

Estimation adaptative d'une fonction de transfert à deux termes avec erreurs sur les entrées

Thierry LEVIANDIER (1)

RÉSUMÉ

On établit un algorithme adaptatif d'estimation d'une fonction de transfert à deux termes lorsque les entrées comportent des erreurs, cas présumé fréquent sur des pluies servant à calculer un débit. On développe pour cela les dérivées des paramètres par rapport aux fonctions d'autocorrélation et d'autocorrélation croisée des variables. L'estimateur est testé par une méthode de Monte Carlo contre une régression multiple récursive, lorsque le vrai modèle est l'un ou l'autre. On montre sur un exemple que l'erreur d'hypothèse de modélisation des erreurs augmente le risque de mauvaise prévision sans trop affecter la qualité moyenne des résultats.

MOTS-CLÉS : Fonction de transfert – Erreurs d'entrée – Algorithme adaptatif.

ABSTRACT

ADAPTIVE ESTIMATION OF A TWO LAG TRANSFER FUNCTION WITH ERRORS IN THE INPUT DATA

An adaptive algorithm for estimating a two lag transfer function when there are errors in input is derived by developing the derivatives with respect to the parameters on the autocorrelation and cross correlation functions of the variables. The estimator is checked by a Monte Carlo method against a recursive multiple regression, when each model is the true one. It is shown in an example that an incorrect assumption on the structure of errors increases the risk of poor forecasting although the average quality of the results is not very much affected.

KEY WORDS : Transfer function – Input errors – Adaptive algorithm.

INTRODUCTION

Les modèles linéaires sont généralement calés en supposant que la variable observée est égale à la variable prédite par le modèle plus une erreur. On admet également parfois une erreur de mesure sur la variable observée (et ses réalisations antérieures utilisées comme variables explicatives) mais on suppose presque toujours que les autres variables explicatives sont connues sans erreur. Or, dans le cas des modèles pluie-débit, il est plus raisonnable de prendre en compte une erreur globale sur la pluie résultant d'une erreur de mesure, d'une erreur de représentativité spatiale et d'une erreur de transformation de la pluie brute en pluie nette, seule justiciable d'une transformation linéaire. La méthode de la DPFT (DUBAND, 1980) est l'un des rares exemples adoptant ce point de vue en hydrologie.

Par ailleurs, la linéarité n'est qu'une approximation plus ou moins grossière, et sans y renoncer complètement, il est souvent très efficace d'utiliser des modèles linéaires adaptatifs pour des phénomènes faiblement non linéaires ; COTE et BRUNELLE (1982) présentent une estimation adaptative d'une fonction de transfert.

(1) CEMAGREF – Division Hydrologie-Hydraulique BP 121, 92164 Antony Cedex.

Dans ce papier, ces deux approches ont été combinées, en limitant pour l'instant l'étude à des fonctions de transfert définies sur deux pas de temps mais avec un terme autorégressif. Le modèle étudié est donc :

$$Y_i = a Y_{i-1} + b_0 (X_i + \varepsilon_i) + b_1 (X_{i-1} + \varepsilon_{i-1})$$

La méthode est susceptible de généralisation à des ordres supérieurs quoiqu'au prix de calculs un peu complexes. Cependant les fonctions de transfert impliquent un grand nombre de termes résultant souvent d'erreurs dans la modélisation, ainsi l'omission du terme autorégressif relevée par BASTIN (1983) et probablement l'omission de l'erreur sur la variable explicative.

Après avoir établi l'algorithme récursif, nous étudierons l'influence des erreurs sur les variables explicatives par comparaison avec une estimation récursive des moindres carrés, qui n'en tient pas compte.

L'ALGORITHME D'ESTIMATION

Nous nous intéressons au modèle :

$$\begin{aligned} Y_i &= a Y_{i-1} + b_0 V_i + b_1 V_{i-1} \\ V_i &= X_i + \varepsilon_i \\ \varepsilon_i &= \text{erreur aléatoire (bruit blanc)} \end{aligned} \tag{1}$$

Lors des applications à des données réelles, il conviendra de rajouter un terme constant, mais cela ne change pas le principe de la démonstration et est inutile pour une étude de l'estimateur par une méthode de Monte Carlo.

Soit r_i le résidu estimé sur les réalisations (y_i, x_i) des variables

$$y_i = a y_{i-1} + b_0 (x_i + r_i) + b_1 (x_{i-1} + r_{i-1})$$

La prévision de Y_i faite à l'étape $i-1$ est :

$$y_{i|i-1} = a Y_{i-1} + b_0 x_i + b_1 (x_{i-1} + r_{i-1})$$

L'erreur de prévision est donc :

$$b_0 r_i \text{ que nous noterons } \ell_i$$

Posons $z_i = y_i - a y_{i-1} - b_0 x_i - b_1 x_{i-1}$

et $c = -b_1/b_0$

Il vient

$$e_i = c e_{i-1} + z_i \tag{2}$$

Nous voulons minimiser $E_i = \sum_{\ell \leq i} e_\ell^2$ par l'algorithme récursif

$$a_{i+1} = a_i - \frac{\partial e_{i+1}^2}{\partial a} (a_{i+1}) \frac{\partial^2 E_i}{\partial a^2} (a_{i+1}) \tag{3}$$

et les équations analogues sur b_0 et b_1 . Nous devons calculer e'_i et $\sum_{\ell \leq i} e_\ell^{2''}$, la dérivation étant prise par rapport à chacun des paramètres a, b_0, b_1 . Cet algorithme comporte une simplification puisqu'il ignore les termes non diagonaux de la matrice des dérivées partielles d'ordre 2. L'effet de cette simplification est souvent négligeable dans des problèmes d'estimation lorsque les paramètres d'un modèle ne sont pas redondants, a fortiori lorsqu'on effectue le calage sur des données simulées conformément au modèle. Les conclusions que l'on pourra tirer ne devraient être que renforcées si l'on utilisait l'algorithme exact.

CALCUL DES DÉRIVÉES

Le calcul sera conduit sans préciser le paramètre. Notons que l'algorithme est donc aussi valable pour un modèle linéaire avec une erreur à moyenne mobile d'ordre 1.

$$z_i = e_i + c e_{i-1}$$

identique à un signe près (arbitraire) à l'équation (2)

$$e'_\ell = c' e_{\ell-1} + e'_{\ell-1} + z'_\ell \tag{4}$$

$$e''_\ell = c'' e_{\ell-1} + 2c' e'_{\ell-1} + c e''_{\ell-1} \tag{5}$$

(le terme z''_ℓ , qui devrait apparaître dans la dernière équation, est toujours nul).

En utilisant (4) de façon itérative :

$$\begin{aligned} e'_i &= c' e_{i-1} + c(c' e_{i-2} + c e'_{i-2} + z'_{i-1}) + z'_i \\ &= c'(e_{i-1} + c e_{i-2}) + c^2 e'_{i-2} + (z'_i + c z'_{i-1}) \\ &= c'(z_{i-1} + 2c z_{i-2} + 2c^2 e_{i-3}) + c^2 e'_{i-2} + (z'_i + c z'_{i-1}) \end{aligned}$$

Dans la suite nous nous restreindrons au cas où $|c| < 1$. Les termes inconnus antérieurs au début des données ($i = 1$) peuvent donc être négligés et l'expression précédente se généralise par récurrence :

$$e'_i = \sum_{j \geq 0} (c'(j+1)c^j z_{i-1-j} + c^j z'_{i-j}) \quad (6)$$

Le calcul de $\sum_{\ell \leq i} (e^2_{\ell}/2)''$ s'appuie sur le même principe

$$\begin{aligned} (e^2_{\ell}/2)'' &= e_{\ell} e''_{\ell} + e'^2_{\ell} \\ &= (ce_{\ell-1} + z\ell)(c''e_{\ell-1} + 2c'e'_{\ell-1} + ce''_{\ell-1}) + (c'e_{\ell-1} + ce'_{\ell-1} + z'\ell)^2 \\ &= c^2(e_{\ell-1} e''_{\ell-1} + e'^2_{\ell-1}) + 4cc'e_{\ell-1} e'_{\ell-1} + (ce_{\ell-1} + z\ell)c''e_{\ell-1} \\ &\quad + (c'e_{\ell-1} + z'\ell)^2 + 2(c'z\ell + cz'\ell) e'_{\ell-1} + cz\ell c''e_{\ell-1} \end{aligned}$$

