Tests d'indépendance en analyse multivariée et tests de normalité dans les modèles ARMA

par

Pierre LAFAYE DE MICHEAUX

Thèse de doctorat effectuée en cotutelle
au

Département de mathématiques et de statistique

Faculté des arts et des sciences

Université de Montréal

ET

Département des sciences mathématiques

Formation Doctorale Biostatistique

École Doctorale Information, Structures, Systèmes

Université Montpellier II
Sciences et Techniques du Languedoc

Thèse présentée à la Faculté des études supérieures de l'Université de Montréal
en vue de l'obtention du grade de Philosophiæ Doctor (Ph.D.) en statistique

et à

l'Université Montpellier II en vue de l'obtention du grade de Docteur
d'Université en Mathématiques appliquées et applications des mathématiques

décembre 2002

Université de Montréal
Faculté des études supérieures
et
Université Montpellier II
Laboratoire de Probabilité et Statistiques

Cette thèse de doctorat effectuée en cotutelle et intitulée

Tests d'indépendance en analyse multivariée et tests de normalité dans les modèles ARMA

a été présentée et soutenue publiquement à l'Université de Montréal par

Pierre LAFAYE DE MICHEAUX

Elle a été évaluée par un jury composé des personnes suivantes :

Président-rapporteur *Université de Montréal*	*Cléroux R., Professeur, Département de mathématiques et de statistique*
Directeur de recherche *Université de Montréal*	*Bilodeau M., Professeur, Département de mathématiques et de statistique*
Directeur de recherche *Université Montpellier II*	*Ducharme G., Professeur, Département des sciences mathématiques*
Examinateur externe *University of Toronto*	*Feuerverger A., Professor, Department of Statistics*
Examinateur externe* *Université de Marne la Vallée*	*Diebolt J., Professeur, Laboratoire d'Analyse et de Mathématiques Appliquées*
Membre du jury *Université Montpellier II*	*Escoufier Y., Professeur, Département des sciences mathématiques*
*Représentant de l'examinateur externe	Roch Roy, Professeur, Département des sciences mathématiques, Université de Montréal
Représentant du doyen de la FES, *Université de Montréal*	*Cléroux R., Professeur, Département de mathématiques et de statistique*

Thèse acceptée le:
16 décembre 2002

أَلنَجَابَةُ أَلجَدِيدَةُ هِي أَلمَعرِفَةُ

L’art de la connaissance et l’exercice des vertus,
éternelle noblesse du chercheur.

À ma femme Dominique Delseny et notre fils Luka.

À mon meilleur ami Fabien Baudrier.

À mes parents.

REMERCIEMENTS

Les premières personnes que je tiens à honorer ici sont mes deux directeurs de recherche Martin Bilodeau et Gilles Ducharme.

Je considère Gilles Ducharme comme mon père spirituel en statistique ; il m'a appris l'essentiel de ce que j'en sais. Depuis son arrivée à Montpellier, période à laquelle j'ai entamé mes études dans ce domaine, j'ai commencé à apprécier son approche innovante de la statistique au travers de ses enseignements beaucoup plus intuitifs que lourdement formels. Il m'a ensuite initié à la recherche lors de mon année passée en DEA. Après avoir goûté à sa grande expérience, à l'originalité de ses sujets de recherche et à la pertinence de son jugement pour en maîtriser les difficultés techniques menant à leur résolution, j'ai replongé avec plaisir pour quelques années supplémentaires ! Il m'a enseigné un métier passionnant. Je lui suis aussi reconnaissant pour son soutien moral dans certaines périodes difficiles, pour les contributions financières et matérielles apportées et pour la chance qu'il m'a donnée de pouvoir travailler avec un chercheur de grande valeur à Montréal.

Martin Bilodeau a pris le relais avec beaucoup d'attention et de professionnalisme. J'ai apprécié sa constante fiabilité, sa précieuse disponibilité et sa discrétion. Je lui suis également reconnaissant pour les connaissances scientifiques qu'il m'a transmises et pour les ressources financières dont j'ai bénéficié. A son contact d'une grande probité intellectuelle, j'ai pu apprécier et m'imprégner de son souci d'efficacité et de concision. Je veux lui témoigner ici toute mon estime.

Je tiens à exprimer ma reconnaissance aux membres du jury qui ont accepté de prendre du temps pour lire mon travail, pour leurs conseils judicieux et leurs remarques pertinentes ainsi que pour s'être déplacés pour assister à ma soutenance, ce rite d'acceptation dans une nouvelle communauté.

