



TP3 : différences finies pour l'optimisation

O. Cots, J. Gergaud, S. Jerad et D. Ruiz

1 Contexte

Pour résoudre un problème d'optimisation aux moindres carrés non linéaire via l'algorithme de Gauss-Newton, on utilise la matrice jacobienne de la fonction qui code les résidus. Celle-ci n'est pas toujours facile à coder ou ne peut être obtenue que par une approximation numérique via les différences finies. On se propose ici de coder le plus correctement possible cette approximation. $\|\cdot\|$ dénote la norme euclidienne usuelle et $\mathcal{N}(0, 1)$ une variable aléatoire Gaussienne centrée réduite.

2 Schéma avant et erreur numérique

2.1 Formulation mathématique

Soit f une fonction lisse de \mathbb{R}^n dans \mathbb{R}^m , x un point de \mathbb{R}^n et v un vecteur de \mathbb{R}^n . Si on note, $g : h \mapsto f(x + hv)$. Pour h proche de 0, on a d'après la formule de Taylor-Young :

$$g(h) = \sum_{i=0}^n \frac{h^i}{i!} g^{(i)}(0) + R_n(h), \quad R_n(h) = o(h^n),$$

ou d'après Taylor-Lagrange,

$$\|R_n(h)\| \leq \frac{M_n h^{n+1}}{(n+1)!},$$

de même,

$$g(-h) = \sum_{i=0}^k \frac{(-h)^i}{i!} g^{(i)}(0) + R_n(-h).$$

Formule des différences finies avants La méthode des différences finies avants consiste à approcher la différentielle de f en x dans la direction v par

la formule suivante :

$$\frac{f(x + hv) - f(x)}{h} = \frac{g(h) - g(0)}{h} = g'(0) + \frac{h}{2}g''(0) + \frac{h^2}{6}g^{(3)}(0) + o(h^2). \quad (1)$$

L'approximation ainsi obtenue de $g'(0) = df(x).v \in \mathbb{R}^m$ est **d'ordre 1** si $g^{(2)}(0) \neq 0$ ou au moins d'ordre 2 sinon.

Remarque 2.1. En prenant $v = e_i$ le i -ème vecteur de la base canonique et $h = h_i$, On obtient la i -ème colonne de la matrice $J_f(x)$.

Définition 2.2. Sur machine ou si les données sont issus d'expériences physique, les calculs se font en virgule flottante et peuvent contenir aussi des termes d'erreurs intrinsèques. On note ω , une borne sur le terme d'erreur de tel sorte que si nous notons $num(g, h)$ la valeur de $g(h)$ calculé numériquement, on suppose que l'on puisse majorer l'erreur relative numérique par :

$$\|num(g, h) - g(h)\| := \|e_h\| \leq \omega L_f, \quad (2)$$

ou L_f est une constante qui dépend de la valeur de f sur le domaine d'intérêt. Ainsi, pour $h > 0$ petit, on a :

$$\begin{aligned} \left\| \frac{num(g, h) - num(g, 0)}{h} - g'(0) \right\| &= \left\| \frac{g(h) + e_h - g(0) - e_0}{h} - g'(0) \right\| \\ &= \left\| \frac{g(h) - g(0) - hg'(0)}{h} + \frac{e_h - e_0}{h} \right\| \\ &= \left\| \frac{R_1(h)}{h} + \frac{e_h - e_0}{h} \right\| \\ &= \underbrace{\frac{M_1 h}{2}}_{\text{Erreur d'approximation}} + \underbrace{2 \frac{\omega L_f}{h}}_{\text{Erreur numérique}}. \end{aligned} \quad (3)$$

Le majorant trouvé atteint son minimum en

$$h_* = 2\sqrt{\frac{\omega L_f}{M_1}}.$$

Par exemple pour des nombres en virgule flottante, ω sera égale à eps_{mach} qui est le plus petit nombre tel que $1 + eps_{mach} \neq 1$. Cette constante (qui dépend des machines) est donnée par **eps** en MATLAB. Attention si vous affectez à **eps** une quantité, vous perdez l'accès à cette constante. Cette quantité dépend de la machine et de l'encodage des données.

En considérant que $L_f \simeq M_1$, alors le choix se révélant le plus optimal est

$$h_* \simeq \sqrt{\omega}. \quad (4)$$

2.2 Simulations numériques sur fonctions test

On demande de tester pour la fonction \cos aux points $x_0 = \pi/3$, $x_1 = 10^6 * \pi/3$ et la fonction $\cos + 10^{-8}\mathcal{N}(0, 1)$ au point $x_0 = \pi/3$ l'erreur entre les différences finies et la dérivée au point considéré en fonction de h . On prendra $h = 10^{-i}$ pour $i = \{1, \dots, 16\} \cup \{-\log_{10}(h_*)\}$ et on tracera ces erreurs dans une échelle logarithmique (fonction MATLAB `loglog`).

Codez la fonction

```
%function Jac = forwardfiniteDiff(fun,x, h)
```

dans le fichier *main_diff_finies_cosinus.m* et exécuter le fichier. Quels commentaires pouvez-vous faire ? Quand l'approximation (4) est-elle valide ?

3 Schéma centré

3.1 Définition

On peut utiliser un schéma de différences finies centrées pour calculer la dérivée de g .

$$\frac{f(x + hv) - f(x - hv)}{2h} = \frac{g(h) - g(-h)}{2h} = g'(0) + g^{(3)}(0)\frac{h^2}{6} + \mathcal{O}(h^4), \quad (5)$$

l'approximation ainsi obtenue de $df(x).v \in \mathbb{R}^m$ est **d'ordre 2** si $g^{(3)}(0) \neq 0$ ou au moins d'ordre 4 sinon. À noter que ce schéma nécessite plus d'évaluations de la fonction f .

