Explorer l'analyse de La régression non linéairee avec Maple.

Han-Ping Li, Francis Ngaka

Explorer L'analyse de La Régression Non Linéairee avec Maple

Han-Ping LI, Francis NGAKA
Université Louis Pasteur, Strasbourg, France

Résumé

In this paper, a formal Maple programme is developped which allows us to read the input of a nonlinear regression problem $y = f(x, \theta) + \epsilon$ and gives us the point estimation and different confidence regions for the parametre θ.

AMT Codes: 62J02, 62F10, 62F25

Key Words: Nonlinear regression, Point estimation, Confidence region.

Table des matières

Chapitre 1

Introduction

1.1 Rappels sur la régression

L'analyse de la régression est une méthode statistique permettant de modéliser une relation entre Y une variable aléatoire et $X = (x_1, ..., x_k)$ un vecteur de k variables déterminées réelles selon la relation suivante :

$$Y = f(X, \boldsymbol{\theta}) + \varepsilon$$

où $\boldsymbol{\theta} = {}^t(\theta_1, ..., \theta_p)$ est un vecteur de p paramètres inconnus fixés et ε le terme d'erreur qui est une variable aléatoire centrée et de variance constante σ^2. Si f est linéaire en $\boldsymbol{\theta}$ c'est-à-dire $f(X, \boldsymbol{\theta}) = \sum_{j=0}^{p} \theta_j h_j(X)$ où les h_j sont des fonctions connues, nous parlons alors de modèle de **régression linéaire**. Par contre si $f(X, \boldsymbol{\theta})$ n'est pas linéaire en $\boldsymbol{\theta}$ alors nous aurons un modèle de **régression non linéaire**.

On dispose de n observations du couple (X, Y) avec Y vecteur aléatoire $X = (x_1, ..., x_k)$ constante qui vérifient pour chaque i $\in \{1,...,n\}$: $Y_i = f(X_i, \theta) + \varepsilon_i$. Si on note

$$\boldsymbol{Y} = \begin{pmatrix} Y_1 \\ \vdots \\ Y_n \end{pmatrix}, \boldsymbol{F(X)} = \begin{pmatrix} f(X_1, \theta) \\ \vdots \\ f(X_n, \theta) \end{pmatrix} \text{ et } \boldsymbol{\varepsilon} = \begin{pmatrix} \varepsilon_1 \\ \vdots \\ \varepsilon_n \end{pmatrix},$$

$$\boldsymbol{X} = \begin{pmatrix} x_{11} & \cdots & x_{1k} \\ x_{21} & \cdots & x_{2k} \\ \vdots & \vdots & \vdots \\ x_{n1} & \cdots & x_{nk} \end{pmatrix}$$

on a alors une relation générale

$$\boldsymbol{Y} = \boldsymbol{F}(\boldsymbol{X}) + \boldsymbol{\varepsilon}$$

où $\boldsymbol{\varepsilon} = {}^t(\varepsilon_1, ..., \varepsilon_n)$ est un vecteur de variables aléatoires centrées qui a pour matrice de variance-covariance constante $\sigma^2 I_n$ c'est-à-dire

$$\mathbf{Y} = F(\boldsymbol{X}, \boldsymbol{\theta}) + \varepsilon$$

avec $F(\boldsymbol{X}, \boldsymbol{\theta}) = {}^t(f(X_1, \theta), ..., f(X_n, \theta))$ et $\boldsymbol{\varepsilon} = {}^t(\varepsilon_1, ..., \varepsilon_n)$.

Ainsi nous allons étudier deux aspects de la régression non linéaire :

- **l'estimation** du paramètre $\boldsymbol{\theta}$ selon le critère des moindres carrées à partir d'un échantillon $(X_1, Y_1), ..., (X_n, Y_n)$ qui dans le cas non linéaire ne peut pas être calculée explicitement

- la construction des **régions de confiance** pour ce paramètre $\boldsymbol{\theta}$ par trois différentes approches :
  i) probabiliste
  ii) par la vraisemblance
  iii) par Bayes

1.2 Objectifs de l'étude de la régression non linéaire

Nous nous proposons de développer un programme formel en **Maple** qui nous permet de lire un fichier de données, la fonction f et résoudre les deux problèmes posés par la régression non linéaire que sont l'estimation et les régions de confiance du paramètre θ.

Ce programme devra ainsi pouvoir résoudre formellement ces deux problèmes à partir de n'importe quel modèle de régression non linéaire régulier.

Pour illustrer le programme, notamment l'aspect géométrique nous prenons comme exemple les deux modèles suivants :

- le modèle de **Michaelis-Menten**

- le modèle de **BOD** (biochimical oxygen demand) d'une expérience réalisée par D.Marske en 1967.

où **p**=dim(θ)=2.

1.3 Présentation des modèles étudiés

Nous allons résoudre les problèmes **d'estimation** et de construction de **regions de confiance** du paramètre $\boldsymbol{\theta}$ posés par la **régression non linéaire** pour les deux modèles expérimentales **Michaelis-Menten** et **BOD**.

1.3.1 Le modèle Michaelis-Menten

Ce modèle permet d'établir une relation entre la vélocité de la réaction chimique d'un enzyme et sa concentration x par l'équation :

$$f(x, \theta) = \frac{\theta_1 x}{\theta_2 + x}$$

Le tableau suivant nous donne les données des différentes vélocités de la reaction suivant les concentrations de l'enzyme traitée à la puromycine obtenues par l'expérimentation.

substance concentration (ppm)	vélocité $(counts/min^2)$
0.02	76
	47
0.06	97
	107
0.11	123
	139
0.22	159
	152
0.56	191
	201
1.10	207
	200

Y désignera les valeurs des vélocités et X celles des concentrations.

1.3.2 Le modèle BOD

Ce modèle traite du besoin d'oxygène de microorganismes à partir de l'expérience de D.Marske (1967) en fonction du temps d'incubation par l'équation suivante :

$$f(x, \boldsymbol{\theta}) = \theta_1(1 - e^{-\theta_2 x})$$

Les valeurs expérimentales sont données par le tableau ci-dessous :

temps (jour)	Besoin Biochimique en Oxygène
1	8.3
2	10.3
3	19.0
4	16.0
5	15.6
7	19.8

La variable aléatoire Y représente la quantité d'oxygène utilisée et la variable explicative X le temps d'incubation en jours.

Remarque : Nous voyons que pour ces deux exemples de régression non linéaire il n'y a q'une seule variable explicative x, c'est-à-dire $k = 1$.

Chapitre 2

Estimation du paramètre θ

2.1 Etude théorique de l'estimation

2.1.1 Critère des moindres carrées

Supposons que l'on a n observations (X_i, Y_i), $i = 1...n$ d'un modèle de régression non linéaire avec fonctionnelle f :

$$Y_i = f(X_i, \theta^*) + \varepsilon_i$$

où θ^* désigne la vraie valeur de θ, paramètre inconnu. L'estimateur de θ^*, noté $\hat{\theta}$, minimise la somme des carrés des erreurs $S(\hat{\theta}) = \min_{\boldsymbol{\theta}} S(\boldsymbol{\theta})$ où

$$S(\boldsymbol{\theta}) = \sum_{i=1}^{n} (Y_i - f(X_i, \boldsymbol{\theta}))^2$$

Ainsi nous allons chercher $\hat{\theta} = \hat{\theta}(Y_1, ..., Y_n)$ l'estimateur de $\boldsymbol{\theta}$ qui minimise $S(\boldsymbol{\theta})$. Nous savons que pour une régression linéaire $\hat{\theta} = ({}^tZZ)^{-1}{}^tZY$ où Z est la matrice des dérivées partielles de f or quand il s'agit d'une régression non linéaire, il n'existe pas d'expression explicite de $\hat{\theta}$.