Soit en désignant par $F\ell$ les derniers termes de cette expression :

$$(e^2_{\ell}/2)'' = c^2 (e^2_{\ell-1}/2)'' + 4cc'e_{\ell-1} e'_{\ell-1} + F\ell$$

par sommation

$$(e^2_{\ell}/2)'' + (1-c^2) \sum_{\ell \leq i} (e^2_{\ell}/2)'' = 4cc' \sum_{\ell \leq i} e_{\ell-1} e'_{\ell-1} + \sum_{\ell \leq i} F\ell$$

Nous calculerons séparément les deux derniers termes notés F_0 et F_1 . Notons que F_i devrait être nul si les valeurs initiales sont bien estimées puisque c'est la dérivée de $\sum_{\ell < i} e^2_{\ell}$ que l'on cherche à minimiser.

$$F_1^i = (e^2_{\ell}/2)'' + (1-c^2) \sum_{\ell \leq i} (e^2_{\ell}/2)'' = \sum_{\ell \leq i} P^1_{\ell} + Q^1_{\ell} e'_{\ell-1} + R^1_{\ell} e''_{\ell-1}$$

avec

$$\begin{aligned} P^1_{\ell} &= (ce_{\ell-1} + z\ell) c''e_{\ell-1} + (c'e_{\ell-1} + z'\ell)^2 \\ &= (cc'' + c'^2) e^2_{\ell-1} + e_{\ell-1} (c''z\ell + 2c'z'\ell) + z'e_{\ell-1}{}^2 \quad (7) \\ Q^1_{\ell} &= 2(c'z\ell + cz'\ell) \\ R^1_{\ell} &= cz\ell \end{aligned}$$

En utilisant les relations de récurrence (4) et (5) F_1^i peut être exprimé avec les dérivées première et seconde des e d'indices inférieurs ou égaux à $\ell-k$.

$$F_1^i = P_i^k + Q_i^k e'_{i-k} + R_i^k e''_{i-k} + \sum_{\ell \leq i-k} P^1_{\ell} + Q^1_{\ell} e'_{\ell-1} + R^1_{\ell} e''_{\ell-1}$$

avec les relations

$$\begin{aligned} P_i^{k+1} &= P_i^k + Q_i^k (c'\ell_{i-k-1} + z'_{i-k}) + R_i^k c''_{i-k-1} + P_{i-k}^1 \\ Q_i^{k+1} &= c Q_i^k + 2c' R_i^k + Q_{i-k}^1 \\ R_i^{k+1} &= c R_i^k + R_{i-k}^1 \end{aligned} \quad (8)$$

Or les dérivées figurant dans l'expression de F_i s'annulent à partir de $k = i$, d'où :

$$F_1^i = P_i^i = \sum_{k \geq 1} Q_i^k (c' e_{i-k-1} + z'_{i-k}) + R_i^k c''_{i-k-1} + \sum_{k \geq 0} P_{i-k}^1 \quad (9)$$

Pour évaluer cette expression, nous identifierons e_i , R_i^k et Q_i^k à des suites R et S dont nous utiliserons certaines propriétés algébriques démontrées dans l'appendice A (références indexées par a).

$$\begin{aligned} z_i &= S(0, z_i) \\ e_i &= S(1, z_i) \\ R_i^k &= c R(1, z_i, k) \\ Q_i^k &= 2(c' R(1, z_i, k) + c R(1, z'_i, k) + cc' R(2, z_i, k-1)) \end{aligned}$$

Les deux premières formules résultent de la définition de S ; les deux dernières se démontrent par récurrence. En effet

$$R_{i-k}^1 = c z_{i-k} = c R(0, z_i, k+1)$$

si

$$R_i^k = c R(1, z_i, k)$$

alors

$$\begin{aligned} R_i^{k+1} &= c^2 R(1, z_i, k) + c R(0, z_i, k+1) = c R(1, z_i, k+1) \\ Q_i^1 &= 2(c'R(1, z_i, 1) + c R(1, z'_i, 1)) \end{aligned}$$

si

$$Q_i^k = 2(c'R(1, z_i, k) + c R(1, z'_i, k) + cc' R(2, z_i, k-1))$$

alors

$$Q^{k+1} = 2(c'R(1, z_i, k+1) + c R(1, z'_i, k+1) + c'(c^2R(2, z_i, k-1) + c R(1, z_i, k))) \\ = 2(c'R(1, z_i, k+1) + c R(1, z'_i, k+1) + c'c R(2, z_i, k))$$

L'équation 9 s'écrit alors après avoir remplacé P^{1-k} par sa valeur tirée de 7,

$$F^1_i = \sum_{k \geq 1} 2(c'R(1, z_i, k) + c R(1, z'_i, k) + cc' R(2, z_i, k-1)) (c'S(1, z_{i-k})) \\ + S(0, z'_{i-k}) + cc'' \sum_{k \geq 1} R(1, z_i, k) S(1, z_{i-k-1}) \\ + \sum_{\ell \leq i} (cc'' + c'^2) S(1, z_{i-\ell})^2 + S(1, z_{i-\ell}) (c'' z^\ell + 2c'z'\ell) + z'\ell^2$$

En utilisant les relations (4a), (10a), (11a), (13a)

$$F^1_i = (2c'^2 + cc'')T_2(z_i, z_{i-1}) + 2cc'T_2(z'_i, z_{i-1}) + 2cc'^2 T_3(z_i, z_{i-2}) \\ + 2c'T_1(z_i, z'_i) + 2c T_1(z'_i, z_i) + 2cc'T_2(z_i, z'_{i-1}) \\ + (cc'' + c'^2)/(1-c^2) (2 cT_1(z_i, z_i) + T_0(z_i, z_{i+1}) - e_i^2) \\ + c''T_1(z_i, z_i) + 2c'T_1(z'_i, z_i) + T_0(z'_i, z'_{i+1})$$

En utilisant (5a) et (6a) pour éliminer les termes en $U_i V_{i-1}$ et $U_i V_{i-2}$, et en posant $d = (cc'' + c'^2)/(1-c^2)$, il vient :

$$F^1_i = 2 c'^2/c T_3(z_i, z_i) + (c'' - c'^2/c) T_2(z_i, z_i) + 2c d T_1(z_i, z_i) \\ + 2c'(T_2(z_i, z'_i) + T_2(z'_i, z_i)) + 2c T_1(z'_i, z'_i) \\ + d (T_0(z_i, z_{i+1}) - e_i^2) + T_0(z'_i, z'_{i+1})$$

F_{0i} se calcule en utilisant l'équation (12a)

$$F_{0i} = \frac{4cc'}{1-c^2} (c T_1(z_i, z'_i) + c T_1(z'_i, z_i) + T_0(z_i, z'_{i+1}) - e_i e'_i)$$

d'où pour la somme de F_{0i} et F^1_i

$$F_{0i} + F^1_i = 2 c'^2/c T_3(z_i, z_i) + (c'' - 2c'^2/c) T_2(z_i, z_i) + 2cd T_1(z_i, z_i) \\ + 2c'(T_2(z_i, z'_i) + T_2(z'_i, z_i)) + \frac{4c'c^2}{1-c^2} (T_1(z_i, z'_i) + T_1(z'_i, z_i)) + 2c T_1(z'_i, z'_i) \\ + d_3 (T_0(z_i, z_{i+1}) - e_i^2) + d_2 (T_0(z_i, z'_{i+1}) - e_i e'_i) + T_0(z'_i, z'_{i+1}) \quad (10)$$

avec

$$d_2 = \frac{4cc'}{1-c^2}$$

Cette expression se réduit, grâce à l'équation (9a), en une somme de $C^j T_0(u_i, V_{i-j})$ avec u_i et V_i pris dans $(z_i z'_i z_{i+1} z'_{i+1})$

Les fonctions c'' , d , d_2 doivent à ce stade être calculées selon le paramètre par rapport auquel on dérive. Rappelons que

$$T_0(u_i, V_i) = u_i V_{i-1} + u_{i-1} V_{i-2} + \dots$$

Le calcul de $T_0(u_i, V_{i-j})$ exige de conserver en mémoire les j dernières valeurs de V et comme j varie de 0 à i , il faudrait théoriquement conserver toute l'histoire du processus. $T_0(u_i, V_{i-j})$ étant pondéré par c^j , il suffit, pour une approximation donnée, de faire le calcul pour $j \leq n_j$, n_j étant fixé, et donc de conserver en mémoire un nombre fini des dernières réalisations de V .