Merci aux chercheurs avec qui j'ai pu avoir des discussions scientifiques profitables tant à Montpellier (Benoît Cadre, Michel Cuer, Pierre Jacob, Irène Larramendy-Valverde) qu'à Montréal (Jean-Francois Angers, Anne Bourlioux, Richard Duncan, Marlene Frigon, Martin Goldstein, Michel Grundland, Anatole Joffe, Christian Léger, Jean-Marc Lina, Urs Maag, Éric Marchand, François Perron, Roch Roy), sans oublier ceux plus anonymes avec qui j'ai échangé des idées sur Internet.

Il est clair aussi que le travail de recherche effectué au cours d'un doctorat doit s'appuyer sur des bases techniques solides. Les principaux acteurs de l'excellente formation en statistique que j'ai suivie sont Martin Bilodeau, Yves Lepage et François Perron à Montréal et Denis Allard, Alain Berlinet, Gilles Caraux, Jean-Pierre Daures, Gilles Ducharme, Jean-François Durand, Ali Gannoun, Pierre Jacob, Jean-Dominique Lebreton, Pascal Monestiez, Robert Sabatier, Gilberte Vignau et Jean-Pierre Vila à Montpellier.

J'ai bénéficié pendant ces « quatre » années, dont trois consacrées à la recherche, d'un précieux soutien informatique de la part de Christopher Albert, Nicolas Beauchemin, Miguel Chagnon, Baptiste Chapuisat, Brigitte Charnomordic, Marc Fangeaud, Michel Lamoureux, Nathalie Malo, Pascal Neveu, Philippe Vismara ainsi que de Leslie Lamport, Linus Torvalds et des milliers de bénévoles oeuvrant à la conception et au développement de Latex, Linux et autres logiciels libres d'excellence.

Je veux également souligner l'appui administratif de qualité apporté par Jacques Bélair, Robert Cléroux, Véronique Hussin, Sabin Lessard, Thérèse Ouellet, Danièle Proulx, Jacqueline Reggiori-Lamoureux, Yvan Saint-Aubin, Danièle Tessier et Janet Zaki à l'Université de Montréal et par Pierrette Arnaud, Michel Averous, Yves Escoufier, Daniel Guin, Bernadette Lacan, Marie-Odile Morda, Florence Picone et Véronique Sals-Vettorel à Montpellier.

Tout ce travail de recherche aurait été certainement plus long, pénible et de moins bonne qualité, si je n'avais pas bénéficié de plusieurs financements octroyés généreusement par le Département de mathématiques et de statistique et la Faculté des Études Supérieures de l'Université de Montréal, par l'Université de Montréal, par l'Institut des Sciences Mathématiques de Montréal ainsi que par mes deux directeurs de recherche Martin Bilodeau et Gilles Ducharme. Merci au Gouvernement du Canada et à celui du Québec pour m'avoir donné ma chance.

Toute ma gratitude aux personnes rencontrées à l'Université Montpellier II et dont une bonne partie m'ont offert leur amitié : Gérard Biau pour ses précieux conseils et son amitié depuis l'année de DEA, Sandie Ferrigno pour son soutien au site du DEA, Benoît Frichot pour les intéressantes discussions et son soutien moral, Olivier Gimenez (il y aurait trop à dire ici), Mariem Mint-el-Mouvid pour les bons moments passés dans notre bureau 14 malgré nos difficultés financières, Nicolas Molinari pour ses précieux conseils et les bons moments passé à Sauve. Une pensée à Omar Anza, Élodie Brunel, André Diatta, Bénédicte Fontez, Laurent Gardes et Hassan Mouhanad. Toutes ces personnes ont contribué, à leur degré, à animer l'espace de création du bâtiment 9. Merci aussi à Alain Delcamp, Ali Gannoun et Jérome Saracco pour les bons moments passés au 3ème étage et à Stéphane Girard et Cécile Amblard pour leur accueil très chaleureux à Montréal.