2.1.2 Algorithme de l'estimation

On commence par fixer une valeur initiale du paramètre θ que l'on va appeler $\theta^{(0)} = {}^t(\theta_1^{(0)}, ..., \theta_p^{(0)})$ en utilisant deux méthodes au choix :

- soit on trace la courbe des **contours** de S(θ) et on prend la valeur initiale de θ dans le contour le plus petit ou l'intersection de tous les contours

– soit on trace sur le même graphe les données expérimentales avec la courbe de la fonction $f(X,\theta)$ pour diverses valeurs de θ et on prend pour $\theta^{(0)}$, la courbe qui se rapproche le mieux de l'ensemble des valeurs expérimentales

Puis nous utilisons une méthode de résolution itérative pour arriver pas à pas à la valeur de l'estimateur $\hat{\theta}$
On part bien sûr de $\theta^{(0)}$ et on calcule :

$$\theta^{(1)} = \theta^{(0)} + \delta^{(0)}$$

$$\theta^{(2)} = \theta^{(1)} + \delta^{(1)}$$

$$\vdots$$

$$\theta^{(l+1)} = \theta^{(l)} + \delta^{(l)}$$

$\delta^{(l)}$ est appelé le **pas** de l'itération jusqu'à ce que le critère de convergence (ou test d'arrêt) suivant soit vérifié :

$$\left| \frac{\theta_i^{(l+1)} - \theta_i^{(l)}}{\theta_i^{(l)}} \right| < c$$

où c est une valeur numérique fixée (par exemple $c = 10^{-4}$)

$$\begin{aligned}
&\iff && |\theta_i^{(l+1)} - \theta_i^{(l)}|^2 < c^2 |\theta_i^{(l)}|^2 \\
&\iff && |\delta_i^{(l)}|^2 < c^2 |\theta_i^{(l)}|^2 \\
&\iff && \sum_{i=1}^{p} |\delta_i^{(l)}|^2 < c^2 \sum_{i=1}^{p} |\theta_i^{(l)}|^2 \\
&\iff && \|\delta^{(l)}\|^2 < c^2 \|\theta^{(l)}\|^2
\end{aligned}$$

2.1.3 Calcul de l'estimation du pas

Nous allons calculer la valeur de l'estimateur $\hat{\theta}$ de θ^* en utilisant l'itération pas à pas.
Dans un petit voisinage de θ^*, on a :

$$f(X,\theta) \simeq f(X,\theta^*) + Z(\theta^*)(\theta - \theta^*)$$

et

$$f(X,\hat{\theta}) \simeq f(X,\theta^*) + Z(\theta^*)(\hat{\theta} - \theta^*)$$

où $Z(\theta^*)$ est la matrice des dérivées partielles de $f(X,\theta)$ pour la valeur de θ^*.
Donc

$$\begin{aligned} S(\theta) &= \|Y - f(X,\theta)\|^2 \\ &\simeq \|Y - f(X,\theta^*) - Z(\theta^*)(\theta - \theta^*)\|^2 \end{aligned}$$

et

$$S(\hat{\theta}) \simeq \|Y - f(X,\theta^*) - Z(\theta^*)(\hat{\theta} - \theta^*)\|^2$$

On sait que pour minimiser $\|Y - f(X,\theta^*) - Z(\theta^*)(\hat{\theta} - \theta^*)\|^2$ il faut que :

$$(\hat{\theta} - \theta^*) = ({}^tZ(\theta^*)Z(\theta^*))^{-1}\,{}^tZ(\theta^*)(Y - f(X,\theta^*))$$

L'estimation $\hat{\delta^{(l)}}$ du **pas** de l'itération d'ordre l se fait en utilisant le principe des moindres carrées. Si $\boldsymbol{\theta}$ est proche de $\theta^{(l)}$ on calcule le développement limité à l'ordre 1 de $f(X_i,\theta)$ qui donne

$$f(X_i,\theta) = f(X_i,\theta^{(l)}) + \sum_{j=1}^{p} \left[\frac{\partial f(X_i,\theta)}{\partial \theta_j}\right]_{\theta=\theta^{(l)}} (\theta_j - \theta_j^{(l)}) + o(\|\theta - \theta^{(l)}\|)$$

On pose alors $f^{(l)} = (f(X_i,\theta^{(l)}))_{i=1,..,n}$, $\delta^{(l)} = \theta - \theta^{(l)}$ et

$$Z_l = \left(\left[\frac{\partial f(X_i,\theta)}{\partial \theta_j}\right]_{\theta=\theta^{(l)}} ; i \leqslant n, j \leqslant p \right)$$

De ce fait on remplace dans l'équation de départ $Y = f(X_i,\theta) + \varepsilon$ par $Y - f^{(l)} = Z_l\delta + \varepsilon$ qui est linéaire et on estime alors $\delta^{(l)}$ par $\hat{\delta^{(l)}}$ en appliquant la méthode **des moindres carrées** qui va minimiser $S(\delta^{(l)}) = \|Y - f^{(l)} - Z_l\delta^{(l)}\|^2$. On a alors

$$\begin{aligned} & {}^tZ_l(Y - f^{(l)} - Z_l\hat{\delta^{(l)}}) = 0 \\ \iff \quad & {}^tZ_l(Y - f^{(l)}) = {}^tZ_lZ_l\hat{\delta^{(l)}} \\ \iff & \hat{\delta^{(l)}} = ({}^tZ_lZ_l)^{-1}\,{}^tZ_l(Y - f^{(l)}) \end{aligned}$$

si $({}^tZ_lZ_l)$ est inversible. On obtient donc $\theta^{(l+1)} = \theta^{(l)} + \hat{\delta}^{(l)}$ puis on remplace $\theta^{(l+1)}$ à la place de $\theta^{(l)}$ pour obtenir $\hat{\delta}^{(l+1)}$ et $\theta^{(l+2)}$ ainsi de suite jusqu'à ce que le critère de convergence soit vérifié.

2.1.4 L'algorithme de NEWTON-GAUSS direct

Cet algorithme consiste donc à trouver $\hat{\theta}$ en utilisant la méthode de résolution itérative vue précedemment à partir de la matrice des dérivées Z. On fixe $\theta^{(0)} = {}^t(\theta_1^{(0)}, ..., \theta_p^{(0)})$ et on pose $\lambda_k = (-1)^k/k^2$ tel que pour k=0 $\lambda_0 = 1$. Ensuite on fixe c pour le critère de convergence et le compteur des iterations $ni = 0$. Tant que le critère de convergence n'est pas satisfait faire : $\hat{\delta} = ({}^tZ_lZ_l)^{-1}{}^tZ_l(Y - f^{(0)})$ on incrémente le compteur ni de 1 puis on évalue $\theta^1 = \theta^{(0)} + \lambda_k\hat{\delta}$.
Si $S(\theta_1) > S(\theta_0)$ alors tant que $S(\theta^{(0)} + \lambda_k\hat{\delta}) > S(\theta^{(0)})$ on incrémente k de 1 jusqu'à ce que $S(\theta^{(0)} + \lambda_k\hat{\delta}) < S(\theta^{(0)})$
sinon $\theta^{(0)} = \theta^{(0)} + \lambda_k\hat{\delta}$ et $S(\theta^{(0)}) = S(\theta^{(0)} + \lambda_k\hat{\delta})$ puis on recommence jusqu'à arriver à $\hat{\theta}$.

2.1.5 L'algorithme de NEWTON-GAUSS avec Q-R décomposition

Cet algorithme est le même que précedemment par contre il change au niveau du calcul de l'estimation $\hat{\delta}$ du pas d'itération. Nous utilisons cette fois-ci la décomposition Q-R de la matrice Z des dérivées partielles de $f(X, \theta)$; $Z = Q * R$ où Q est une matrice orthogonale et R est une matrice triangulaire supérieure.

$$\begin{aligned} Z &= QR \\ &= (Q_1Q_2)\begin{pmatrix} R_1 \\ 0 \end{pmatrix} \\ &= Q_1R_1 \end{aligned}$$

où Q_1 est une matrice orthogonale $(n \times p)$ et R_1 est une matrice triangulaire supérieure carrée $(p \times p)$. De ce fait avec cette transformation Q-R le calcul de l'estimateur $\hat{\delta}$ du pas d'itération change aussi. En partant de $\theta^{(0)}$ on a :

$$\begin{aligned} \hat{\delta} &= ({}^tZZ)^{-1}{}^tZ(Y - f^{(0)}) \\ &= ({}^tR_1{}^tQ_1Q_1R_1)^{-1}{}^tR_1{}^tQ_1(Y - f^{(0)}) \\ &= ({}^tR_1R_1)^{-1}{}^tR_1{}^tQ_1(Y - f^{(0)}) \\ &= R_1^{-1}({}^tR_1)^{-1}{}^tR_1{}^tQ_1(Y - f^{(0)}) \\ &= R_1^{-1}{}^tQ_1(Y - f^{(0)}) \end{aligned}$$

Puis on applique la même démarche que précédemment.

Conclusion : Nous allons pouvoir comparer les deux versions de l'algorithme de NEWTON-GAUSS direct et avec la décomposition Q-R pour déterminer l'estimateur $\hat{\theta}$ du paramètre θ.

2.2 Etude appliquée de l'estimation aux deux modèles.

Naturellement nous allons appliquer cette théorie du calcul de l'estimation du paramètre θ sur les deux modèles présentés Michaelis-Menten et BOD en utilisant et en comparant les deux versions de l'algorithme de NEWTON-GAUSS.

2.2.1 Estimation de θ pour le modèle de Michaelis-Menten

Nous commençons par trouver une valeur initiale $\theta^{(0)}$ qui soit proche de l'estimateur $\hat{\theta}$ pour que les algorithmes de NEWTON-GAUSS puissent être bien appliqués. Nous disposons donc de deux méthodes ; la première consiste à tracer la courbe de contours de la fonction $S(\boldsymbol{\theta}) = \sum_{i=1}^{n} (Y_i - f(X_i, \boldsymbol{\theta}))^2$ avec $f(X, \boldsymbol{\theta}) = \frac{\theta_1 X}{\theta_2 + X}$.
On obtient comme graphe :

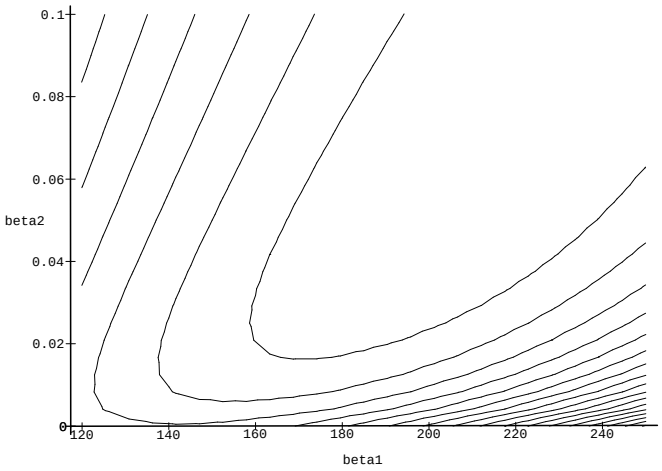

FIG. 1 - *Courbe de contours pour* $S(\theta)$.

