



TP – Optimisation

1 Introduction

1.1 Objectif

L'objectif de ces 2 TP d'optimisation est de programmer les méthodes de Gauß-Newton (voir page 58 du polycopié sous Moodle) et de Newton (voir page 57 du polycopié sous Moodle) pour un problème aux moindres carrés

$$(\mathcal{P}) \begin{cases} \text{Min } f(\beta) = (1/2) \|r(\beta)\|^2 \\ \beta \in \mathbb{R}^p \end{cases}$$

où r est la fonction résidus

$$\begin{aligned} r: \mathbb{R}^p &\longrightarrow \mathbb{R}^n \\ \beta &\longmapsto r(\beta). \end{aligned}$$

1.2 Exemple traité

Le carbone radioactif ^{14}C est produit dans l'atmosphère par l'effet des rayons cosmiques sur l'azote atmosphérique. Il est oxydé en $^{14}\text{CO}_2$ et absorbé sous cette forme par les organismes vivants qui, par suite, contiennent un certain pourcentage de carbone radioactif relativement aux carbones ^{12}C et ^{13}C qui sont stables. On suppose que la production de carbone ^{14}C atmosphérique est demeurée constante durant les derniers millénaires. On suppose d'autre part que, lorsqu'un organisme meurt, ses échanges avec l'atmosphère cessent et que la radioactivité due au carbone ^{14}C décroît suivant la loi exponentielle suivante :

$$A(t) = A_0 e^{-\lambda t}$$

où λ est une constante positive, t représente le temps en année, et $A(t)$ est la radioactivité exprimée en nombre de désintégrations par minute et par gramme de carbone. On désire estimer les paramètres A_0 et λ par la méthode des moindres carrés. Pour cela on analyse les troncs (le bois est un tissu mort) de très vieux arbres *Sequoia gigantea* et *Pinus aristata*. Par un prélèvement effectué sur le tronc, on peut obtenir :

- son âge t en année, en comptant le nombre des anneaux de croissance,
- sa radioactivité A en mesurant le nombre de désintégration.

t	500	1000	2000	3000	4000	5000	6300
A	14.5	13.5	12.0	10.8	9.9	8.9	8.0

2 Travail demandé

2.1 Introduction

- Petit complément Matlab :
 - Les fonctions : voir <https://www.mathworks.com/help/matlab/ref/function.html> ;
 - Les boucles `while` : voir <https://www.mathworks.com/help/matlab/ref/while.html>.
- **Il ne faut en aucun cas modifier les interfaces des fonctions qui sont définies dans les codent fournis.** Voir les entêtes de ces fonctions dans les fichiers fournis pour connaître cette interface.
- **Les résultats que vous devez obtenir sont en annexe du présent document.**

2.2 Algorithme de Gauß-Newton

- Compléter, à la fin du fichier `Modelisation_C14.m`, les 2 fonctions `residu_C14` et `J_residu_C14` qui codent respectivement la fonction résidus r et la fonction matrice jacobienne des résidus J_r . On codera ces fonctions sans aucune boucle. Concernant la fonction `J_residu_C14`, on vous demande de coder la fonction que vous aurez calculer auparavant à la main. Il ne faut en aucun cas utiliser de fonction Matlab du type `jacobian` ou autre.
- Compléter la fonction `Algo_Gauss_Newton` définit dans le fichier de même nom qui code l'algorithme de Newton. On utilisera dans un premier temps comme seul test d'arrêt le nombre maximum d'itérations atteint.
- Exécuter le script `Modelisation_C14.m` afin de vérifier vos résultats.
- Implémenter dans la fonction `Algo_Gauss_Newton` tous les tests d'arrêt suivants. On note $\beta^{(0)}$ le point initial et $\beta^{(k)}$ l'itéré courant. On pose aussi `Tol_rel` et `Tol_abs` = $\sqrt{\varepsilon_{mach}}$ (`sqrt(eps)` en Matlab). Les tests d'arrêt seront les suivants :
 1. $\|\nabla f(\beta^{(k+1)})\|$ petit : $\|\nabla f(\beta^{(k+1)})\| \leq \max(\text{Tol_rel}\|\nabla f(\beta^{(0)})\|, \text{Tol_abs})$;
 2. Évolution de $f(\beta^{(k+1)})$ petit : $|f(\beta^{(k+1)}) - f(\beta^{(k)})| \leq \max(\text{Tol_rel}|f(\beta^{(k)})|, \text{Tol_abs})$
 3. Évolution du pas $\delta^{(k)} = \beta^{(k+1)} - \beta^{(k)}$ petit : $\|\beta^{(k+1)} - \beta^{(k)}\| \leq \max(\text{Tol_rel}\|\beta^{(k)}\|, \text{Tol_abs})$
 4. Le nombre d'itération maximal est atteint.