Enfin merci à tous mes nouveaux amis rencontrés à Montréal : Yves Atchade (Bénin), Chafik Bouhaddioui (Maroc), Alain Desgagné (Québec), Alexandre Leblanc (Québec), Ghislain Rocheleau (Québec), Ndeye Rokhaya Gueye (Sénégal) qui ont fait une partie de leur doctorat en même temps que moi ; et aussi Christopher Albert (USA)

et Carole (France), Marie-Soleil Beaudoin (Québec), Pascal Bouchard (Québec), Jean-Francois Boudreau (Québec), Pascal Croteau (Québec), Alexandre Cusson (Québec), Alina Dyachenko (Russie), Alexis Gerbeau (Québec), Mohammed Haddou (Algérie), Hassiba et Djamal Hellal (Algérie), Abdelaziz Khatouri (Maroc), Vincent Lemaire (France), Nathalie Malo (Québec), Hacène Nedjar (Algérie), Philippe Paradis (Québec), Fritz Pierre (Haiti), Alice Savu (Roumanie), Jib et Sarah. Vous avez facilité mon intégration. J'espère garder des liens solides avec la plupart d'entre vous.

TABLE DES MATIÈRES

Remerciements 5

Liste des tableaux 11

Liste des figures 12

Sigles et abréviations 13

Chapitre 1. Introduction 14

Bibliographie 23

Chapitre 2. Goodness-of-fit tests of normality for the innovations in ARMA models 24

1. Introduction 25

2. Smooth test of normality in the ARMA context 28

3. Choosing the order K of the alternative 31

4. Simulation Results 32

4.1. Levels 33

4.2. Power 39

5. An example 41

Appendix A 41

Appendix B 43

Appendix C 44

Appendix D 45

Bibliography 47

Javascript application 49

Chapitre 3. A multivariate empirical characteristic function test of independence with normal marginals 50

1. Introduction 51

2. Testing independence: the non-serial situation 53

2.1. The case of known parameters 53

2.2. The case of unknown parameters 54

2.3. Relation to V-statistics . . . 56
2.4. Consistency . . . 56
3. Testing independence: the serial situation . . . 57
3.1. The case of known parameters . . . 57
3.2. The case of unknown parameters . . . 57
4. Properties of the limiting processes . . . 59
5. One-way MANOVA model with random effects . . . 64
6. Proofs . . . 66
6.1. Proof of Theorem 2.1 . . . 66
6.2. Proof of Theorem 2.2 . . . 67
6.3. Proof of Theorem 2.3 . . . 68
6.4. Proof of Theorem 2.4 . . . 69
6.5. Proof of Theorem 3.1 . . . 71
6.6. Proof of Theorem 3.2 . . . 71
6.7. Proof of Theorem 3.3 . . . 72
Acknowledgements . . . 72
Bibliography . . . 73
Chapitre 4. Conclusion . . . 75
Bibliographie . . . 77
Index des Auteurs . . . 78
Annexe A. Les Programmes Fortran 77 du premier article . . . 80
A.1. Le script *compile* . . . 83
A.2. Liste des différents programmes . . . 84
A.3. Moyennes et variances empiriques des différentes lois . . . 86
A.4. Le programme MAIN . . . 87
A.5. Les programmes big_prog_ARMApq . . . 90
A.6. Les programmes creerdat_ARMApq . . . 159
A.7. Le programme calcstat.f . . . 195
A.8. Les programmes simulARMApq.f . . . 214
A.9. Les programmes pour la “random shock method” de Burn . . . 227
A.10. Les programmes pour la simulation de différentes lois . . . 234
A.11. Les programmes pour les différents polynômes de Legendre . . . 248
A.12. Les autres petits programmes utiles . . . 277
A.13. Le programme du calcul des quantiles du test de Brockwell et Davis . 280
Annexe B. Les Programmes C++ du deuxième article . . . 282
B.1. Quantiles théoriques . . . 283
B.2. Quantiles empiriques . . . 342

B.3. Puissance comparée avec celle du test de Wilks 348

Curriculum Vitae 354

Documents spéciaux 355

LISTE DES TABLEAUX

2.3.1 Some quantiles obtained from approximation (2) 32
2.4.1 Distribution of the empirical p-values for the tests $\mathcal{R}_3$ and $\mathcal{R}_{\hat{K}}(2)$ 35
2.4.2 Distribution of the empirical p-values of various tests 38
2.4.3 Empirical power of various tests when $T = 50$ 40

3.4.1 Critical Values of the Distribution of $T_{b,A}$ for $b = 0.1$. 61
3.4.2 Distribution of $T^*_{b,A}$ for $b = 0.1$. 62
3.4.3 Empirical Percentage Points of $nT_{n,b,A}$: non-serial case 63
3.4.4 Empirical Percentage Points of $nT_{n,b,A}$: serial case 64
3.5.1 MANOVA table. 65
3.5.2 Empirical power of $nT_{n,b,A}$ and Wilks test 65

LISTE DES FIGURES

1.1.1 10th Century time line (Funkhauser, 1936, pp 260-262) 15

SIGLES ET ABRÉVIATIONS

Permet de se déplacer dans la séquence des pages consultées :
GB : GO BACK - Vers la dernière page consultée.
GF : GO FORWARD - Vers la prochaine page consultée.