Avec ce graphe nous prendrons les valeurs initiales $\theta^{(0)}$ dans la région délimitée par le niveau le plus bas c'est-à-dire l'intersection de toutes les régions. Pour cet exemple nous voyons que du fait que cette région est vaste on peut prendre un grand nombre de valeurs initiales.

Maintenant on trace sur le même graphe toutes les valeurs expérimentales et le graphe de $f(X, \theta)$ pour deux valeurs différentes de θ , $\theta_1 = {}^t(200, 0.05)$ et $\theta_2 = {}^t(220, 0.08)$ et nous avons alors les graphes suivants :

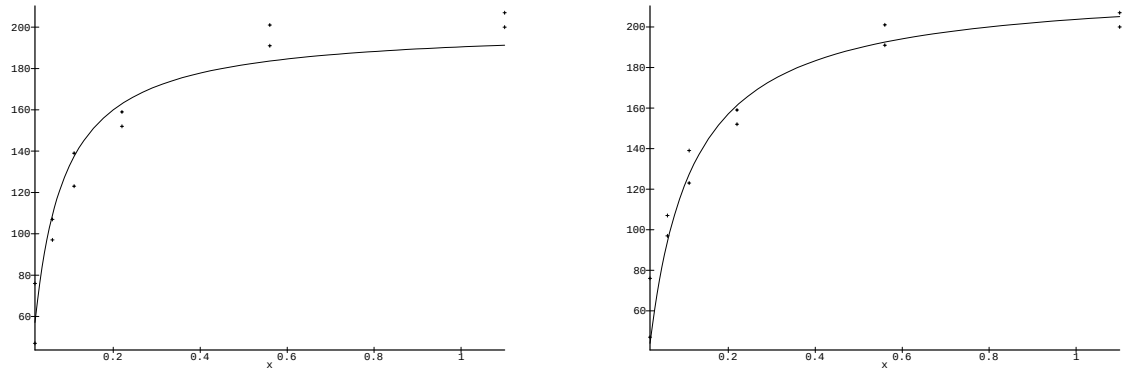

FIG. 2 - *Tracés de $f(X,\theta)$ pour θ_1 et θ_2.*

Nous voyons que cette deuxième méthode est plus précise pour trouver une bonne valeur initiale $\theta^{(0)}$ que la première où il y a un plus grand choix. Ainsi nous prendrons pour valeur initiale du paramètre θ :

$$\theta^{(0)} = {}^t(220, 0.08)$$

car elle donne un meilleur tracé que pour θ_1 .

Maintenant à partir de cette valeur initiale $\theta^{(0)}$ nous allons appliquer l'algorithme de NEWTON-GAUSS direct :
$\theta^{(0)} = {}^t(220, 0.08)$ et $S(\theta^{(0)}) = 1552.976$. On obtient pour l'estimation du pas $\hat{\delta}^{(0)} = {}^t(-6.971, -0.016)$ donc $\theta^{(1)} = {}^t(213.029, 0.064)$ et $S(\theta^{(1)}) = 1196.342$.
On continue alors et on obtient au bout de 7 itérations l'estimation de

$$\hat{\theta} = {}^t(212.684, 0.064)$$

qui minimise $S(\theta)$ avec $S(\hat{\theta}) = 1195.449$.
Voici un exemple de l'exécution de l'algorithme détaillé :

```
> newton(220,0.08,10^(-6));

                    theta[0] = [ 220, .08 ]

                      SS[0] = 1552.975533

                   c'est la premiere etape

                theta[1] = [ 213.0288945, .06405866369 ]
```

```
          SS[1] = 1196.342401

          C'est la 2eme etape

theta[2] = [ 212.6798595, .06411512228 ]

          SS[2] = 1195.448874

          C'est la 3eme etape

theta[3] = [ 212.6833621, .06412068795 ]

          SS[3] = 1195.448815

          C'est la 4eme etape

theta[4] = [ 212.6837064, .06412122443 ]

          SS[4] = 1195.448818

          C'est la 5eme etape

           Nombre de pas = 1

theta[5] = [ 212.6830178, .06412015147 ]

          SS[5] = 1195.448815

          C'est la 6eme etape

theta[6] = [ 212.6836731, .06412117267 ]

          SS[6] = 1195.448812

          C'est la 7eme etape

theta[7] = [ 212.6837365, .06412127118 ]
```

```
SS[7] = 1195.448812

Nombre total d'iterations = 7
```

Puis nous appliquons l'algorithme de NEWTON-GAUSS avec la décomposition Q-R en partant toujours de $\theta^{(0)} = {}^t(220, 0.08)$, on obtient $\hat{\delta}^{(0)} = {}^t(-6.971, -0.016)$ et $\theta^{(1)} = \theta^{(0)} + \lambda_0 \hat{\delta}^{(0)}$ d'où $\theta^{(1)} = {}^t(213.029, 0.064)$ avec $S(\theta^{(1)}) = 1196.342$. Comme $S(\theta^{(1)}) < S(\theta^{(0)})$ nous continuons l'iteration et au bout de seulement 5 itérations nous obtenons l'estimation de

$$\hat{\theta} = {}^t(212.684, 0.064)$$

avec bien sûr $S(\hat{\theta}) = 1195.449$
Exemple de l'exécution de l'algorithme à chaque itération :

```
> newton1(220,0.08,10^(-6));

                theta[0] = [ 220, .08 ]

                  SS[0] = 1552.975533

                c'est la premiere etape

          theta[1] = [ 213.0288945, .06405866369 ]

                  SS[1] = 1196.342401

                  C'est la 2eme etape

          theta[2] = [ 212.6798595, .06411512224 ]

                  SS[2] = 1195.448873

                  C'est la 3eme etape

          theta[3] = [ 212.6833621, .06412068803 ]

                  SS[3] = 1195.448818

                  C'est la 4eme etape
```

```
theta[4] = [ 212.6837064, .06412122450 ]

SS[4] = 1195.448817

C'est la 5eme etape

theta[5] = [ 212.6837397, .06412127625 ]

SS[5] = 1195.448812

Nombre total d'iterations = 5

[ 212.6837397, .06412127625 ]
```

Interpretation des résultats:
L'utilisation de la deuxième version de l'algorithme de NEWTON-GAUSS c'est-à-dire avec la décompisition Q-R fait diminuer le nombre d'itérations utiles pour arriver à l'estimateur $\hat{\theta}$ de θ. En effet il nous faut seulement 5 itérations contre 7 seulement à la première version directe de l'algorithme à partir de $\theta^{(0)} = {}^t(220, 0.08)$. Cette diminution s'explique par le fait qu'avec la décomposition Q-R l'algorithme de NEWTON-GAUSS devient plus stable et ainsi augmente la vitesse de convergence vers l'estimateur $\hat{\theta}$.

2.2.2 Estimation de θ pour le modèle BOD

On refait la même étude que pour le modèle précédent mais avec $f(X, \theta) = \theta_1(1 - e^{-\theta_2 X})$ et on part de $\theta^{(0)} = {}^t(20, 0.24)$ avec $S(\theta^{(0)}) = 128.182$. Vérifions avec le graphe des contours si nous avons une bonne valeur initiale:

