comparaison est le nombre d'évaluations des fonctions ϕ, ϕ' et ϕ'' lorsqu'on utilise les modèles de Newton.

Commençons par regarder ARC avec la variante de Newton ($H = \phi''(t)$). Les résultats sont présentés dans l'annexe A. Dans la table 4, on remarque que le nombre d'évaluations de fonctions avec la nouvelle version est toujours plus petit ou égal à celui de l'ancienne version. Les problèmes pour lesquels le nombre d'évaluations est strictement plus petit sont AMPGO20, Shpak2 et Shpak6.

Or, on remarque qu'avec le même algorithme, si on change le paramètre r_1 (qui gère si on réduit où non l'importance de α), nos résultats vont changer. Dans l'exemple précédent, nous avions une valeur de $r_1 = 0.1$, mais si l'on prend une valeur de $r_1 = 0.4$, on obtient des résultats différents (voir table 4). Pour le problème Shpak2, l'ancienne version de l'algorithme a besoin de moins d'évaluations de fonctions. Pour les problèmes Shpak6 et AMPGO20, la nouvelle version a des meilleures performances.

On observe un phénomène similaire pour les algorithmes sécantes où le choix de r_1 influence beaucoup l'efficacité de la nouvelle version de l'algorithme. Par exemple, avec ARC-Sécante et r_1=0.1 (table 5), on observe que la nouvelle version est meilleure pour les problèmes Shpak6, Shpak2 et Shpak1. Or, l'ancienne version est meilleure pour les problèmes AMPGO02 et Dus2_9. Si on prend $r_1 = 0.4$, on obtient des résultats différents. Avec ce nouveau paramètre, l'ancienne version est plus efficace pour le problème AMPGO05. Toutefois la nouvelle version est meilleure pour les problèmes AMPGO02, AMPGO03, AMPGO08 et Shpak2.

Ainsi, le choix du paramètre r_1 est un facteur qui influence la performance des deux versions de l'algorithme. La nouvelle version de la procédure de sélection de d reste toutefois meilleure dans la majorité des cas. Il n'existe pas de choix optimal pour r_1 et r_2. De manière générale on prend $r_1 = 0.25$ et $r_2 = 0.75$ ou $r_1 = 0.3$ et $r_2 = 0.7$.

5 Conclusion

Pour conclure, nous avons vu le fonctionnement des méthodes de région de confiance ainsi que des méthodes de régularisation cubique adaptative en dimension un. Ces méthodes approximent une fonction objectif ϕ par une une fonction quadratique (q_t) ou cubique (c_t) et minimisent ces dernières. Nous avons montré qu'il est possible de trouver des minima globaux de ces fonctions de manière simple et efficace en utilisant seulement le signe des coefficients de degré un et deux de q_t et c_t. Évidemment, ces méthodes ne sont pas restreintes aux polynômes de degré deux ou trois, ainsi il est possible de prendre des polynômes de degré supérieur. Cependant, des outils de calcul de racines automatisés seront nécessaires pour déterminer les minima globaux. Le perfectionnement des

algorithmes de minimisation unidimensionnelle compte d'autres applications en optimisation. Entre autres, les méthodes de région de confiance et ARC ont été adaptées pour la recherche linéaire. Les méthodes ARC et TR sont également adaptables au cas multidimensionnel.

Références

[AMP] 1-D Test Function. `http://infinity77.net/global\_optimization/test\_functions\_1d.html`. Accédé le : 2017-05-08.

[A.S95] A.Shpak : Global optimization in one-dimensional case using analytically defined derivatives of objective function. *Computer Science Journal of Moldova*, 3(2):168–184, 1995.

[BKSE12] Jeff Bezanzon, Stefan Karpinski, Viral Shah et Alan Edelman : Julia : A Fast Dynamic Language for Technical Computing. *In Lang.NEXT*, avril 2012.

[Dus15] J.-P. Dussault : *Optimisation mathématiques avec applications en imagerie*. Sherbrooke, Québec, Canada, 2015.

[XZ01] Chengxian Xu et Jianzhong Zhang : A Survey of Quasi-Newton Equations and Quasi-Newton Methods for Optimization. *Annals of Operations Research*, 103(1):213–234, 2001.