Cependant u_i et V_i dépendent des paramètres a , b_0 , b_1 réestimés à chaque pas de temps et doivent en fait être calculés à partir des valeurs de x_i et y_i conservées en mémoire et des valeurs courantes estimées de a , b_0 , b_1 (et $c = -b_1/b_0$), au moins dans le cas où les vraies valeurs de ces paramètres sont supposées constantes.

Il est plus rapide d'exprimer les $T_0(V_i, V_j)$ en fonction des $T_0(x_i, x_j)$ $T_0(x_i, y_i)$ $T_0(y_i, y_j)$. Ce calcul complémentaire est effectué dans l'appendice B. Le calcul de

$$F_{0i} + F^1_i = (e_i^2/2)'' + (1-c^2) \sum_{\ell \leq i} (e_\ell^2/2)''$$

permet celui de

$$G_i = \sum_{\ell \leq i} (e_\ell^2/2)'' = c^2 G_{i-1} + F_{0i} + F^1_i$$

Résolution de l'équation

Utilisons maintenant des notations vectorielles :

$$A_i = \begin{pmatrix} a \\ b_0 \\ b_1 \\ -1 \end{pmatrix} = \begin{pmatrix} a_1 \\ a' \\ -1 \end{pmatrix} \quad X_i = \begin{pmatrix} Y_{i-1} \\ x_i \\ X_{i-1} \\ Y_i \end{pmatrix}$$

En considérant la dérivation par rapport à chacun des paramètres, l'équation (3) devient, en utilisant (6)

$$A_{i+1} = A_i + e_{i+1} (L_i + M_i e'_{i+1})$$

avec

$$\begin{aligned} L_i &= 0 \\ L_i' &= \frac{\delta c}{\delta a_r} / \frac{\delta^2 E_i}{\delta a^2_r} (A_{i+1}) \\ M_i' &= \frac{\delta z_i}{\delta a_r} / \frac{\delta^2 E_i}{\delta a^2_r} (A_{i+1}) \\ e_{i+1} &= \sum_{j \geq 0} c_j X_{i+1-j} \quad A_{i+1} = U_{i+1} A_{i+1} \\ e'_{i+1} &= \sum_{j \geq 0} (j+1) c_j X_{i+1-j} \quad A_{i+1} = V_{i+1} A_{i+1} \end{aligned}$$

avec :

$$U_i = \sum_{j \geq 0} c_j X_{i-j} \quad \text{et} \quad V_i = \sum_{j \geq 0} (j+1) c_j X_{i-j}$$

En omettant les indices *i*, on a donc l'équation vectorielle

$$A = A_0 + (U^t A) (L + M (V^t A)) \tag{11}$$

ou *A* est l'inconnue. Cette équation, de la forme

$$A = f (U(A), V(A), L(A), M(A), A, A_0)$$

est trop fortement implicite en *A* pour être résolue par une méthode itérative. Si l'on considère *U*, *V* et *M* constants, on a un système du second degré en *A* dont la résolution donnera une équation $A = g(U(A), V(A), L(A), M(A), A_0)$ plus facile à résoudre par une méthode itérative.

Notons de plus

$$\varepsilon = U^t A_0 \quad \varepsilon' = V^t A_0 \tag{12}$$

sachant que

$$e = U^t A \quad e' = V^t A$$

Dans le problème statistique, *e* est le résidu avec les paramètres optimisés en tenant compte des données du pas de temps courant, *ε* le résidu avec les paramètres optimisés au pas de temps précédent.

$$\begin{aligned} A - A_0 &= e(L + Me') = (\varepsilon e - \varepsilon) (L + M(\varepsilon' + (e - \varepsilon))) \\ &= (L + M\varepsilon') + (e - \varepsilon) + M\varepsilon' + L (e' - \varepsilon') + M (e - \varepsilon) (e' - \varepsilon') \end{aligned}$$

Soit

$$\begin{aligned} A_1 &= L + M\varepsilon' \\ A_2 &= M\varepsilon \\ A - A_0 &= \varepsilon A_1 + A_1 U^t (A - A_0) + A_2 V^t (A - A_0) + M (e - \varepsilon) (e' - \varepsilon') \end{aligned}$$

soit

$$\begin{aligned} \Omega &= I - A_1 U^t - A_2 V^t \\ (A - A_0) &= \varepsilon A_1 + M (e - \varepsilon) (e' - \varepsilon') \end{aligned} \tag{13}$$

Éliminons d'abord $A - A_0$ entre (12) et (13)

$$\begin{aligned} e - \varepsilon &= U^t (A - A_0) = U^t \Omega^{-1} (\varepsilon A_1 + M (e - \varepsilon) (e' - \varepsilon')) \\ e' - \varepsilon' &= V^t (A - A_0) = V^t \Omega^{-1} (\varepsilon A_1 + M (e - \varepsilon) (e' - \varepsilon')) \end{aligned}$$

soit

$$A_3 = \Omega^{-1} A_1 \quad \text{et} \quad M_1 = \Omega^{-1} M$$

On obtient l'équation (scalaire) du second degré en $(e - \varepsilon) (e' - \varepsilon')$

$$(e - \varepsilon) (e' - \varepsilon') = \varepsilon^2 U^t A_3 V^t A_3 + 2(U^t + V^t) M_1 (e - \varepsilon) (e' - \varepsilon') + U^t M_1 V^t M_1 ((e - \varepsilon) (e' - \varepsilon'))^2$$

Le choix entre les deux racines pose un problème. Les essais permettent seulement de dire qu'il est préférable d'utiliser toujours la racine de plus grande valeur absolue, plutôt que toujours celle de moins grande valeur absolue, du moins pour l'exemple traité, en partant des valeurs initiales exactes.

La résolution de (13) s'ensuit. De plus, la forme particulière de la matrice facilite le calcul de son inverse.

$$(I - A_1 U^t - A_2 V^t)^{-1} = \frac{(1 - V^t A_2) A_1 U^t + (U^t A_2) A_1 V^t + (1 - U^t A_1) A_2 V^t + (V^t A_1) U^t}{(1 - U^t A_1) (1 - V^t A_2) - U^t A_2 V^t A_1}$$

Le calcul se termine de façon itérative

$$A_{k+1} = g(A_k) \quad \text{ou} \quad A_{k+1} = \frac{A_k + r g(A_k)}{1 + r}$$

Plutôt que d'affiner la méthode itérative en allongeant le temps de calcul, on a toléré que la méthode ne converge pas à certaines étapes dans certains essais, auquel cas on passe au pas de temps suivant sans correction.

ÉTUDE EXPÉRIMENTALE

L'étude théorique de la qualité de l'estimateur semble hors de portée, et l'étude par simulation est assez coûteuse en temps de calcul. Une étude partielle a été conduite par une méthode de Monte Carlo pour un jeu particulier de paramètres :

$$Y_i = 0.7 Y_{i-1} + 0.6 V_i + 0.2 V_{i-1}$$

$$V_i = X_i + \varepsilon_i$$

X_i et ε_i étant tirés au hasard dans une loi normale (et seule la valeur de X_i est utilisée dans l'algorithme). Trois valeurs du rapport des écarts types $\sigma_\varepsilon/\sigma_x$ ont été testées : 0.2, 0.4 et 1.

Ce contrôle ne teste l'algorithme que sous ses hypothèses théoriques, c'est-à-dire, valeurs réelles des paramètres constants et valeurs initiales connues, donc avec un intérêt opérationnel limité aux corrections en temps réel, mais c'est évidemment le premier point à vérifier.