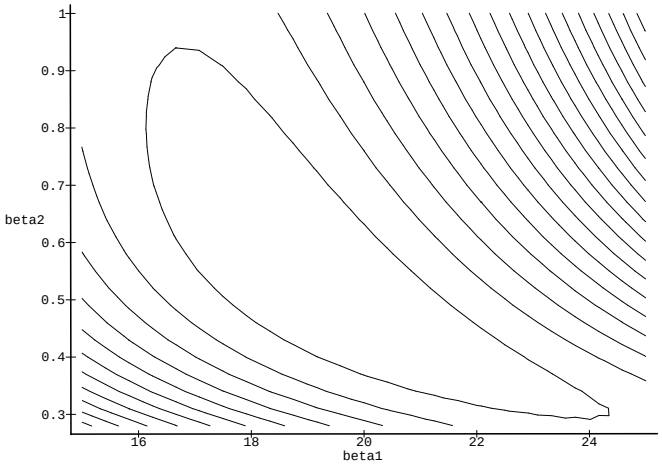

FIG. 3 - *Courbe de contours de $S(\theta)$ pour le modele BOD*

Nous voyons bien que cette valeur initiale n'est pas optimale pour l'exécution de l'algorithme de NEWTON-GAUSS car elle ne correspond pas au contour inférieur central. Ainsi à partir de cette valeur initiale donnée nous exécutons les algorithmes direct et avec Q-R décomposition. Pour le premier $\theta^{(1)} = {}^t(13.610, 0.525)$ avec $S(\theta^{(1)}) = 145.168$ et nous obtenons au bout de 6 itérations l'estimation de $\hat{\theta} = {}^t(19.143, 0.531)$ qui minimise $S(\theta)$ avec $S(\hat{\theta}) = 25.990$.
Voici un exemple de l'exécution detaillé de l'algorithme direct :

```
> newton(20,0.24,10^(-6));

                    theta[0] = [ 20, .24 ]

                    SS[0] = 128.1817643

                  c'est la premiere etape

          theta[1] = [ 13.60962868, .5245414131 ]

                    SS[1] = 145.1684594

                    C'est la 2eme etape

                     Nombre de pas = 2
```

```
theta[2] = [ 18.40240717, .3111353533 ]

SS[2] = 97.63358422

C'est la 3eme etape

theta[3] = [ 17.31704051, .5315740741 ]

SS[3] = 38.67148264

C'est la 4eme etape

theta[4] = [ 19.14248860, .5310492151 ]

SS[4] = 25.99026864

C'est la 5eme etape

theta[5] = [ 19.14258257, .5310906796 ]

SS[5] = 25.99026727

C'est la 6eme etape

theta[6] = [ 19.14257541, .5310913652 ]

SS[6] = 25.99026724

Nombre total d'iterations = 6
```

Or pour l'algorithme avec la décomposition Q-R nous observons les mêmes résultats que pour l'algorithme direct c'est-à-dire on obtient le même nombre d'itérations nécessaires 6 pour atteindre l'estimation de $\hat{\theta}$. Voici un exemple détaillé de l'exécution de l'algorithme avec la Q-R décomposition :

```
> newton1(20,0.24,10^(-6));

theta[0] = [ 20, .24 ]
```

```
          SS[0] = 128.1817643

        c'est la premiere etape

theta[1] = [ 13.60962873, .5245414139 ]

          SS[1] = 145.1684569

          C'est la 2eme etape

           Nombre de pas = 2

theta[2] = [ 18.40240718, .3111353535 ]

          SS[2] = 97.63358382

          C'est la 3eme etape

theta[3] = [ 17.31704054, .5315740732 ]

          SS[3] = 38.67148232

          C'est la 4eme etape

theta[4] = [ 19.14248860, .5310492153 ]

          SS[4] = 25.99026863

          C'est la 5eme etape

theta[5] = [ 19.14258257, .5310906791 ]

          SS[5] = 25.99026728

          C'est la 6eme etape

theta[6] = [ 19.14257540, .5310913657 ]

          SS[6] = 25.99026727
```

```
Nombre total d'iterations = 6

    [ 19.14257540, .5310913657 ]
```

Chapitre 3

Les régions de confiance du paramètre θ

3.1 Etude générale des régions de confiance

Le but principal de la construction des régions de confiance est de trouver pour un niveau $1-\alpha$ de confiance donné ($\alpha \in]0,1[$) une région $R(\hat{\theta},\alpha) \subset \mathbb{R}^p$ dépendant de θ et de α telle que

$$P\{\theta \in R(\hat{\theta},\alpha)\} = 1-\alpha$$

Cette construction nécessite de poser une hypothèse supplémentaire pour le vecteur $\boldsymbol{\varepsilon} = {}^t(\varepsilon_1,...,\varepsilon_n)$; $\boldsymbol{\varepsilon}$ suit une loi $\mathcal{N}(0,\sigma^2 I_n)$.
La grosse difficulté de cette étude est qu'il n'est pas aisé d'obtenir des régions de confiance exactes car en général on ne connaît pas la loi exacte de $\hat{\theta}$. De ce fait on recherchera le plus souvent des régions de confiance **approchées**. Ainsi nous voyons ici trois types de régions de confiance approchées :

- la première fondée sur une approximation de la loi de $S(\hat{\theta})$ que nous nommons **région de confiance approximative I**
- la seconde basée sur l'approximation linéaire de $f(X,\hat{\theta})$ autour de $\hat{\theta}$ que nous appellons **région de confiance approximative II**
- la dernière, elle est fondée sur l'approximation de **Bayes** nommée **région de confiance bayesienne**

3.1.1 La région de confiance approximative I

A partir de la nouvelle hypothèse, on sait que $S(\hat{\theta})$ suit une loi de $\sigma^2\chi^2$ à (n-p) degrés de liberté. Notons θ^* la vraie valeur de θ inconnu. Dans un petit voisinage de θ^*, on a :

$$f(X_i,\theta) \simeq f(X_i,\theta^*) + \sum_{r=1}^{p} \left[\frac{\partial f(X_i,\theta)}{\partial \theta_r}\right]_{\theta=\theta^*} (\theta_r - \theta_r^*)$$

où

$$f(X,\theta) \simeq f(X,\theta^*) + Z(\theta^*)(\theta - \theta^*)$$

$$\begin{aligned} S(\hat{\theta}) &= \|Y - f(X,\hat{\theta})\|^2 \\ &\simeq \|Y - f(X,\theta^*) - Z(\theta^*)(\hat{\theta} - \theta^*)\|^2 \\ &\simeq \| \left(I_n - Z(\theta^*)({}^tZ(\theta^*)Z(\theta^*))^{-1}{}^tZ(\theta^*)\right)\varepsilon\|^2 \end{aligned}$$

$\frac{S(\hat{\theta})}{\sigma^2}$ suit approximativement une loi de χ^2 à (n-p) degrés de liberté.

$$\begin{aligned} S(\theta^*) - S(\hat{\theta}) &= \|Y - f(X,\theta^*)\|^2 - \|Y - f(X,\hat{\theta})\|^2 \\ &\simeq \|\varepsilon\|^2 - \| \left(I_n - Z({}^tZZ)^{-1}{}^tZ\right)\varepsilon\|^2 \\ &\simeq \| \left(Z({}^tZZ)^{-1}{}^tZ\right)\varepsilon\|^2 \end{aligned}$$

$\frac{S(\theta^*)-S(\hat{\theta})}{\sigma^2}$ suit approximativement une loi de χ^2 à p degrés de liberté. Donc $S(\hat{\theta})$ et $S(\theta^*)-S(\hat{\theta})$ sont approximativement indépendants. De ce fait $S(\theta)-S(\hat{\theta})$ suit une loi de $\sigma^2\chi^2$ à p degrés de liberté et comme $S(\theta)-S(\hat{\theta})$ et $S(\hat{\theta})$ sont approximativement indépendantes , alors

$$\frac{(S(\theta) - S(\hat{\theta}))(n-p)}{S(\hat{\theta})p}$$

suit approximativement une loi de **Fisher** à (p,n-p) degrés de liberté. D'où la définition de cette région au niveau de confiance $1-\alpha$:

$$\left\{\theta : S(\theta) \leqslant S(\hat{\theta})\left[1 + \frac{p}{n-p}F_{p,n-p}(\alpha)\right]\right\}$$

3.1.2 La région de confiance approximative II

Elle utilise l'approximation linéaire suivante de $f(X,\hat{\theta})$:

$$f(X,\hat{\theta}) \simeq f(X,\theta^*) + Z(\hat{\theta} - \theta^*)$$

et bien sûr

$$Z = \left(\left[\frac{\partial f(X_i, \theta)}{\partial \theta_j} \right]_{\theta=\theta^*} ; i \leqslant n, j \leqslant p \right)$$

ε suivant une loi $\mathcal{N}(0, \sigma^2 I_n)$, ${}^t(\theta - \hat{\theta}){}^tZZ(\theta - \hat{\theta})$ suit une loi de $\sigma^2\chi^2$ à p degrés de liberté et $S(\hat{\theta})$ suit elle approximativement une loi de $\sigma^2\chi^2$ à (n-p) degrés de liberté. Comme ${}^t(\theta - \hat{\theta}){}^tZZ(\theta - \hat{\theta})$ et $S(\hat{\theta})$ sont indépendants alors

$$\frac{{}^t(\theta - \hat{\theta}){}^tZZ(\theta - \hat{\theta})(n-p)}{S(\hat{\theta})p}$$

suit approximativement une loi de **Fisher** à (p,n-p) degrés de liberté. D'où la définition de la région de confiance approximative II :

$$\left\{ \theta : {}^t(\theta - \hat{\theta}){}^tZZ(\theta - \hat{\theta}) \leqslant S(\hat{\theta}) \frac{p}{n-p} F_{p,n-p}(\alpha) \right\}$$

3.1.3 La région de confiance bayesienne

Cette région de confiance est donnée par l'approximation de **Bayes** dont nous vous épargnons les détails. On se borne donc à donner ici la définition de la région de confiance bayesienne :

$$\left\{ \theta : \frac{S(\theta)}{|{}^tZ(\theta)Z(\theta)|^{1/n}} \leqslant \frac{S(\hat{\theta})}{|{}^tZ(\hat{\theta})Z(\hat{\theta})|^{1/n}} \left[1 + \frac{p}{n-p} F_{p,n-p}(\alpha) \right] \right\}$$

3.1.4 La région de confiance exacte

Enfin nous comparons les trois régions de confiance approximatives avec cette dernière qui a pour définition :

$$\left\{ \theta : \frac{\|{}^tQ_1(Y - f(X,\theta))\|^2/p}{\|{}^tQ_2(Y - f(X,\theta))\|^2/n-p} \leqslant F_{p,n-p}(\alpha) \right\}$$

où Q_1 est la sous-matrice $(n \times p)$ et Q_2 la sous-matrice $(n \times n-p)$ de la matrice Q de la décomposition Q-R de Z.