On peut montrer comme dans (4), le meilleur h est de l'ordre

$$h_* \simeq \sqrt[3]{\omega}. \quad (6)$$

ou ω est définie dans (2).

3.2 Simulations numériques

En reprenant les mêmes points et fonctions que dans la sous-section 2.2, codez la fonction

```
%function Jac = centredfiniteDiff(fun,x, h)
```

dans le fichier *main_diff_finies_cosinus.m* et changer la valeur de la variable `methode_finite_diff` à `centrees` pour utiliser cette méthode. Quels commentaires pouvez-vous faire ?

4 Schéma complexe

Les formules des schémas avant et centrées sont sensibles aux calculs de la différence $\Delta f = f(x+h) - f(x)$ ou $\Delta f = f(x+h) - f(x-h)$. Pour remédier à ce problème, les différences finies à pas complexe ont été introduites¹.

Si on suppose que la fonction g est holomorphe, (c'est à dire dérivable au sens complexe), on peut considérer un pas complexe ih . Un développement limité de g en 0 s'écrit

$$f(x + ihv) = g(ih) = g(0) + ihg'(0) - \frac{h^2}{2}g^{(2)}(0) - i\frac{h^3}{6}g^{(3)}(0) + \mathcal{O}(h^3),$$

On considère l'approximation :

$$df(x).v = g'(0) = \frac{\text{Im}(g(x + ih))}{h} + \mathcal{O}(h^2) \simeq \frac{\text{Im}(g(x + ih))}{h}, \quad (7)$$

on peut prouver que l'approximation (7) est au moins d'ordre 2 et aussi démontrer que l'ordre du grandeur de h_* est :

$$h_* \leq \sqrt{\text{eps}_{mach}}. \quad (8)$$

(eps_{mach} est le plus petit nombre tel que $1 + \text{eps}_{mach} \neq 1$.)

4.1 Simulations numériques

En reprenant les mêmes points et fonctions que dans la sous-section (2.2), codez la fonction

```
%function derivee = derivee_complexe(fun, x, h)
```

dans le fichier *main_diff_finies_cosinus.m* et changer la valeur de la variable `methode_finite_diff` à `complexes` pour utiliser cette méthode. Quels commentaires pouvez-vous faire ?

5 Application à l'algorithme de Gauß-Newton

Dans cette section, on se propose de reprendre l'algorithme de Gauß-Newton mais en utilisant cette fois l'approximation par différences finies au lieu d'utiliser la vraie matrice jacobienne. Les résultats seront obtenus en exécutant les fichiers *C14_diff_finies.m* et *exem1_diff_finies.m*.

1. Pour plus de détails, vous pouvez consulter l'article *Martins, J. R., Sturdza, P., et Alonso, J. J. (2003). The complex-step derivative approximation. ACM Transactions on Mathematical Software (TOMS), 29(3), 245-262.*

5.1 Différences finies avant

S'inspirant de l'analyse faite précédemment, on choisit un pas de calcul h_j proche de h_* de (4), on définit

- $\omega = \max(\epsilon_{\text{smach}}, 10^{-\text{ndigit}})$ où ndigit est le nombre de décimales calculé de façon exacte lors d'une évaluation de la fonction f .
- $h_j = \sqrt{\omega} \max(|x_j|, 1) \text{signe}(x_j)$ (ici x_j représente la j^{e} composante du vecteur x).

Remarque 5.1. Attention ici la fonction *signe* renvoie 1 si $x = 0$, contrairement à la fonction *sign* de MATLAB.

Dans le fichier *diff_finies_avant.m*, coder la fonction suivante à l'aide de la formule de l'équation (1).

```
function Jac = diff_finies_avant(fun,x,option)
```

Exécuter ensuite les deux fichiers de test. Varier *ndigit* et commenter son influence sur la convergence du Gauß-Newton approximatif (Dans le fichier test, changer la valeur de la variable *ndigits*). Comparer l'algorithme approximatif avec un algorithme exact (Dans le fichier test, changer la valeur du booléen *True_Jacobienne*).

Remarque 5.2. Les résultats de vos expériences numériques sont sauvegardés dans un fichier *.txt*

5.2 Différences finies centrées

S'inspirant de l'analyse faite en section 3, on choisit un pas h_j proche de la valeur trouvée h_* dans l'équation (6).

- $\omega = \max(\epsilon_{\text{smach}}, 10^{-\text{ndigit}})$ où ndigit est le nombre de décimales calculé de façon exacte lors d'une évaluation de la fonction f .
- $h_j = \sqrt[3]{\omega} \max(|x_j|, 1) \text{signe}(x_j)$ (ici x_j représente la j^{e} composante du vecteur x).

Dans le fichier *diff_finies_centree.m*, coder la fonction suivante avec la formule de l'équation (5)

```
function Jac = diff_finies_centree(fun,x, option)
```

Exécuter les deux fichiers tests et reprenez le même protocole que dans la partie 5.1 .

6 Questions Théoriques

1ère question Montrer que le schéma complexe (7) est d'ordre 2.

2ème question En reprenant la même démarche que dans (3) , retrouver l'ordre de grandeur pour le h_* donnée en (6) pour le schéma centrée (5).