Dix essais ont été simulés pour chaque valeur de $\frac{T_2}{T_x}$ et on a noté les coefficients de détermination $\left(= 1 - \frac{\text{(résidu } T_x)}{Y} \right)$ moyens et minimaux calculés sur 30, 50, 80 et 100 points, après une phase d'initialisation de 10 points. Le résidu utilisé dans le calcul du coefficient de détermination est celui calculé avec les paramètres optimisés sur le pas de temps courant mais les résultats avec l'erreur de prévision calculée avec les paramètres optimisés au pas de temps précédent ne sont inférieurs que de quelques millièmes.

On a également estimé concurremment la régression multiple récursive (RM) sur les mêmes variables et les mêmes données simulées par le modèle à erreur sur les entrées, ou modèle à variable instrumentale (VI), pour voir l'effet d'une telle structure d'erreur sur les procédures d'estimation habituelles et juger de l'intérêt éventuel d'un modèle plus complexe, à confirmer bien entendu sur des cas réels.

n nombre de points
 a moyennes des paramètres (sur 10 essais), a , b_0 , b_1
 da^2 variance de a — a_{vrai} en 10^{-4}
 R^2_{Min} minimum du coefficient de détermination
 R^2_{Moy} moyenne du coefficient de détermination

(Voir tableaux pages suivantes)

Pour $\sigma_\varepsilon/\sigma_x = 0.2$, les estimations selon le modèle exact sont à tous égards supérieures à celles de la régression, sauf que le coefficient de la variable Y_{i-1} présente un biais plus important.

Pour $\sigma_\varepsilon/\sigma_x = 0.4$, le coefficient de Y_{i-1} est également moins bien estimé et le R^2 moyen est à peine supérieur. Cependant le critère du R^2 minimum, qui représente le risque d'estimations particulièrement mauvaises, est bien favorable au modèle exact.

Pour $\sigma_\varepsilon/\sigma_x = 1$, la supériorité du modèle correct redevient manifeste.

Même si le nombre d'essais est sans doute faible pour des conclusions quantitatives définitives, il se dégage une tendance à la dégradation des performances d'une régression récursive, lorsque les entrées du modèle comportent des erreurs, même si ses paramètres ne sont pas trop mal estimés.

L'algorithme selon le modèle correct atteint les valeurs théoriques du coefficient de détermination

$$1 - \frac{0.459}{1 + \left(\frac{\sigma_\varepsilon}{\sigma_x} \right)^2} \quad \text{et le dépasse même pour } \frac{\sigma_\varepsilon}{\sigma_x} = 1.$$

TABLEAU I

 $\sigma\epsilon/\sigma x = .2$

n	simulation modèle VI	estimation modèle RM	R ² Min	R ² Moy
30	a .696 d _a ² 42	.599 28 .225 53	.663	.855
50	a .692 d _a ² 37	.598 30 .226 54	.786	.913
80	a .687 d _a ² 29	.599 31 .226 54	.804	.914
100	a .690 d _a ² 27	.599 31 .226 54	.874	.929

TABLEAU II

 $\sigma\epsilon/\sigma x = .4$

n	simulation modèle VI	estimation modèle RM	R ² Min	R ² Moy
30	a .710 d _a ² 53	.611 8 .175 139	.950	.972
50	a .712 d _a ² 51	.612 9 .175 142	.949	.973
80	a .720 d _a ² 47	.612 9 .174 143	.946	.976
100	a .718 d _a ² 46	.612 9 .174 143	.940	.975

n	simulation modèle VI	estimation modèle VI	R ² Min	R ² Moy
30	a .614 d _a ² 281	.587 27 .193 17	.734	.853
30	a .626 d _a ² 237	.585 26 .193 17	.822	.910
80	a .638 d _a ² 152	.585 26 .193 17	.853	.916
100	a .658 d _a ² 79	.585 26 .193 18	.900	.931

n	simulation modèle VI	estimation modèle VI	R ² Min	R ² Moy
30	a .656 d _a ² 34	.596 5 .213 9	.971	.981
50	a .667 d _a ² 17	.596 5 .213 9	.968	.982
80	a .672 d _a ² 12	.596 5 .213 9	.978	.983
100	a .674 d _a ² 10	.596 5 .213 9	.981	.984

TABLEAU III

$\sigma\epsilon/\sigma_x = 1$

n		simulation modèle RM estimation modèle RM			R ² Min	R ² Moy
30	a	.709	.597	.191	.876	.943
	d _a ²	17	34	53		
50	a	.708	.597	.191	.876	.940
	d _a ²	17	35	53		
80	a	.717	.598	.191	.913	.950
	d _a ²	14	35	54		
100	a	.712	.598	.191	.903	.947
	d _a ²	12	35	54		

TABLEAU IV

n		simulation modèle VI estimation modèle RM			R ² Min	R ² Moy
30	a	.751	.631	.134	.477	.709
	d _a ²	80	26	68		
50	a	.750	.631	.135	.514	.771
	d _a ²	43	27	66		
80	a	.735	.631	.135	.700	.803
	d _a ²	18	27	67		
100	a	.744	.631	.135	.727	.799
	d _a ²	27	27	67		

n		simulation modèle RM estimation modèle VI			R ² Min	R ² Moy
30	a	.683	.588	.209	.800	.916
	d _a ²	210	37	6		
50	a	.661	.586	.209	.847	.918
	d _a ²	184	32	6		
80	a	.653	.582	.208	.856	.922
	d _a ²	172	29	6		
100	a	.647	.582	.207	.858	.926
	d _a ²	142	29	6		

n		simulation modèle VI estimation modèle VI			R ² Min	R ² Moy
30	a	.679	.590	.175	.662	.778
	d _a ²	166	47	36		
50	a	.692	.589	.174	.708	.806
	d _a ²	85	51	42		
80	a	.700	.588	.174	.708	.816
	d _a ²	58	51	43		
100	a	.713	.588	.174	.732	.818
	d _a ²	65	50	43		

On peut remarquer aussi que l'algorithme ne parvient pas à améliorer les 2^e et 3^e coefficients entre les 30^e et 100^e points.

On a par ailleurs calé les deux modèles sur des valeurs simulées selon le modèle de régression multiple, avec un niveau de bruit équivalent au cas intermédiaire. On voit ainsi par comparaison que l'application d'une régression multiple sur des données qui ne satisfont pas ses hypothèses ne donne pas un très grand biais ni une très grande variance aux estimations de paramètres (cependant paradoxalement plus forte pour le coefficient d'autocorrélation, qui ne semblait pas concerné) mais détériore cependant la qualité des plus mauvais résultats que l'on risque d'obtenir avec le modèle, surtout pour les échantillons de taille réduite.

Par ailleurs, il serait plutôt moins grave d'utiliser à tort un modèle à variable instrumentale, que d'utiliser à tort un modèle de régression multiple (testés respectivement lorsque l'autre est le vrai modèle).

CONCLUSION

Avec un modèle sur deux pas de temps, on a montré l'effet nuisible des erreurs sur les entrées, sur l'utilisation en temps réel d'une estimation récursive d'un modèle linéaire.

L'algorithme mis au point pour tenir compte de cette structure d'erreur, malgré quelques difficultés de convergence, se révèle satisfaisant sur des données simulées, la vérification ayant été faite jusqu'à un niveau de bruit de variance égal à la variance de la commande.

L'application à des données réelles nécessite quelques travaux complémentaires comme l'estimation d'un terme constant, surtout si on veut utiliser l'algorithme, non plus en correction, mais pour une estimation à partir de valeurs initiales très incertaines. La généralisation souhaitable à plus de deux pas de temps est assez facile en ce qui concerne la résolution de l'équation fondamentale, qui a d'ailleurs été traitée sous forme vectorielle, mais plus difficile pour le calcul des dérivées secondes.

Par ailleurs, le fait que le modèle présenté ici se réduise pour $c = c' = c'' = 0$ à la régression, suggère qu'un modèle intermédiaire puisse être intéressant sur des cas réels.