Autres fonctionnalités :
⏮ : Première page.
◀ : Page précédente.
◇ : Plein écran.
⊖ : Pleine largeur d'écran.
⊙ : Vraie taille.
▶ : Page suivante.
⏭ : Dernière page.
☒ : Quitter.

$\xrightarrow{L}$: Convergence en loi.
$\xrightarrow{fd}$: Convergence des lois de dimension finie.
$\Rightarrow$: Convergence faible.
ARMA : Auto-Regressive Moving Average.
i.i.d. : Indépendant(e)s et identiquement distribué(e)s.
$\equiv$: signe d'égalité fonctionnelle.
$\langle \cdot\,, \cdot \rangle$: Produit scalaire.
Ω : Espace des évenements.
P : Mesure de probabilité sur l'espace des évenements.
$\pi_{\boldsymbol{t}_1,\ldots,\boldsymbol{t}_k}$: Projection.
$C \equiv C(\mathbb{R}^p, \mathbb{C})$: Espace des fonctions continues de $\mathbb{R}^p$ dans $\mathbb{C}$.
ρ : Métrique sur $C(\mathbb{R}^p, \mathbb{C})$.
m.l.e. : Maximum likelihood estimator.
$^{\mathsf{T}}$: Transposée d'un vecteur.
MANOVA : Multivariate analysis of variance.
EDF : Empirical distribution function.
CF : Cubature formula.

CHAPITRE 1

Introduction

Une procédure consistant à déterminer si un modèle probabiliste particulier est approprié pour un phénomène aléatoire donné...

Depuis le tout début de la statistique, nombre de statisticiens ont commencé leur analyse en proposant une distribution pour leurs observations et ont ensuite tenté de vérifier si leur distribution était la bonne. Ainsi, au fil des ans, un grand nombre de telles procédures sont apparues commençant alors à constituer un vaste champ d'études portant le nom de « tests d'ajustement ». Il est important de noter que, si l'on ne disposait pas des outils que sont les tests d'ajustement, il faudrait se baser sur des critères subjectifs pour valider la qualité d'un modèle. Malheureusement, comme l'a si bien souligné Fisher (1925)

> *No eye observation of such diagrams, however experienced, is really capable of discriminating whether or not the observations differ from the expectation by more than we would expect from the circumstances of random sampling.*

L'apport des tests d'ajustement est triple. Il permet d'obtenir une description compacte des données en leur attribuant une loi de probabilité. Ensuite, certaines techniques paramétriques puissantes sont valides uniquement sous l'hypothèse de normalité. Enfin, cela permet de mieux comprendre le mécanisme ayant généré les données, en obtenant de l'information sur les raisons ayant conduit à un rejet de l'hypothèse de travail.

Mathématiquement, le problème se présente de la façon suivante. Soit Y un élément aléatoire dont la fonction de répartition F est absolument continue par rapport à la mesure de Lebesgue. On désire tester l'hypothèse

$$H_0 : F(x) \in \{F_0(x, \boldsymbol{\theta}), \boldsymbol{\theta} \in \Theta\} \text{ versus } H_1 : F(x) \notin \{F_0(x, \boldsymbol{\theta}), \boldsymbol{\theta} \in \Theta\},$$

où Θ est un certain espace paramétrique. Pour ce problème, on peut distinguer deux grandes classes de tests. Les tests « omnibus » concernent les situations où l'on n'a

a priori aucune indication sur la façon dont la distribution réelle F pourrait s'écarter de l'hypothèse nulle. Ils sont efficaces contre des alternatives non spécifiées et sont généralement basés sur la fonction de répartition expérimentale ou sur la fonction caractéristique expérimentale. On peut citer par exemple les tests de Kolmogorov, d'Anderson-Darling ou de Cramér-von Mises. Les tests « directionnels » quant à eux permettent de prendre en compte certaines informations sur les écarts les plus plausibles à l'hypothèse nulle. Ils sont construits de façon à détecter avec plus de puissance certains types d'orientation que pourrait prendre la distribution de Y. Au cours de la première partie de la recherche envisagée, on s'est intéressé à la seconde classe en adaptant la théorie des tests lisses, introduite par Neyman (1937), au contexte particulier de données dépendantes, issues d'une loi non entièrement spécifiée.