Précision:$F_{p,n-p}(\alpha)$ désigne le quantile d'ordre $(1-\alpha)$ de la distribution de **Fisher-Snédécor** à (p,n-p) degrés de liberté.

3.2 Régions de confiance pour les deux modèles

Nous allons maintenant étudier les quatre régions de confiance vues précedemment pour les modèles de Michaelis-Menten et de BOD.
Ainsi pour chaque modèle nous traçons les différentes régions de confiance pour un niveau de confiance déterminé.

3.2.1 Régions de confiance pour le modèle de Michaelis-Menten

Nous commençons par tracer la première région de confiance approximative au niveau de confiance de 95 %.
Nous obtenons comme graphe :

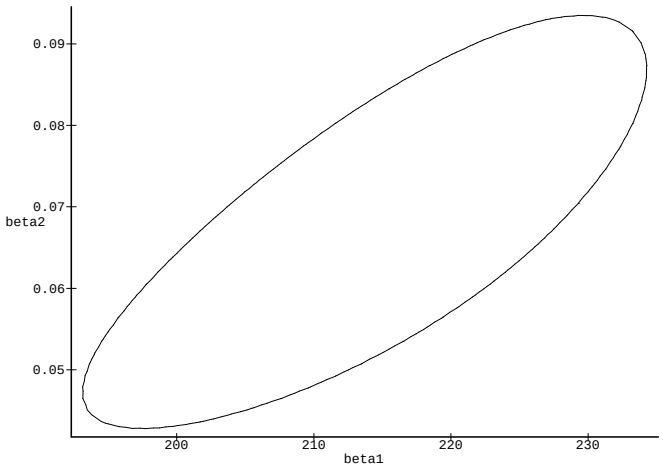

FIG. 4 - *Région de confiance approximative I au niveau de confiance de 95 %.*

Puis nous traçons la deuxième région de confiance au même niveau de confiance :

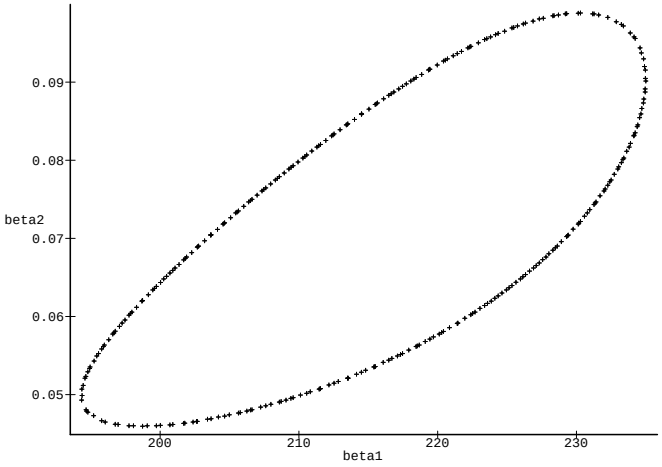

FIG. 5 - *Région de confiance approximative II au niveau de confiance de 95 %.*

Pour la région de confiance bayesienne nous obtenons :

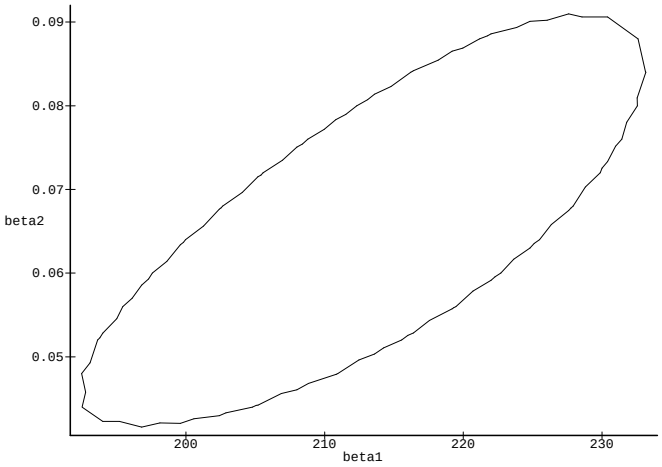

FIG. 6 - *Région de confiance bayesienne au niveau de confiance de 95 %.*

Interprétation des graphes : Nous obtenons de ces graphiques des $(1-\alpha)$-régions de confiance approximatives qui ont toutes la même forme **ellipsoïdale** et sont pratiquement de même taille. Ainsi on a fait la comparaison en traçant les trois régions de confiances approximatives sur le même graphe :

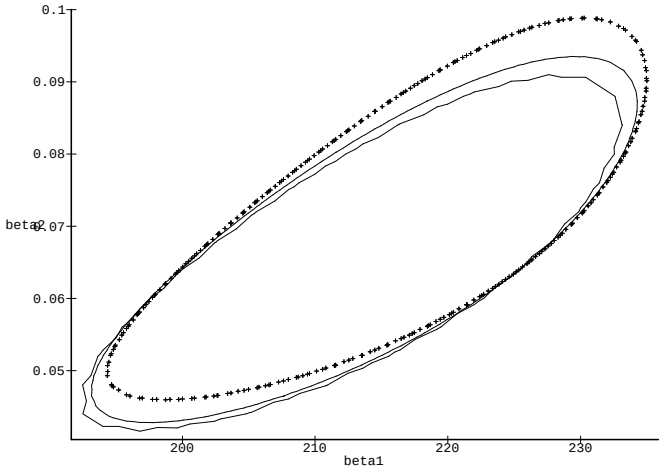

FIG. 7 - *Différentes régions de confiance approximatives au niveau de confiance de 95 %.*

Puis nous étudions graphiquement la variation de l'estimation de θ par l'algorithme de NEWTON-GAUSS par rapport à la région de confiance approximative I avec différents niveaux de confiance.

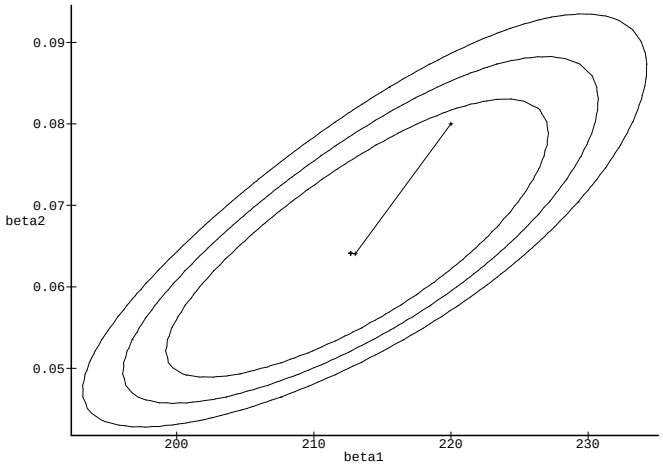

FIG. 8 - *Variation de l'estimation de θ par rapport aux différentes régions approximatives I.*

3.2.2 Régions de confiance pour le modèle BOD

Nous refaisons exactement la même chose que pour le modèle précedent en traçant les différentes régions de confiance au niveau de confiance unique de 90 %. On obtient alors les graphes suivants :