REMERCIEMENTS

Cette étude fait partie d'un travail effectué dans le cadre d'une convention avec le Ministère de l'Environnement (n° 237018240144) et mené conjointement avec le Lab. National d'Hydraulique (EDF). La mise au point du programme a été achevée sur l'ordinateur du Natural Environment Research Council à l'Institut d'Hydrologie de Wallingford (Grande-Bretagne).

Manuscrit accepté par le Comité de Rédaction le 15.9.1986

BIBLIOGRAPHIE

- BASTIN (G.), 1983. - Techniques d'identification de la relation Pluie-Débit. in : Crues et précipitations intenses. Système d'annonce - Prévision-valeurs extrêmes. Ec. Natle Sup. d'Hydraulique, Grenoble.
- COTE et BRUNELLE, 1982. - Fonction de transfert linéaire autoadaptative *J. Hydraul. Res.* 20(5).
- GUILLOT (P.) et DUBAND (D.), 1980. - Fonction de transfert Pluie-Débit sur les bassins versants de l'ordre de 1 000 km² - La Houille Blanche, numéro spécial 4-5/1980.

APPENDICE A

Propriétés algébriques des suites R et S intervenant dans le calcul de la dérivée seconde

Soit deux suites quelconques μ_i et v_i , une constante c et les suites R et S telles que :

$$\begin{aligned} R(0, u_i, k) &= \mu_{i-k+1} \\ m \geq 0 \quad R(m, u_i, 1) &= \mu_i \\ k > 1 \quad R(m, u_i, k) &= c R(m, u_i, k-1) + R(m-1, \mu_i, k) \end{aligned} \tag{1}$$

$$\begin{aligned} S(0, v_i) &= v_i \\ S(m, v_i) &= c S(m, v_i) + S(m-1, v_i) \end{aligned} \tag{2}$$

et la forme bilinéaire

$$T_{mm'}(\mu_i, v_i) = \sum_{k \geq 1} R(m, u_i, k) S(m', v_{i-k}) \tag{3}$$

En utilisant les relations de définition de R et S, on peut vérifier par récurrence sur k que :

$$\begin{aligned} T_{mm'}(i, v_i) &= \sum_{1 \leq k < k_1} R(m+1, u_i, k) S(m'-1, v_{i-k}) \\ &\quad + R(m+1, u_i, k_1) S(m', v_{i-k_1}) \\ &\quad + \sum_{k > k_1} R(m, u_i, k) S(m', v_{i-k}) \end{aligned}$$

$$\text{d'où } T_{mm'}(u_i, v_i) = \sum_{k \geq 1} R(m+1, u_i, k) S(m'-1, v_{i-k}) = T_{m+m'}(u_i, v_i)$$

$$\text{d'où par récurrence } T_{mm'}(u_i, v_i) = T_{m+m'}(u_i, v_i) \tag{4}$$

T peut donc s'écrire avec un seul indice.

On a en particulier

$$T(u_i, v_i) = \sum_{k \geq 1} u_{i-k+1} v_{i-k} = \sum_{k \geq 0} u_{i-k-1} v_{i-k-1}$$

R, S et T possèdent d'autres propriétés :

$$\begin{aligned} T_m(u_i, v_i) &= \sum_{k \geq 1} R(m-1, u_i, k) S(1, v_{i-k}) \\ &= \sum_{k \geq 1} R(m-1, u_i, k) (S(0, v_{i-k}) + c S(1, v_{i-k-1})) \\ &= T_{m-1}(u_i, v_i) + c T_m(u_i, v_{i-1}) \end{aligned}$$

$$\text{d'où } T_m(u_i, v_{i-1}) = 1/c (T_m(u_i, v_i) - T_{m-1}(u_i, v_i)) \tag{5}$$

$$\text{de même } T_m(u_i, v_{i-2}) = 1/c^2 (T_m(u_i, v_i) - 2T_{m-1}(u_i, v_i) + T_{m-2}(u_i, v_i)) \tag{6}$$

D'après sa définition R(m, u_i, k) peut se mettre sous la forme

$$R(m, u_i, k) = \sum_{j=0}^{k-1} \lambda_j^m c^j u_{i-k+j+1} \tag{7}$$

avec $\lambda_j^m = \lambda_j^{m-1} + \lambda_{j-1}^{m-1}$

$$\text{d'où } \lambda_j^m = \binom{m+j-1}{j} \text{ pour } m \geq 1 \text{ et } \lambda_0^1 = 1 \tag{8}$$

$$T_m(u_i, v_i) = \sum_{k \geq 1} R(m, u_i, k) S(0, v_{i-k}) = \sum_{k \geq 1} \sum_{j=0}^{k-1} \lambda_j^m c^j u_{i-k+j+1} v_{i-k}$$

soit $k' = k-j$, $p-1 \geq j \Leftrightarrow k' > 1$.

$$\begin{aligned} T_m(u_i, v_i) &= \sum_{j \geq 0} \lambda_j^m c^j \sum_{k' \geq 1} u_{i+1-k'} v_{i-k'-j} \\ &= \sum_{j \geq 0} \lambda_j^m c^j \sum_{k' \geq 1} R(0, u_i, k') S(0, v_{i-j-k'}) \end{aligned}$$

$$T_m(u_i, v_i) = \sum_{j \geq 0} \lambda_j^m c^j T_0(u_i, v_{i-j}) \tag{9}$$

Les propriétés suivantes sont de simples manipulations d'indices qui permettent de reconnaître des formes T qui se présentent dans le calcul avec des suites u_i et v_i identiques que nous noterons z_i .

$$\begin{aligned} \sum_{\ell \leq i} z^2 \ell &= \sum_{k \geq 1} z_{i-k+1} z_{i-k+1} = \sum_{k \geq 1} R(0, z_i, k) S(0, z_{i+1-k}) \\ \sum_{\ell \leq i} z^2 \ell &= T_0(z_i, z_{i+1}) \end{aligned} \tag{10}$$

$$\sum_{\ell \leq i} z \ell S(m, z_{i-1}) = \sum_{k \geq 1} z_{i-k+1} S(m, z_{i-k}) = \sum_{k \geq 1} R(0, z_i, k) S(m, z_{i-k})$$

$$\sum_{\ell \leq i} z \ell S(m, z_{i-1}) = T_m(z_i, z_i) \tag{11}$$

En faisant le produit puis en sommant sur $\ell \leq 1$

$$S(1, U_i) = c S(1, U_{i-1}) + U_i$$

$$S(1, V_i) = c S(1, V_{i-1}) + V_i$$

$$S(1, U_i)S(A, V_i) + (1-c^2) \sum_{\ell \leq 1} S(1, U_{i-\ell})S(1, V_{i-\ell}) = c T_1(U_i, V_i) + c T_1(V_i, U_i) + U_i V_i \quad (12)$$

En particulier

$$S(1, z_i)^2 + (1-c^2) \sum_{\ell \leq i} S(1, z_{i-\ell})^2 = 2c \sum_{\ell \leq i} z_\ell S(1, z_{i-\ell}) + \sum_{\ell \leq i} z_\ell^2 = 2c T_1(z_i, z_i) + T_0(z_i, z_{i+1}) \quad (13)$$

APPENDICE B

Calcul des produits en z en fonction des produits en x et y

Notons maintenant * la forme bilinéaire précédemment notée T_0 .

$$u_i^* v_i = \sum_{k \geq 1} u_{i-k+1} v_{i-k} = \sum_{k \geq 0} u_{i-k} v_{i-1-k}$$

La contribution C_i des termes $z_i^* z_i, z_i^* z'_i, z'_i^* z_i$ et $z'_i^* z'_i$ pour chacun des paramètres (par rapport auquel est faite la dérivation) peut se mettre sous une forme unique :

$$C_i = \sum_j h_j (\lambda_0 y_i + \lambda_1 y_{i-1} + \lambda_2 x_i + \lambda_3 x_{i-1})^* (\mu_0 y_i + \mu_1 y_{i-1} + \mu_2 x_i + \mu_3 x_{i-1})$$

avec par exemple pour $z_i^* z'_i$, ' représentant la dérivation par rapport à b_0 :

$$\begin{aligned} \lambda_0 &= 1 & \lambda_1 &= -a & \lambda_2 &= -b_0 & \lambda_3 &= -b_1 \\ \mu_0 &= 0 & \mu_1 &= 0 & \mu_2 &= -1 & \mu_3 &= 0 \end{aligned}$$