En effet, une autre branche importante et originale de la statistique concerne l'analyse des séries chronologiques. Sa caractéristique essentielle réside en une dépendance des phénomènes étudiés vis-à-vis du temps, concept essentiel tant au niveau scientifique que philosophique. Il est de fait peu de disciplines qui ne soient confrontées à l'étude de variables évoluant dans le temps et qu'on désire décrire, expliquer, contrôler ou prévoir. Cette discipline puise ses origines dans le Moyen Âge comme en témoigne ce diagramme temporel (représentant l'inclinaison des orbites de planètes en fonction du temps) considéré comme l'un des plus anciens du monde occidental.

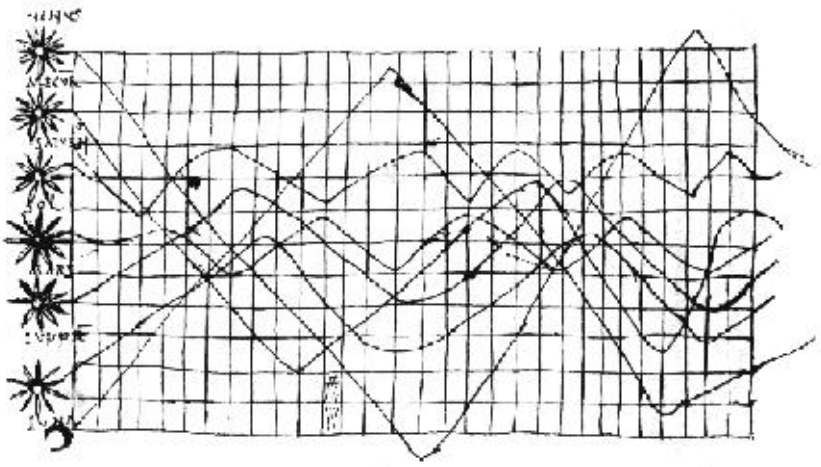


FIG. 1.1. 10th Century time line (Funkhauser, 1936, pp 260-262)

Une avancée importante dans l'étude des séries temporelles a été de supposer que la série chronologique observée est engendrée par un processus stochastique $\{Y_t, t \in \mathbb{Z}\}$. Une condition souvent imposée sur ce processus générateur est qu'il soit stationnaire du second ordre. Un processus digne d'intérêt satisfaisant à ces conditions est le processus autorégressif à moyenne mobile (ARMA). Ce processus est très utilisé du fait de sa simplicité. Son introduction nécessite quelques définitions préalables.

Définition 1. *Stationnarité faible*
Un processus $\{Y_t; t \in \mathbb{Z}\}$ est dit stationnaire au second ordre, ou stationnaire au sens faible, ou stationnaire d'ordre deux si les trois conditions suivantes sont satisfaites :

– $\forall t \in \mathbb{Z}, \quad E(Y_t^2) < \infty$,

– $\forall t \in \mathbb{Z}, \quad E(Y_t) = m$, *indépendant de* t,
– $\forall (t,h) \in \mathbb{Z}^2, \quad cov(Y_t, Y_{t+h}) = E[(Y_{t+h} - m)(Y_t - m)] = \gamma(h)$, *indépendant de* t.

Nous avons aussi besoin de la notion de bruit blanc (ou white noise).

Définition 2. *Bruit blanc*
Un processus $\{\epsilon_t; t \in \mathbb{Z}\}$ *est un bruit blanc s'il satisfait aux deux conditions suivantes* $\forall t \in \mathbb{Z}$ *:*

– $E(\epsilon_t) = 0$,
– $\gamma(h) = E[\epsilon_t \epsilon_{t-h}] = \begin{cases} \sigma^2, \ h = 0, \\ 0, \ h \neq 0. \end{cases}$

En s'appuyant sur ces deux définitions, on peut introduire le modèle ARMA.