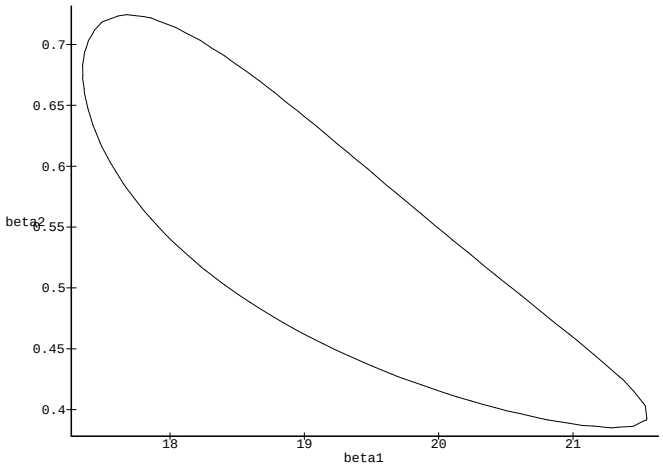

FIG. 9 - *Région de confiance approximative I au niveau de confiance de 90 %.*

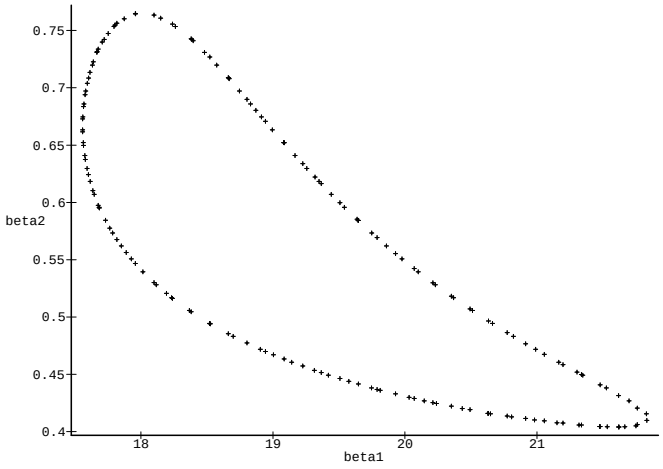

FIG. 10 - *Région de confiance approximative II au niveau de confiance de 90 %.*

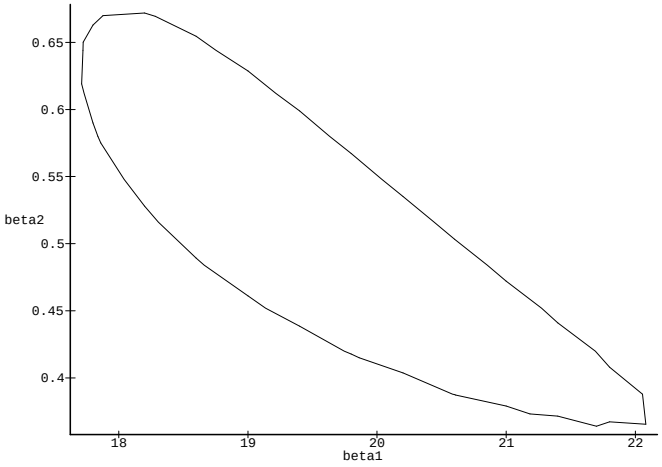

FIG. 11 - *Région de confiance bayesienne au niveau de confiance de 90 %.*

Nous allons ainsi comparer ces différentes régions de confiance en les traçant sur un même graphe intitulé **comparaison des trois différentes régions approximatives** :

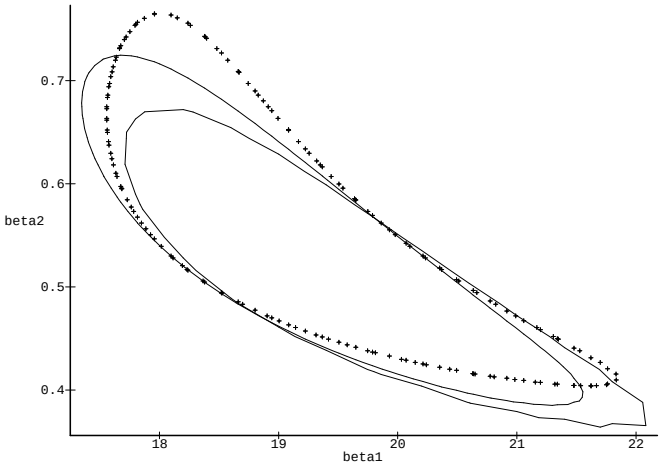

FIG. 12 - *les trois Régions de confiance approchées au niveau de confiance de 90 %.*

Commentaire des graphiques :
Nous voyons ainsi par ce graphe de comparaison que les trois régions de confiance approximatives sont bien différentes. Cela s'explique par le fait qu'il y a deux sens

à l'approximation des régions de confiance. Nous avons les régions de confiance où l'approximation est qualitative c'est-à-dire le niveau de confiance est exact mais la région ainsi définie n'est pas précise. Par contre une approximation du niveau de confiance va nous donner une région assez précise mais dont le niveau de confiance ne sera pas exact.

Puis nous voyons graphiquement la variation de l'estimation de θ par l'algorithme de NEWTON-GAUSS par rapport à la région de confiance approximative I tracée précedemment.

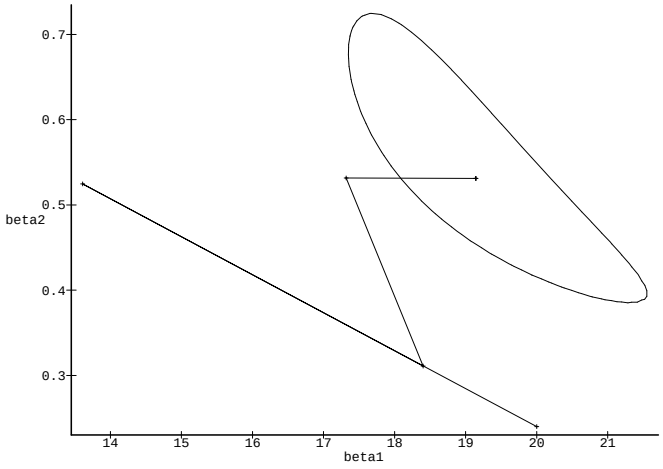

FIG. 13 - *Variation de l'estimation de θ par rapport la région approximative I de niveau de confiance 90 %.*

Chapitre 4

Programmation

Dans cette partie nous voyons le programme en **Maple** qui traite par les deux modèles présentés l'estimation du paramètre θ et bien sûr la construction de la région de confiance de ce dernier. Le programme comporte trois grandes parties de programmation :

- la première partie appelée **traitement des données et fonction de régression**
- la seconde partie nommée **programme principal de régression** qui est la plus grande partie de la programmation
- et enfin la dernière appelée **décomposition Q-R** qui va permettre la décomposition en matrice orthogonale et triangulaire supérieure de la matrice des dérivées partielles Z de $f(X,\boldsymbol{\theta})$.

4.1 Le programme traitement des données et fonction de régression

Cette première partie comporte très peu de procédures et elle est propre à chaque modèle de régression. Elle sert notammment à définir la taille de l'échantillon **n** à partir d'un fichier DONNÉES et les vecteurs $\boldsymbol{Y} = {}^t(Y_1, ..., Y_n)$ et $\boldsymbol{X} = {}^t(X_1, ..., X_n)$ à partir des n observations que l'on dispose. Puis enfin dans un autre fichier FONCTION on va définir la fonction de régression $f(X,\boldsymbol{\theta})$ sous forme procédurale et déclarer le paramètre $\boldsymbol{\theta}$ sous forme de vecteur et ainsi obtenir sa taille **p**.

4.2 Le programme principal

C'est la partie la plus importante de la programmation car on y trouve les plus importantes procédures qui permettent l'étude de la régression non linéaire. Les procédures sont réparties en trois catégories :

- les procédures qui exécutent **l'estimation par itération simple**
- les procédures qui permettent **l'estimation par itération avec la Q-R décomposition**
- et enfin les procédures qui vont tracer **les différentes régions de confiance**

Avant tout cela nous avons la procédure **A** qui retourne la matrice des dérivées partielles Z de la fonction $f(X,\boldsymbol{\theta})$, la procédure **df** qui donne l'élément Z_{ij} de la matrice Z, la procédure **SS** qui calcule la somme des carrés des erreurs et enfin la procédure **Gcontour** qui trace la courbe des contours de **SS**.

4.2.1 Les procédures de l'estimation simple

Nous avons deux procédures pour l'estimation par itération simple :
la procédure **direction** qui va calculer l'estimation du pas de l'itération à partir de la matrice des dérivées partielles Z de $f(X,\boldsymbol{\theta})$. Puis on a la grande procédure **newton** qui va exécuter l'algorithme de NEWTON-GAUSS direct et qui retourne l'estimation de $\hat{\theta}$ appelée **valeur_estimation**.

4.2.2 Les procédures de l'estimation avec Q-R décomposition

Elles sont aussi au nombre de deux et sont similaires aux deux procédures de l'estimation simple mais cette fois-ci elles font appel à la décomposition Q-R de la matrice Z des dérivées partielles de $f(X,\boldsymbol{\theta})$. On a donc la procédure **direction1** et la procédure **newton1** qui vont exécuter l'algorithme de NEWTON-GAUSS avec la Q-R décomposition.

4.2.3 Les procédures de construction des régions de confiance

Nous disposons dans le programme principal de quatre procédures qui permettent la construction de trois régions de confiance approximatives et bien sûr la région de confiance exacte. On pose au préalable le nombre de composantes paramétrées de θ à **deux**. Ainsi on tracera les régions de confiance en fonction de ces deux

composantes inconnues. De plus on utilisera la valeur de l'estimation de θ obtenue par les procédures **newton** ou **newton1**.
Nous avons alors :

- la procédure **RcAppro1** qui va construire la région de confiance approximative I à un niveau de confiance $1-\alpha$ donné
- la procédure **RcAppro2** qui donnera la région de confiance approximative II à un seuil de confiance α donné
- la procédure **RcBayes** à partir d'un niveau $1-\alpha$ de confiance donné nous construira la région de confiance bayesienne
- et enfin la procédure **RcExacte** permettra de construire la région de confiance exacte au niveau de confiance $1-\alpha$ fixé.