Ce terme peut être développé :

$$C_i = \sum_{j \geq 0} h_j (\lambda_0 y_i + \lambda_2 x_i)^* (\mu_0 y_{i-j} + \mu_1 y_{i-j-1} + \mu_2 x_{i-j} + \mu_3 x_{i-j-1})$$

$$+ h_j (\lambda_1 y_{i-1} + \lambda_3 x_{i-1})^* (\mu_0 y_{i-1-j-1} + \mu_1 y_{i-1-j} + \mu_2 x_{i-1-j-1} + \mu_3 x_{i-1-j})$$

$$\text{soit } (h^1_j = \mu_0 h_j + \mu_1 h_{j-1})$$

$$\text{et } (h^2_j = \mu_2 h_j + \mu_3 h_{j-1})$$

$$C_i = \sum_j (\lambda_0 y_i + \lambda_2 x_i)^* (h^1_j y_{i-j} + h^2_j x_{i-j}) + (\lambda_1 y_{i-1} + \lambda_3 x_{i-1})^* (h^1_{j+1} y_{i-1-j} + h^2_{j+1} x_{i-1-j})$$

les termes $u_i^* v_{i-j}$ interviennent dans C_i avec l'indice $i-1$ et l'indice i . Or, ces termes sont des sommes incrémentées à chaque pas de l'algorithme. Il suffit donc de les multiplier par les coefficients correspondants respectivement avant et après l'incrément. En soulignant les coefficients qui doivent être utilisés avant l'incrément, nous pouvons écrire :

$$C_i = (\lambda_0 h^1_j + \lambda_1 h^1_{j+1}) y_i^* y_{i-j} + (\lambda_2 h^1_j + \lambda_3 h^1_{j+1}) x_i^* y_{i-j} + (\lambda_0 h^2_j + \lambda_1 h^2_{j+1}) y_i^* x_{i-j} + (\lambda_2 h^2_j + \lambda_3 h^2_{j+1}) x_i^* x_{i-j}$$

Il suffit donc d'incrémenter quatre sommes dépendant des données pour obtenir tous les termes $T_k(u_i, v_i)$.

APPENDICE C

Listing des sous-programmes utilisés

CURRENT FILE :
Z052RECURS
00001

```

SUBROUTINE RECUVI(I,IVI,XI,YI,IOUT, A, EI,EITR)
C CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C < SOUS PROGRAMME RECUVI(I,IVI,XI,YI,IOUT,A,EI,EITR)
C <
C < ALGORITHME RECURSIF DE CALAGE D'UN MODELE A VARIABLE INSTRUMENTALE
C < (AVEC SIMPLIFICATION TERMES DIAGONAUX NULS)
C
C Y(I+1) = YI + A(1) *YI +A(2) *( XI +EPS(I)) + A(3)* (X(I-1)+EPS(I-1))
C EN ENTREE
C -----
C I =1 POUT INITIALISATION
C IVI =1 POUR MODELE A VARIABLE INSTRUMENTALE
C IVI =0 POUR ALGORITHME DEGENERE ( REGRESSION MULTIPLE)
C ( AVEC MEME SIMPLIFICATION)
C (A DEFAUT D'UN SOUS PROGRAMME 100 FOIS PLUS SIMPLE)
C XI COMMANDE AU PAS DE TEMPS COURANT
C YI VALEUR OBSERVEE DE LA VARIABLE A PREVOIR
C IOUT UNITE LOGIQUE POUR SORTIE DES EVENTUELS DIAGNOSTICS DE
C NON CONVERGENCE
C EN ENTREE SORTIE
C -----
C A ( ) PARAMETRES
C EN SORTIE
C -----
C EI RESIDU A POSTERIORI AVEC PARAMETRES OPTIMISE SUR LE MEME P.D..T
C EITR ERREUR DE PREVISION EN TEMPS REEL (AVEC PARAMETRES OPTIMISES
C AU PAS DE TEMPS PRECEDENT)
C
REAL LAMBDA,MU
DIMENSION A(1),AK(4),DA(4),E2(4),C1(4),U(4),V(4),LAMBDA(4),MU(4)
DIMENSION Y(10),X(10),FJ(14,5)
EQUIVALENCE (Y,FJ),(X,FJ(1,2))
DIMENSION SXX(10,2),SXY(10,2),SYX(10,2),SYY(10,2)
DATA NJ/10/
IF(I.EQ.1)CALL RECO(II,CURM,R,NK,FJ,SXX,SXY,SYX,SYY)
IF(I.EQ.1)NKK=0
II=II+1
IF(II.GT.NJ)II=1
III=II-1
IF(III.LE.0)III=III+NJ
A(4)=-1.
AK(1)=A(1)
AK(2)=A(2)
AK(3)=A(3)
CALLSOM(II,XI,YI,NJ, X,Y,SXX(1,2),SXY(1,2),SYX(1,2),SYY(1,2))

```

```

C
C.....
C
C   BOUCLE SUR ITERATIONS
C
      K=0
100 K=K+1
      CALLREC(IVI,0,AK,NJ,XI,YI,II,X,Y,SXX,SXY,SYX,SY, E2)
      CALLREC(IVI,1,AK,NJ,XI,YI,II,X,Y,SXX(1,2),SXY(1,2),
& SYX(1,2),SY(1,2),E2)
      C=-AK(3)/AK(2)
      IF(ABS(AK(2)).LE.1.E-04.OR.ABS(AK(3)).LE.1.E-04)C=0.

      CORMAX=0.
      DCORMX=0.
      C1(1)=0.
      C1(3)=-FLOAT(IVI)/AK(2)
      C1(2)=C*C1(3)
      CC=C*IVI
      DO 200 L=1,5
      IF(L.GT.2)FJ(II,L)=E2(L-2)
      IF(L.EQ.3)CC=C*C
200 CALL FILTAR(CC,NJ,II, FJ(1,L))
      CALL CONDIT(FJ,C1, U,V,INDEF,LAMBDA,MU)
      IF(INDEF.EQ.0)CALL E2DVEC(4,A,U,V,LAMBDA,MU, DA)
      DO 300 L=1,3
300 CORMAX=AMAX1(CORMAX,ABS(DA(L)))
      EI=U(4)
      DO 400 L=1,3
      COR=DA(L)
      IF(ABS(E2(L)).LT.1.E-03)COR=0.
      CORM=CORM* CORM / (AMAX1(CORM,CORMAX) )
      AL=A(L)+COR
      DCOR=AK(L)
      AK(L)=( R* AL + IVI * AK(L) ) / ( R+ FLOAT(IVI) )
      DCORMX=AMAX1(DCORMX,ABS(AK(L)-DCOR))
400 EI=EI-U(L)*AK(L)
      IF(K.EQ.1)EITR=EI
C   WRITE(6,1002)K,(AK(L),L=1,3)
1002 FORMAT( I4,3F9.3)
      IF(IVI.EQ.1.AND.DCORMX.GT..005.AND.K.LT.NK)          GOTO100
      IF(K.EQ.NK)WRITE(IDOUT,1010)DCORMX
1010 FORMAT(' NON CONVERGENCE , ECART MAXI SUR UN PARAMETRE :',F9.3)
      CALL SOM(II,XI,YI,NJ, X,Y,SXX,SXY,SYX,SY)
      IF(K.EQ.NK.AND.DCORMX.GT..01)          RETURN
      A(1)=AK(1)
      A(2)=AK(2)
      A(3)=AK(3)
      RETURN
      END

C   INITIALISATION DE RECVI
      SUBROUTINE RECO(II,CORM,R,NK,FJ,SXX,SXY,SYX,SY)
      DATA CORM0/.1/,R0/.5/,NKO/10/
C   TOUS ARGUMENTS EN SURTIE
C   CORM,R,NK   INITIALISES EN DATA
C   CORM      : CORRECTION MAXIMALE ACCEPTEE SUR 1 PARAMETE ENTRE 2
C               ITERATIONS