Définition 3. *Modèle ARMA(p,q)*
On appelle processus autorégressif à moyenne mobile d'ordre (p,q) un processus stationnaire $\{Y_t; t \in \mathbb{Z}\}$ *vérifiant une relation du type*

$$Y_t - \sum_{i=1}^{p} \varphi_i Y_{t-i} = \sum_{i=1}^{q} \theta_i \epsilon_{t-i} + \epsilon_t \ , \forall t \in \mathbb{Z}$$

où les φ_i *et les* θ_i *sont des réels et où l'erreur* $(\epsilon_t; t \in \mathbb{Z})$ *est un bruit blanc de variance* σ^2.

Dans la démarche de modélisation d'une série temporelle, il est courant de suivre la procédure indiquée par Box et Jenkins (1976). Cette procédure se déroule en cinq étapes : « Stationnarisation », « Désaisonnalisation », « Identification », « Validation et Tests », « Prévisions ».

L'attention dans la première partie de cette étude s'est portée sur la phase de « Validation et Tests », et plus particulièrement sur la construction d'un test lisse de normalité pour les erreurs d'un modèle ARMA(p,q) univarié complètement identifié, de moyenne connue.

Pour cela, la statistique du *score de Rao* est un outil classique pour la construction de tests d'hypothèses de la forme $H_0 : \boldsymbol{\eta} = \boldsymbol{\eta}_0$. Le test qui en découle est basé sur le principe général que le vecteur gradient du modèle non restreint (par H_0), évalué en l'estimateur restreint, suit asymptotiquement une loi normale de moyenne 0, si H_0 est vraie. Si l'hypothèse alternative est décrite par une certaine famille exponentielle de dimension K, le test du score résultant est aussi appelé test lisse (*smooth test*) ou test lisse de Neyman (1937) d'ordre K. L'idée du test lisse est en fait d'emboîter la fonction de densité de l'hypothèse nulle dans une famille paramétrique plus générale. Cette famille doit être choisie pour détecter les alternatives les plus probables si l'hypothèse nulle est fausse. De cette façon, l'hypothèse nulle devient une hypothèse paramétrique où l'on cherche à tester la nullité de paramètres de la densité alternative. Cette approche fournit non seulement une statistique de test simple, mais aussi une bonne puissance pour une vaste famille d'alternatives. Considérons un échantillon $\epsilon_1, \ldots, \epsilon_n$ de variables aléatoires continues *i.i.d.* ayant pour fonction de répartition F et pour lequel on souhaite tester l'hypothèse $H_0 : F = F_0$. Il est possible de transformer

cet échantillon en un échantillon $U_1 = 2F_0(\epsilon_1) - 1, \ldots, U_n = 2F_0(\epsilon_n) - 1$ de variables aléatoires *i.i.d.* de densité g qui, sous H_0, sont de loi uniforme sur l'intervalle [-1,1]. Utilisant cette propriété, Neyman (1937) propose de considérer l'hypothèse

$$H_0 : g(y) = \begin{cases} 1/2 & \text{si } y \in [-1, 1], \\ 0 & \text{sinon} \end{cases} \tag{1}$$

et de choisir pour cette loi uniforme la famille alternative d'imbrication de fonctions de densité $c.\exp[P(y)]$ où $P(y)$ représente un polynôme et c est la constante de normalisation. Il appelle de telles densités des alternatives « lisses » car elles sont représentées graphiquement par des courbes lisses coupant la densite sous H_0 un petit nombre de fois. Le fait de pouvoir se restreindre, par des considérations physiques ou en faisant appel à son intuition, à ce type d'alternatives permet d'obtenir un test plus sensible. Ensuite, il suggère de choisir le polynôme $P(y)$ comme étant une combinaison linéaire des éléments d'un des systèmes de polynômes orthonormaux $\pi_0(y), \pi_1(y), \ldots$ sur l'intervalle [-1,1]. Rappelons que, dans un tel système, le polynôme $\pi_i(y)$ est de degré i, et que pour tout i et $j = 0, 1, 2, \ldots$, on a

$$1/2 \int_{-1}^{1} \pi_i(y)\pi_j(y)dy = \delta_{ij}$$

où

$$\delta_{ij} = \begin{cases} 0 & \text{si } i \neq j, \\ 1 & \text{si } i = j. \end{cases}$$