[ZX01] Jianzhong Zhang et Chengxian Xu : Properties and numerical performance of quasi-Newton methods with modified quasi-Newton equations. *Journal of Computational and Applied Mathematics*, 137(2):269–278, 2001.

Samuel Goyette
Département de mathématiques, Université de Sherbrooke
Courriel: `Samuel.Goyette@USherbrooke.ca`

A Résultats numériques pour ARC-Nwt et ARC-Sec

On présente les résultats numériques de ARC-Nwt et ARC-Sec. On a appliqué ces algorithmes à une collection de problèmes avec différentes valeurs de r_1 pour illustrer que ce paramètre n'a pas d'influence sur la convergence. Ainsi, l'algorithme modifié est plus performant que sa version standard.

Table 4 – Nombre d'évaluations de fonction avec ARC-Nwt

	$r_1 = 0.1$		$r_1 = 0.4$	
Nom du problème	Ancienne version	Nouvelle version	Ancienne version	Nouvelle version
AMPGO02	20	20	22	22
AMPGO03	24	24	24	24
AMPGO04	18	18	18	18
AMPGO05	24	24	24	24
AMPGO06	6	6	6	6
AMPGO07	20	22	22	22
AMPGO08	15	15	15	15
AMPGO09	15	15	15	15
AMPGO10	12	12	12	12
AMPGO12	6	6	6	6
AMPGO18	15	15	15	15
AMPGO20	31	28	31	28
AMPGO22	26	26	25	25
DUS2 1	114	114	114	114
DUS2 3	12	12	12	12
DUS2 9	24	24	24	24
DUSCUBE	20	20	20	20
SHPAK1	18	18	17	17
SHPAK2	20	17	19	22
SHPAK3	15	15	21	21
SHPAK5	17	17	17	17
SHPAK6	23	20	26	20

Table 5 – Nombre d'évaluations de fonction avec ARC-Sec

	$r_1 = 0.1$		$r_1 = 0.4$	
Nom du problème	Ancienne version	Nouvelle version	Ancienne version	Nouvelle version
AMPGO02	16	24	18	16
AMPGO03	34	34	34	32
AMPGO04	12	12	12	12
AMPGO05	34	34	32	34
AMPGO06	4	4	4	4
AMPGO07	26	26	16	16
AMPGO08	34	34	30	28
AMPGO09	14	14	14	14
AMPGO10	4	4	4	4
AMPGO12	4	4	4	4
AMPGO18	12	12	12	12
AMPGO20	44	44	44	44
AMPGO22	14	14	14	14
DUS2 1	102	102	102	102
DUS2 3	14	14	14	14
DUS2 9	66	70	24	24
DUSCUBE	16	16	18	18
SHPAK1	18	16	16	16
SHPAK2	18	16	20	18
SHPAK3	28	28	28	28
SHPAK5	36	36	38	38
SHPAK6	38	36	36	36

B Ancienne méthode choix de d pour ARC

Nous présentons ici l'ancienne procédure de sélection du minimiseur global de c_t. Étant donné que cette méthode manque de finition et ne résulte pas d'un raisonnement mathématique que l'on peut trouver, elle n'est pas présentée dans le cadre principal de l'article. Malgré le fait que cette procédure ressemble à celle que l'on a déjà, elle est incomplète et utilise des évaluations de c_t.

Algorithme 6 : Choix traditionnel de la direction de descente pour ARC

$\delta_0 = H^2 - 4\frac{g}{\alpha}$;
si $\delta_0 < 0$ **alors** $\delta_0 = H^2 + 4\frac{g}{\alpha}$;
si $H < 0$ **alors**
    $d_R p = \frac{-H+\sqrt{\delta_0}}{\frac{2}{\alpha}}$
si $H \geq 0$ **alors**
    $d_R p = \frac{-2g}{H=\sqrt{\delta_0}}$
$d_R n = \frac{-H+\sqrt{\delta_0}}{\frac{2}{\alpha}}$;
si $c_t(d_R n) < c_t(d_R p)$ **alors** $d_R = d_R n$;
sinon $d_R = d_R p$;
On retourne d_R