```

```

C      R      : POIDS DE LA VALEUR CORRIGEE THEORIQUE PAR RAPPORT A
C              LA VALEUR PRECEDENTE , DANS LE CALCUL DE LA VALEUR
C              EFFECTIVEMENT CORRIGEE A CHAQUE ITERATION
C      NK      : NOMBRE MAXIMUM D'ITERATIONS
C
      DIMENSION FJ(14,5),SXX(10,1),SXY(10,1),SYX(10,1),SYY(10,1)
      R=R0
      CORM=CORM0
      NK=NK0
      II=0
      DO 200 J=1,10
      DO 100 L=1,5
100    FJ(J,L)=0.
      DO 200 L=1,2
      SXX(J,L)=0.
      SXY(J,L)=0.
      SYX(J,L)=0.
200    SYY(J,L)=0.
      RETURN
      END

C      MISE A JOUR DE LA MEMOIRE GLISSANTE DE LONGUEUR FINIE NJ :
C      X,Y ET 'AUTOCORRELATIONS' SXX.. . (II RANG DU POINT TRANSMIS)
      SUBROUTINE SOM(II,XI,YI,NJ ,X,Y,SXX,SXY,SYX,SYY)
      DATA X11,Y11/2*0./
      DATA Z/1./
      REAL LAMBDA,MU
      DIMENSION X(1),Y(1),SXX(1),SXY(1),SYX(1),SYY(1)
      X(II)=XI
      Y(II)=YI
      I11=II-1
      IF(I11.EQ.0)I11=I11+NJ
      X11=X(I11)
      Y11=Y(I11)
      DO 100 L=1,2
      I1J=II-3+L
      DO 100 J=8*L-7,6+2*L
      IF(I1J.LE.0)I1J=I1J+NJ
      SXX(J)=Z * SXX(J) + X11 * X(I1J)
      SYX(J)=Z * SYX(J) + Y11 * X(I1J)
      SXY(J)=Z * SXY(J) + X11 * Y(I1J)
      SYY(J)=Z * SYY(J) + Y11 * Y(I1J)
100    I1J=I1J-1
      RETURN
      END

C      CALCUL DE E2 ,DERIVEE SECONDE SOMME DES ECARTS
      SUBROUTINE REC(IVI,JO,AK,NJ,XI,YI,II,X,Y,SXX,SXY,SYX,SYY ,E2)
C      JO=0 INITIALISATION ET CONTRIBUTION DES SOMMES AVANT INCREMENTATION
C      JO=1 AJOUT DE LA CONTRIBUTION APRES INCREMENTATION (CF APPENDICE B)
C      AK( ) VECTEUR PARAMETRE AU COURS DE L'ITERATION
C      NJ .... SYY ETAT DE LA MEMOIRE
C      E2 SORTIE
      DIMENSION X(1),Y(1),AK(1),E2(1),H(10),D(7),EE(4),EE1(4)
      DIMENSION SXX(1),SXY(1),SYX(1),SYY(1)
      REAL LAMBDA(4),MU(4)
      EXTERNAL UN,FEE,FEE1
      H(10)=0.

```

```

C=-AK(3)/AK(2)
IF(ABS(AK(3)).LE.1.E-04)C=0.
DO 100 K=1,3
C1=0.
IF(K.EQ.2)C1=-C/AK(2)
IF(K.EQ.3)C1=-1./AK(2)
IF(ABS(AK(2)).LE.1.E-04)C1=0
C2=0.
IF(K.EQ.2)C2=-2*C1/AK(2)
IF(ABS(AK(2)).LE.1.E-04)C2=0.
C=IVI*C
C1=IVI*C1
C2=IVI*C2
DO=1.-C*C
D(1)=(C*C2+C1*C1)/DO
D(2)=4*C*C1/DO
D(3)=D(1)+D(2)*C*C1/DO
D(4)=2*C*D(3)+C1*D(2)
D(5)=C2
D(6)=C1*C1/C
D(7)=2*C*C/DO
IF(JO.EQ.0)E2(K)=0.
C
C
C
EPRIM * EPRIM
CALLEPRIM(K, LAMBDA)
CALLEPRIM(K, MU)
IF(IVI.EQ.1)CALLHDEJ(NJ,C,D,2*C,UN, H)
IF(IVI.EQ.1)CALLEEDEXY(JO,LAMBDA,MU,1,NJ,X,Y,H,SXX,SXY,SYX,SY,
1, E2(K) )
H(9)=1.
CALLEEDEXY(JO,LAMBDA,MU,2,NJ,X,Y,H,SXX,SXY,SYX,SY, E2(K))
C
C
C
E * EPRIM
CALLE(AK, LAMBDA)
H(9)=D(2)
CALLEEDEXY(JO,LAMBDA,MU,2,NJ,X,Y,H,SXX,SXY,SYX,SY, E2(K))
IF(IVI.EQ.0)GOTO100
CALLEEDEXY(JO,LAMBDA,MU,NJ,X,Y,II,C,C1,IVI,EE(K), EE1(K))
IF(JO.EQ.1)E2(K)=E2(K)-(D(3)*EE(K)+D(2)*EE1(K) )* EE(K)
CALLHDEJ(NJ,C,D,2*C1,FEE1, H)
CALLEEDEXY(JO,LAMBDA,MU,1,NJ,X,Y,H,SXX,SXY,SYX,SY, E2(K))
C
C
C
C
E * E
CALLE(AK,MU)
CALLHDEJ(NJ,C,D,1.,FEE,H)
CALLEEDEXY(JO,LAMBDA,MU,1,NJ,X,Y,H,SXX,SXY,SYX,SY,E2(K))
H(9)=D(3)
CALLEEDEXY(JO,LAMBDA,MU,2,NJ,X,Y,H,SXX,SXY,SYX,SY, E2(K))
C
C
C
EPRIM * E
CALLEPRIM(K, LAMBDA)
CALLHDEJ(NJ,C,D,2*C1,FEE1, H)
CALLEEDEXY(JO,LAMBDA,MU,1,NJ,X,Y,H,SXX,SXY,SYX,SY, E2(K))

```

```

100 CONTINUE
    RETURN
    END

C CHARGEMENT DU VECTEUR PARAMETRE
SUBROUTINE E(AK, A)
    DIMENSION AK(1),A(1)
    A(1)=1.
    DO 100 L=1,3
100 A(L+1)=-AK(L)
    RETURN
    END

C CHARGEMENT DE LA DERIVEE DU VECTEUR PARAMETRE ,PAR RAPPORT A L'UN D'EUX
SUBROUTINE EPRIM(K, A)
    DIMENSION A(1)
    DO 100 L=1,4
100 A(L)=0.
    A(K+1)=-1.
    RETURN
    END

C CALCUL DE H DE J
SUBROUTINE HDEJ(NJ,C,D, HO,F,H)
    DIMENSION H(1),D(1)
    CC=HO
    DO 100 J=1, 8
    H(J)=CC*F(J,D)
100 IF(ABS(C).LT.1)CC=CC*C
    RETURN
    END

C (POUR HOMOGENEITE JJ), APPELEE EN EXTERNAL)
FUNCTION UN(J,D)
    UN=1.
    RETURN
    END

C FONCTION DE J DANS LE CALCUL DE H POUR LES TERMES EN E * E
FUNCTION FEE(J,D)
    DIMENSION D(1)
    FEE=D(4) + J * (D(5)+ D(6)*(J-1))
    RETURN
    END

C FONCTION DE J DANS LE CALCUL DE H POUR LES TERMES EN E * E1 ET E1 * E
FUNCTION FEE1(J,D)
    DIMENSION D(1)
    FEE1=J+D(7)
    RETURN
    END

C CALCUL DE E ET E1 (EGALEMENT FAIT PAR UNE AUTRE METHODE DANS RECVI)
SUBROUTINE EDEXY(JO,LAMBDA,MU,NJ,X,Y,II,C,C1,I,VI, EE,EE1)
    REAL LAMBDA,MU,LAO1,LA23,MUO1,MU23
    DIMENSION LAMBDA(1),MU(1),X(1),Y(1)
    EER=JO*EE
    EE=0.