Ces systèmes de polynômes orthonormaux peuvent être construits de plusieurs façons (Marsden, 1974, p. 347). Neyman (1937), pour sa part, utilise le système de polynômes de Legendre. La famille d'imbrication s'écrit alors

$$g_K(y, \boldsymbol{\eta}) = \begin{cases} C(\boldsymbol{\eta}) \exp\left(\sum_{i=1}^{K} \eta_i \pi_i(y)\right) & \text{si } y \in [-1, 1], \\ 0 & \text{sinon.} \end{cases} \tag{2}$$

où $\boldsymbol{\eta} = (\eta_1, \ldots, \eta_k)^t \in \mathbb{R}^K$, $C(\boldsymbol{\eta})$ est la constante de normalisation, K est appelé l'ordre de l'alternative et les $\pi_i(\cdot), i = 1, \ldots, K$ sont les K premiers polynômes de Legendre. Nous dénotons par $g_0(\cdot)$ la fonction de densité sous l'hypothèse nulle, c'est-à-dire la distribution uniforme sur l'intervalle [-1,1]. Ainsi, éprouver l'hypothèse nulle (1) contre l'hypothèse voulant que la fonction de densité appartienne à la famille d'imbrication $H_1 : g(\cdot) \in \{g_k(\cdot, \boldsymbol{\eta}) : \boldsymbol{\eta} \in \mathbb{R}^K\}$, donnée en (2), est donc équivalent à éprouver l'hypothèse

$$H_0 : \boldsymbol{\eta} = 0 \text{ versus } H_1 : \boldsymbol{\eta} \neq 0. \tag{3}$$

Il suffit alors de construire le test du *score* pour l'hypothèse (3). On définit le vecteur du *score* par

$$\boldsymbol{a}_{n\boldsymbol{\eta}} = \left(\frac{1}{n} \sum_{i=1}^{n} \frac{\partial Logg_K(Y_i, \boldsymbol{\eta})}{\partial \eta_1}, \ldots, \frac{1}{n} \sum_{i=1}^{n} \frac{\partial Logg_K(Y_i, \boldsymbol{\eta})}{\partial \eta_K} \right)^{\mathsf{T}}.$$

Alors la statistique du score de Rao (1947) est

$$R_K = n\boldsymbol{a}^{\mathsf{T}}_{n\boldsymbol{\eta}_0} I^{-1}_{\boldsymbol{\eta}_0} \boldsymbol{a}_{n\boldsymbol{\eta}_0}$$

où

$$I_{\boldsymbol{\eta}} = \left[E_{\boldsymbol{\eta}} \left\{ \frac{\partial Logg_K(Y, \boldsymbol{\eta})}{\partial \eta_i} \cdot \frac{\partial Logg_K(Y, \boldsymbol{\eta})}{\partial \eta_j} \right\} \right]_{K \times K}$$

est la matrice d'information de Fisher de $\boldsymbol{\eta}$. On peut ensuite montrer que, sous H_0, $R_K \xrightarrow{L} \chi^2_K$, loi du khi-deux à K degrés de liberté.

Cette présentation du test de Neyman (1937) repose sur l'hypothèse que F_0 est entièrement spécifiée. Une généralisation au cas où un ou plusieurs paramètres seraient inconnus a été faite par Kopecky et Pierce (1979) et Rayner et Best (1988).

Remarque 1. *Une justification théorique que l'on peut offrir pour le choix de la famille d'imbrication précédente repose sur la théorie des espaces de Hilbert (Royden, 1968, chap.10 sec.8). En effet, on peut montrer que toute fonction h dans l'espace de Hilbert des fonctions de carré intégrable par rapport à $g_0(y)$ peut s'écrire*

$$h(y) = \sum_{i=0}^{\infty} \eta_i \pi_i(y).$$

Par conséquent, la vraie distribution de Y peut s'écrire

$$f(y) = \begin{cases} c.exp\left(\sum_{i=0}^{\infty} \eta_i \pi_i(y) \right) & \text{si } y \in [-1, 1], \\ 0 & \text{sinon,} \end{cases}$$

et la famille d'imbrication proposée est une approximation de la vraie densité, d'autant meilleure que K est grand.

Il est possible d'adapter ce qui précède au cas de données issues d'un processus de type ARMA, c'est l'objet du Chapitre 2.

Le Chapitre 3 est consacré à un problème connexe. On pourrait en effet supposer, dans le même ordre d'idée de la problématique précédente, que les erreurs ϵ_t du modèle ARMA sont Gaussiennes et se demander si ces erreurs sont engendrées par un processus bruit blanc ou, ce qui est équivalent dans ce cas de figure, s'il y a indépendance sérielle entre les ϵ_t. Dans l'optique d'une future généralisation des résultats du Chapitre 2 au cas d'un ARMA multivarié, il semble intéressant de construire un test d'indépendance sérielle pour des vecteurs aléatoires. De plus, afin de mieux appréhender la complexité du problème, il s'avère avantageux d'échelonner la réponse à cette question en commençant par bâtir un test d'indépendance dans le cas non sériel.