Les procédures **g, rapport, Rapp** permettent de définir les fonctions respectives définissant la région de confiance II, la région bayesienne et la région de confiance exacte.

4.3 La décomposition Q-R

Cette dernière partie de la programmation traite uniquement de la décomposition de la matrice des dérivées partielles Z de $f(X,\boldsymbol{\theta})$ en une matrice orthogonale Q et une matrice triangulaire supérieure R. La procédure **v** va nous donner le vecteur de la k-ième colonne d'une matrice en annulant les (k-1) premières coordonnées de ce vecteur pour l'application de la méthode de Householder. Puis nous avons la procédure **Rdec** qui calcule en utilisant la procédure précédente la matrice R. La matrice orthogonale Q est donnée par la procédure **Qdec** en utilisant toujours la procédure **v**. Puis nous avons les procédures **Rdec1, Qdec1** et **Qdec2** qui donnent respectivement la sous-matrice carrée $(p \times p)$ R_1 de R, la sous-matrice $(n \times p)$ Q_1 et la sous-matrice $(n \times n-p)$ Q_2 de Q.

Les procédures **QDEC, QDEC1** et **QDEC2** ont exactement les mêmes finalités que **Qdec, Qdec1** et **Qdec2** sauf qu'elles sont particulièrement destinées au calcul formel pour établir la région de confiance exacte.

Conclusion générale

Cette étude de la **régression non linéaire** a été centrée sur deux de ses aspects que sont **l'estimation** du paramètre θ et la construction des **régions de confiance** pour ce dernier. Nous n'avons pas aborder sur la régression non linéaire le problème des **prévisions** pour $f(X, \boldsymbol{\theta})$ à partir d'un vecteur X fixé et celui de l'étude des **résidus** pour mesurer l'écart entre le modèle et les observations. De plus nous n'avons utiliser que la méthode de NEWTON-GAUSS avec deux versions différentes (directe et avec la Q-R décomposition) pour l'estimation de θ.

Ainsi avec l'utilisation du logiciel scientifique **Maple** nous avons obtenu de bons résultats pour l'estimation de θ à partir de l'algorithme de NEWTON-GAUSS. Il en ressort que pour un modèle de régression proche d'un modèle linéaire (comme le modèle de Michaelis-Menten) l'utilisation de l'algorithme de NEWTON-GAUSS avec la décomposition Q-R permet d'accroître la vitesse de convergence de l'algorithme vers la valeur de l'estimateur $\hat{\theta}$. Cela est dû au fait de la plus grande stabilité de calcul de l'algorithme de NEWTON-GAUSS avec la décomposition Q-R. Par contre avec **Maple** nous obtenons pour la construction des régions de confiance pour θ des résultats moyens. Car même si nous avons obtenu les différentes régions de confiance approximatives, **Maple** n'a pas pu permettre la construction de la région de confiance **exacte** qui est essentielle pour l'étude et l'interprétation des régions de confiance. Ainsi une comparaison entre la région de confiance exacte et les différentes régions approximatives nous aurait permis de tirer des renseignements précis sur les modèles de régression traités. Ce que ne permettent évidemment pas les seules régions de confiance approximatives qui manquent de précision soit par la qualité de la région définie soit par le niveau de confiance déterminé.

Ainsi le programme en **Maple** qui a été développé, permet à partir d'un fichier de données et une fonction de régression f de résoudre le problème de l'estimation du paramètre θ avec l'algorithme de NEWTON-GAUSS (avec et sans Q-R décomposition) pour n'importe quel modèle de régression non linéaire régulier. Par contre, il faudra attendre une version améliorée et plus puissante en calcul formel de **Maple** pour que ce programme résoud le problème de la région de confiance exacte d'un modèle de régression non linéaire. Néanmoins, on pourra ajouter une extension à

ce programme qui permettra l'estimation du paramètre θ par d'autres méthodes par exemple ou la résolution d'autres problèmes posés par la régression non linéaire comme les prévisions pour la fonction f et l'étude des résidus pour mesurer l'écart entre le modèle et les observations.

Annexe

Dans cette partie nous donnons le listing du programme en **Maple** que nous avons utilisé et détaillé précedemment.

Traitement des données et fonction de régression

Le modèle de Michaelis-Menten

Pour ce modèle, nous partons du fichier DONNEES suivant que nous allons traiter :

```
0.02 76
0.02 47
0.06 97
0.06 107
0.11 123
0.11 139
0.22 159
0.22 152
0.56 191
0.56 201
1.10 207
1.10 200
```

Puis à partir de ce fichier DONNEES nous avons le programme traitement des données qui va définir les deux variables Y et X et bien sûr la dimension de l'échantillon n :

```
restart:
with(linalg):
with(stats):

#Lecture des donnees
```

```
readlib(readdata):
Lista:=readdata(‘donnees‘,2):
mat:=convert(Lista,matrix):
x:=linalg[col](mat,1):
y:=linalg[col](mat,2):
#Taille de l'echantillon
n:=linalg[rowdim](mat):
```

Ensuite nous avons le fichier FONCTION qui déclare la fonction de régression $f(X,\boldsymbol{\theta})$ et le paramètre θ pour nous donner la dimension p de ce dernier :

```
with(linalg):

#Definition de la fonction de la regression
f:=proc(x,theta)
theta[1]*x/(theta[2]+x):
end:

#Definition du parametre theta
theta:=vector(2):
p:=linalg[vectdim](theta):
```

Le modèle de régression BOD

Nous procédons de la même manière que pour le modèle précedent à partir du fichier DONNÉES2 :

```
1 8.3
2 10.3
3 19.0
4 16.0
5 15.6
7 19.8
```

Puis nous avons le fichier TRAITEMENT2 qui va traiter les données :

```
restart:
with(linalg):
with(stats):

#Lecture des donnees
```

```
readlib(readdata):
Lista:=readdata(‘donnees2‘,2):
mat:=convert(Lista,matrix):
x:=linalg[col](mat,1):
y:=linalg[col](mat,2):
#Taille de l'echantillon
n:=linalg[rowdim](mat):
```

Et enfin le fichier FONCTION2 :

```
with(linalg):


#declaration du parametre theta
theta:=array(1..2):

#definition de la fonction de la regression
f:=proc(x,theta)
theta[1]*(1-exp(-theta[2]*x));
end:

#dimension du vecteur theta
p:=vectdim(theta):
```

Le programme décomposition Q-R

```
#read regression
with(linalg):
Id:=array(identity,1..n,1..n,[]):

#Definir la direction
v:=proc(k,B)
local u,i:
u:=col(B,k):
for i from 1 to k-1 do
u[i]:=0:
od:
RETURN(u);
end:
```

```
#Calcul de la matrice R
Rdec:=proc(beta)
local w,u,u1,k,H,H1,R:
w:=col(A(beta),1):
if w[1]>0 then
u:=evalm(w+(norm(w,2)*col(Id,1)))
else
u:=evalm(w-(norm(w,2)*col(Id,1)))
fi:
H:=evalm(Id-((2*u &* transpose(u))/norm(u,2)^2)):
R:=evalm(H &* A(beta)):
for k from 2 to p do
if col(R,k)[k]>0 then
u1:=evalm(v(k,R)+(norm(v(k,R),2)*col(Id,k)))
else
u1:=evalm(v(k,R)-(norm(v(k,R),2)*col(Id,k)))
fi:
H1:=evalm(Id-((2*u1 &* transpose(u1))/norm(u1,2)^2)):
R:=evalm(H1 &* R):
od:
RETURN(eval(evalm(R)));
end:

#Calcul de la matrice Q
Qdec:=proc(beta)
local w,u,u1,k,H,H1,Q:
w:=col(A(beta),1):
if w[1]>0 then
u:=evalm(w+(norm(w,2)*col(Id,1)))
else
u:=evalm(w-(norm(w,2)*col(Id,1)))
fi:
H:=evalm(Id-((2*u &* transpose(u))/norm(u,2)^2)):
R:=evalm(H &* A(beta)):
Q:=H:
for k from 2 to p do
if col(R,k)[k]>0 then
u1:=evalm(v(k,R)+(norm(v(k,R),2)*col(Id,k)))
```

```
else
u1:=evalm(v(k,R)-(norm(v(k,R),2)*col(Id,k)))
fi:
H1:=evalm(Id-((2*u1 &* transpose(u1))/norm(u1,2)^2)):
R:=evalm(H1 &* R):
Q:=evalm(Q &* H1):
od:
RETURN(eval(evalm(Q)));
end:

#Sous-matrice de R
Rdec1:=proc(beta)
RETURN(submatrix(Rdec(beta),1..p,1..p));
end:

#Sous-matrice de Q
Qdec1:=proc(beta)
RETURN(submatrix(Qdec(beta),1..n,1..p));
end:

#Sous-matrice de Q
Qdec2:=proc(beta)
RETURN(submatrix(Qdec(beta),1..n,p+1..n));
end:

#Calcul de la matrice Q pour la region de confiance exacte
QDEC:=proc(beta)
local w,u,u1,k,H,H1,Q,b:
w:=col(A(beta),1):
u:=evalm(w-(norm(w,2)*col(Id,1))):
H:=evalm(Id-((2*u &* transpose(u))/norm(u,2)^2)):
R:=evalm(H &* A(beta)):
Q:=H:
for k from 2 to p do
u1:=evalm(v(k,R)-(norm(v(k,R),2)*col(Id,k))):
H1:=evalm(Id-((2*u1 &* transpose(u1))/norm(u1,2)^2)):
R:=evalm(H1 &* R):
Q:=evalm(Q &* H1):
od:
```

```
RETURN(eval(evalm(Q)));
end:

QDEC1:=proc(beta)
RETURN(submatrix(QDEC(beta),1..n,1..p));
end:

QDEC2:=proc(beta)
RETURN(submatrix(QDEC(beta),1..n,p+1..n));
end:
```