2.3 Algorithme de Newton

- Compléter, à la fin du fichier `Modelisation_C14.m`, la fonction `Hess_f_C14` et qui renvoie :
 - la matrice hessienne de la fonction coût f au point ;
 - les résidus ;
 - La matrice jacobienne des résidus.
- Compléter la fonction `Algo_Newton` définit dans le fichier de même nom qui code l'algorithme de Newton.
- Exécuter le script `Modelisation_C14.m` afin de vérifier vos résultats.

3 Tests numériques

Réaliser quelques tests numériques :

- en modifiant le point de départ $\beta^{(0)}$;
- en modifiant le vecteur des options.

A Résultats

Algorithme de Gauss-Newton

residu_C14(beta0, Donnees)

4.9877
4.4516
3.8127
3.3918
3.1968
2.8347
2.6741

J_residu_C14(beta0, Donnees)

-0.95123 4756.1
-0.90484 9048.4
-0.81873 16375
-0.74082 22225
-0.67032 26813
-0.60653 30327
-0.53259 33553

nb_iter	A0	lambda	f'(beta)	f(beta)	delta	exitflag
0	10	0.0001	4.6322e+05	48.07		
1	15.022	0.00010633	15913	0.10507	5.0219	4
2	15.025	0.00010433	5.9024	0.088621	0.0032964	4
3	15.025	0.00010432	0.39911	0.088621	0.00068766	4
4	15.024	0.00010432	0.004769	0.088621	4.9165e-06	2
4	15.024	0.00010432	0.004769	0.088621	4.9165e-06	2

4	15.024	0.00010432	0.004769	0.088621	4.9165e-06	2
4	15.024	0.00010432	0.004769	0.088621	4.9165e-06	2
4	15.024	0.00010432	0.004769	0.088621	4.9165e-06	2

Algorithme de Newton

Hessienne $f(\beta^{\{0\}})$

4.0436 -50497
-50497 1.8899e+09

nb_iter	A0	lambda	$\ f'(\beta)\ $	$f(\beta)$	$\ \delta\ $	exitflag
0	10	0.0001	4.6322e+05	48.07		
1	12.715	-7.255e-05	3.1785e+06	159.19	2.7154	4
2	12.053	-1.8362e-05	9.7291e+05	34.783	0.66279	4
3	16.55	0.00010635	2.1583e+05	4.2383	4.4973	4
4	14.755	9.8263e-05	9600.9	0.14691	1.7953	4
5	15.022	0.00010427	80.835	0.088626	0.26754	4
6	15.024	0.00010432	0.0064489	0.088621	0.0023669	4
7	15.024	0.00010432	7.1168e-11	0.088621	1.8799e-07	1
7	15.024	0.00010432	7.1168e-11	0.088621	1.8799e-07	1

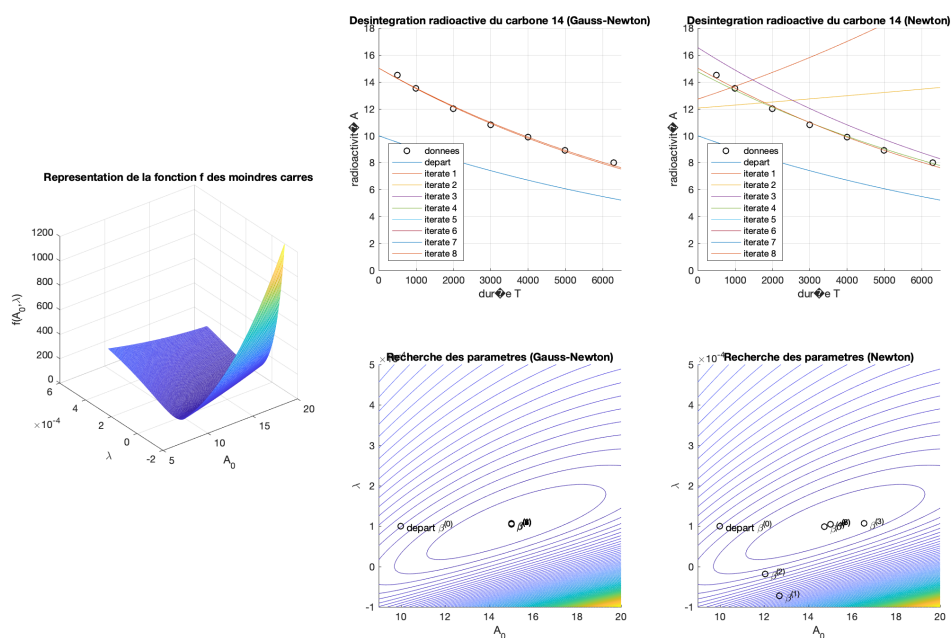


FIGURE 1 – *Algorithme de Gauß-Newton et de Newton, point de départ $\beta^{(0)} = (10, 0.0001)$.*