```



```

EE1R=J0*EE1
EE1=0.
CC=1.
LA01=LAMBDA(2-J0)
LA23=LAMBDA(4-J0)
MU01=MU(2-J0)
MU23=MU(4-J0)
IIJ=II-2+J0
IF(IIJ.LE.0)IIJ=IIJ+NJ
IIJ1=IIJ-1
DO 100 J=1,7*IVI+1
IF(IIJ1.LE.0)IIJ1=IIJ1+NJ
EE=EE+(LA01*Y(IIJ)+LA23*X(IIJ))*CC
EE1=EE1+((LA01*Y(IIJ1)+LA23*X(IIJ1))*J*CC
      + MU01*Y(IIJ)+MU23*X(IIJ) )*CC
IIJ=IIJ1
IIJ1=IIJ1-1
100 IF(ABS(C).LT.1.)CC=CC*C
EE=EE+EER
EE1=EE1+EE1R
RETURN
END

C CALCUL DES TERMES DE LA DERIV. 2NDE EN E1 * E2 EN FONCTION DE X*X,X*Y,..
C (APPELE AVANT ET APRES INCREMENTATION DES SXX,SXY..)
SUBROUTINE EDEXY(J0,LAMBDA,MU,L,NJ,X,Y,H,SXX,SXY,SYX,SY, E2)
DIMENSION LAMBDA(1),MU(1),H(1),SXX(1),SXY(1),SYX(1),SY(1)
COMMON /VI/IVI
REAL LAMBDA,LA01,LA23,MU
LA01=LAMBDA(2-J0)
LA23=LAMBDA(4-J0)
IF(ABS(LA01)+ABS(LA23).LE.1.E-03) RETURN
IF(ABS(H(1))+ABS(H(2)).LE.1.E-04.AND.L.EQ.1) RETURN
DO 100 J=1-L,13-6*L
JH=IABS(J+1-J0)+8*L-7
JHU=IABS(J-J0)+8*L-7
JS=IABS(J)+8*L-7
HJ=H(JH)
HJO=H(JHO)
IF(JH.GT.6+2*L)HJ=0.
IF(JHO.GT.6+2*L)HJO=0.
IF(J.EQ.0.AND.JO.EQ.1.AND.L.EQ.1)HJO=0.
H1=MU(1)*HJ+MU(2)*HJO
H2=MU(3)*HJ+MU(4)*HJO
E2=E2+H1*LA01*SY(JS)+H2*LA23*SXX(JS)
IF(J.GE.0)E2=E2+H1*LA23*SXY(JS)+H2*LA01*SYX(JS)
IF(J.LT.0)E2=E2+H1*LA23*SYX(JS)+H2*LA01*SXY(JS)
100 CONTINUE
RETURN
END

C FILTRE AUTOREGRESSIF
SUBROUTINE FILTER(C,NJ,II, X)
C (MEMOIRE LIMITEE A 10 PAS DE TEMPS)
DIMENSION X(1)
FF=0.
FF1=0.
CC=1.

```

```

      IIJ=II
      J=1
100  IIJ=IIJ-1
      IF(IIJ.LE.0)IIJ=IIJ+NJ
      FF=FF+CC*X(IIJ)
      FF1=FF1+CC*(J-1)*X(IIJ)
      CC=CC*C
      J=J+1
      IF(ABS(CC).GT.1.E-10.AND.J.LE.NJ-1)          GOTO100
      I11=II-1
      IF(I11.LE.0)I11=I11+NJ
      X(11)=FF
      X(12)=X(II)+C*FF
      X(14)=X(I11)+FF1
      X(13)=FF1/C
      IF(ABS(C).GT.1.E-06)X(13)=FF1/C
      RETURN
      END

C  CONDITIONNEMENT DES VECTEURS UTILISES DANS L'EQUATION DU 2ND DEGRE
SUBROUTINE CONDIT(FJ,C1,  U,V,INDEF,LAMBDA,MU)
DIMENSION FJ(14,5),C1(1),U(1),V(1),LAMBDA(1),MU(1)
REAL LAMBDA,MU
INDEF=1
DO 100 L=1,3
  IL=(23+(-1)**L)/2
  JL=L/2+1
  U(L)=FJ(IL,JL)
  V(L)=FJ(IL+2,JL)
  G=FJ(12,2+L)
  IF(ABS(G).LT.1.E-04)          RETURN
  LAMBDA(L)=-U(L)/G
100 MU(L)=-C1(L)/G
  INDEF=0
  LAMBDA(4)=0.
  MU(4)=0.
  U(4)=FJ(12,1)
  V(4)=FJ(14,1)
  RETURN
  END

C  RESOUD EQUATION VECTORIELLE A=A0 + ( UT.A )( LAMBDA + (VT.A) MU )
C  (DONNE DA = A-A0 )
SUBROUTINE E2DVEC(N,AG,U,V,LAMBDA,MU,  DA)
DIMENSION Z(6,6),Z0(6,6),Z1(6,6)
REAL LAMBDA,MU,M1
DIMENSION A0(1),U(1),V(1),LAMBDA(1),MU(1),DA(1)
DIMENSION A1(6),A2(6),A3(6),M1(6)
CALLSCAL(N,V,A0,  EPS1)
CALLSCAL(N,U,A0,  EPS)
DO 100 L=1,N
  A1(L)=LAMBDA(L)+MU(L)*EPS1
100 A2(L)=MU(L)*EPS
  CALL INV(N,A1,U,A2,V,6,  Z)
  CALL MATVEC(N,6,Z,A1,  A3)
  CALL MATVEC(N,6,Z,MU,  M1)
  CALL SCAL(N,U,M1,  P)
  CALL SCAL(N,V,M1,  P1)

```

```

      B=(1.-EPS*(P+P1))/2.
      A=P*P1
      CALL SCAL(N,U,A3, P)
      CALL SCAL(N,V,A3,C)
      C=C*P*EPS*EPS
      D=1.-A*C/B/B
      EEEE=B/A
      IF(D.GT.0.)EEEE=EEEE*(1.+SQRT(D))
      EEE=C/B
      IF(ABS(EEEE).GT.100.)EEEE=0.
      DO 200 L=1,N
200  A1(L)=A1(L)*EPS+ MU(L)*EEEE
      CALL MATVEC(N,6,Z,A1, DA)
      RETURN
      END

C  PRODUIT MATRICE * VECTEUR
  SUBROUTINE MATVEC(N,NDIM,Z,A, B)
  DIMENSION Z(NDIM,1),A(1),B(1)
  DO 100 I=1,N
  CALL SCAL(N, Z(I,1),A, P)
100  B(I)=P
  RETURN
  END

C  PRODUIT SCALAIRE
  SUBROUTINE SCAL(N,X,Y, P)
  DIMENSION X(1),Y(1)
  P=0.
  DO 100 I=1,N
100  P=P+X(I)*Y(I)
  RETURN
  END

C  INCREMENTE PRODUIT TENSORIEL
  SUBROUTINE TENS(N,R,X,Y,NDIM, Z)
  DIMENSION X(1),Y(1),Z(NDIM,1)
  DO 100 I=1,N
  DO 100 J=1,N
100  Z(I,J)=Z(I,J)+R*X(I)*Y(J)
  RETURN
  END

C  INVERSE MATRICE I - A.UT - B.VT
  SUBROUTINE INV(N,A,U,B,V,NDIM, Z)
  DIMENSION A(1),U(1),B(1),V(1),Z(NDIM,1)
  CALL SCAL(N,V,B, VB)
  CALL SCAL(N,U,B, UB)
  CALL SCAL(N,V,A, VA)
  CALL SCAL(N,U,A, UA)
  DO 100 J=1,N
  DO 100 I=1,N
100  Z(I,J)=0.
  CALL TENS(N,1.-VB,A,U,6, Z)
  CALL TENS(N,UB ,A,V,6, Z)
  CALL TENS(N,1.-UA,B,V,6, Z)
  CALL TENS(N,VA ,B,U,6, Z)
  D=(1.-UA)*(1.- VB)-UB*VA

  DO 300 I=1,N
  DO 200 J=1,N
  200  Z(I,J)=Z(I,J)/D
  300  Z(I,I)=1.+Z(I,I)
  RETURN
  END

```

EOF HIT