Le programme principal de regression

```
#                        REGRESSION NON LINEAIRE
#Lecture des donnees et de la fonction de regression
read traitement:
read fonction:

#Calcul de la matrice des derivees partielles de la fonction de regression
df:=proc(i,j,beta)
local d:
d:=diff(f(x[i],var_theta),var_theta[j]):
eval(subs(var_theta=beta,d)):
end:

A:=proc(beta)
local B,i,j:
B:=array(1..n,1..p):
for i to n do
for j to p do
B[i,j]:=df(i,j,beta)
od
od:
RETURN(eval(B));
end:
#Calcul de la somme des carres des erreurs
SS:=proc(theta)
local i:
```

```
sum((y[i]-f(x[i],theta))^2,i=1..n):
end:

#Courbe des contours de SS
Gcontour:=proc(deltax,deltay)
local beta:
beta:=[beta1,beta2]:
contourplot(SS(beta),beta1=deltax,beta2=deltay,axes=normal);
end:

#ESTIMATION DU PARAMETRE THETA
#Estimation par iteration
direction:=proc(beta)
local M,Mat,resi,omega,i:
resi:=array(1..n):for i to n do resi[i]:=y[i]-f(x[i],beta) od:
Mat:=evalm(transpose(A(beta)) &* A(beta)):
M:=evalm(inverse(Mat) &* transpose(A(beta))):
omega:=evalm(M &* resi):
RETURN(eval(omega));
end:

newton:=proc(x,y,epsilon)
local theta0,theta1,delta,ni,k:
theta0:=vector(2,[x,y]):
print(`theta[0]`=evalm(theta0)):
print(`SS`[0]=SS(theta0)):
print(`c'est la premiere etape`):
delta:=direction(theta0):
theta1:=evalm(theta0+delta):
print(`theta[1]`=evalm(theta1)):
print(`SS`[1]=SS(theta1)):
ni:=1:
while (norm(delta,2)^2 > epsilon^2*norm(theta0,2)^2 and ni< 100) do
ni:=ni+1:
print(`C'est la ` .ni.`eme etape`):
if ni>180 then ERROR(`Exces d'itérations (nb>180)`)
elif (SS(theta1)>SS(theta0)) then
k:=0:
while (SS(theta1)>SS(theta0)) do
```

```
k:=k+1:
theta1:=evalm(theta0+((-1)^k/k^2)*delta):
od:
print(`Nombre de pas`=k):
print(`theta[`.ni.`]`=evalm(theta1));
print(`SS`[ni]=SS(theta1)):
else
theta0:=eval(theta1):
delta:=direction(theta0):
theta1:=evalm(theta0+delta):
print(`theta[`.ni.`]`=evalm(theta1)):
print(`SS`[ni]=SS(theta1)):
fi
od:
print(`Nombre total d'iterations`=ni):
RETURN(evalm(theta1)):
end:

#Estimation par Q-R decomposition
read decomatrice:

direction1:=proc(beta)
local Mat,resi,omega,i:
resi:=array(1..n):for i to n do resi[i]:=y[i]-f(x[i],beta) od:
Mat:=evalm(inverse(Rdec1(beta)) &* transpose(Qdec1(beta))):
omega:=evalm(Mat &* resi):
RETURN(eval(omega));
end:

newton1:=proc(x,y,epsilon)
local theta0,theta1,delta,ni,k:
theta0:=vector(2,[x,y]):
print(`theta[0]`=evalm(theta0)):
print(`SS`[0]=SS(theta0)):
print(`c'est la premiere etape`):
delta:=direction1(theta0):
theta1:=evalm(theta0+delta):
print(`theta[1]`=evalm(theta1)):
print(`SS`[1]=SS(theta1)):
```

```
ni:=1:
while (norm(delta,2)^2 > epsilon^2*norm(theta0,2)^2 and ni< 100) do
ni:=ni+1:
print(‘C’est la ‘ .ni.‘eme etape‘):
if ni>180 then ERROR(‘Exces d’itérations (nb>180)‘)
elif (SS(theta1)>SS(theta0)) then
k:=0:
while (SS(theta1)>SS(theta0)) do
k:=k+1:
theta1:=evalm(theta0+((-1)^k/k^2)*delta):
od:
print(‘Nombre de pas‘=k):
print(‘theta[‘.ni.‘]‘=evalm(theta1));
print(‘SS‘[ni]=SS(theta1)):
else
theta0:=eval(theta1):
delta:=direction1(theta0):
theta1:=evalm(theta0+delta):
print(‘theta[‘.ni.‘]‘=evalm(theta1)):
print(‘SS‘[ni]=SS(theta1)):
fi
od:
print(‘Nombre total d’iterations‘=ni):
RETURN(evalm(theta1)):
end:



#REGIONS DE CONFIANCE
#with(plots):


beta:=[beta1,beta2]:
#Mettre des valeurs reelles dans newton1
Estimation:=evalm(newton1(x,y,epsilon)):

#Region de confiance par la methode de la vraisemblace I
RcAppro1:=proc(niveau,deltax,deltay)
local const1,const2:
```

```
const1:=SS(Estimation)*(1+(Fdist(niveau,p,n-p)*p/(n-p))):
implicitplot(SS(beta)=const1,beta1=deltax,beta2=deltay,numpoints=5000);
end:

g:=proc(beta)
local a:
a:=evalm(beta-Estimation):
evalm(transpose(a) &* transpose(A(beta)) &* A(beta) &* a):
end:

#Region de confiance par la methode de la vraisemblace II
RcAppro2:=proc(niveau,deltax,deltay)
local const1,const2:
const1:=evalf(SS(Estimation)*Fdist(niveau,p,n-p)*p/(n-p)):
implicitplot(g(beta)=const1,beta1=deltax,beta2=deltay,numpoints=5000,style=point);
end:

rapport:=proc(beta)
local d,m:
m:=evalm(transpose(A(beta)) &* A(beta)):
d:=(det(m))^(1/n):
RETURN(SS(beta)/d);
end:

#Region de confiance par la methode bayesienne
RcBayes:=proc(niveau,deltax,deltay)
local const1,const2:
const1:=e(Estimation)*(1+(Fdist(niveau,p,n-p)*p/(n-p))):
implicitplot(rapport(beta)=const1,beta1=deltax,beta2=deltay,color=grey);
end:

Rapp:=proc(beta)
local u1,u2,resi,n1,n2,i,j:
resi:=array(1..n):for i to n do resi[i]:=y[i]-f(x[i],beta) od:
u1:=evalm(multiply(transpose(Qr1(beta)),resi)):
u2:=evalm(multiply(transpose(Qr2(beta)),resi)):
n1:=norm(u1,2)^2:
n2:=norm(u2,2)^2:
RETURN(evalf((n1*(n-p))/(n2*p)));
```

```
end:

#Region de confiance exacte
RcExacte:=proc(niveau,deltax,deltay)
local const:
const:=evalf(Fdist(niveau,p,n-p)):
implicitplot(Rapp(beta)=const,beta1=deltax,beta2=deltay,title=‘region de
confiance exacte‘);
end:

#Courbe des trois differentes regions de confiance approximatives
comparaison:=proc(niveau,deltax,deltay)
local l,p,r:
l:=Rcapp(niveau,deltax,deltay):
p:=Rcapp1(niveau,deltax,deltay):
r:=Rcbay(niveau,deltax,deltay):
display([l,p,r],title=‘comparaison des trois differentes region
 de confiance approchees‘);
end:
```

FIN

Bibliographie

Voici une liste de quelques réferences pour une étude plus complète de la régression non linéaire.

1. Y.BARD :*Nonlinear parameter estimation* . Academic Press 1974
2. S.HUET, E.JOLIVET & A.MESSÉAN :*La régression non linéaire : méthodes et applications en biologie*. INRA Editions 1992
3. A.MONFORT : *Cours Statistique mathématiques*.2^e édition Economica 1982
4. G.A.F.SEBER & C.J.WILD : *Nonlinear regression*. Wiley 1989
5. R.TOMASSONE, E.LESQUEROY & C.MILLIER :*La régression, nouveaux regards sur une ancienne méthode statistique*. Masson 1983