La construction de ce test non paramétrique s'appuie sur une caractérisation de l'indépendance introduite par Ghoudi et al. (2001) et le résultat obtenu est une statistique de type Cramér-von Mises d'un certain processus empirique. Ghoudi et al. (2001) ont défini leur processus en utilisant la fonction de répartition empirique. Ici, le processus empirique est basé sur la fonction caractéristique empirique multivariée. Pour être plus formel, considérons le vecteur aléatoire $\boldsymbol{\epsilon} = (\boldsymbol{\epsilon}^{(1)}, \ldots, \boldsymbol{\epsilon}^{(p)})$, constitué de p sous-vecteurs de dimension q et le vecteur $\boldsymbol{t} = (\boldsymbol{t}^{(1)}, \ldots, \boldsymbol{t}^{(p)})$ partitionné de la même façon. En outre, pour tout $A \subset \{1, \ldots, p\}$ de cardinalité supérieure à 1, introduisons

la fonction μ_A définie par

$$\mu_A(\boldsymbol{t}) = \sum_{B \subset A} (-1)^{|A \setminus B|} C_p(\boldsymbol{t}^B) \prod_{j \in A \setminus B} C^{(j)}(\boldsymbol{t}^{(j)})$$

avec

$$(\boldsymbol{t}^B)^{(i)} = \begin{cases} \boldsymbol{t}^{(i)} & i \in B, \\ \mathbf{0} & i \in I_p \setminus B. \end{cases}$$

L'objet C_p est la fonction caractéristique conjointe de $\boldsymbol{\epsilon}$ et les $C^{(j)}$ sont les fonctions caractéristiques des marginales $\boldsymbol{\epsilon}^{(j)}$. Il est alors possible de montrer que $\boldsymbol{\epsilon}^{(1)}, \ldots, \boldsymbol{\epsilon}^{(p)}$ sont indépendants si et seulement si $\mu_A \equiv 0$. Disposant d'un échantillon $\boldsymbol{\epsilon}_1, \ldots, \boldsymbol{\epsilon}_n$, cela amène assez naturellement à définir le processus $R_{n,A}$, sous l'hypothèse de multinormalité des $\boldsymbol{\epsilon}_i^{(j)}$, par

$$R_{n,A}(\boldsymbol{t}) = \sqrt{n} \sum_{B \subset A} (-1)^{|A \setminus B|} \phi_{n,p}(\boldsymbol{t}^B) \prod_{i \in A \setminus B} \phi(\boldsymbol{t}^{(i)}) \tag{4}$$

où

$$\phi_{n,p}(\boldsymbol{t}) = \frac{1}{n} \sum_{j=1}^{n} \exp(\mathrm{i}\langle \boldsymbol{t}, \boldsymbol{\epsilon}_j \rangle)$$

est la fonction caractéristique empirique de l'échantillon et ϕ est la fonction caractéristique d'une loi normale $N_q(\mathbf{0}, \boldsymbol{I})$. La statistique de Cramér-von Mises et le test qui en découle seront construits à partir de ce processus. Introduisons maintenant quelques définitions et notations utiles pour la suite portant sur la convergence faible d'éléments aléatoires dans un espace métrique S.

Définition 4. *Convergence en loi*
La loi de X est par définition la mesure de probabilité $P = \mathsf{P}X^{-1}$ sur $(S, \mathcal{S})$:

$$P(A) = \mathsf{P}(X^{-1}(A)) = \mathsf{P}\{\omega \in \Omega; X(\omega) \in A\} = \mathsf{P}\{X \in A\},\ A \in \mathcal{S}.$$

On dit qu'une suite d'éléments aléatoires $\{X_n\}$ converge en loi vers un élément aléatoire X sur S, et on écrit $X_n \xrightarrow{L} X$, si les lois P_n des X_n convergent faiblement vers la loi P de X et on note $P_n \Rightarrow P$.

Le théorème qui suit (Billingsley, 1968, p. 30) est très utile dans la pratique puisqu'il permet de prouver la convergence faible des mesures induites sur $\mathbb{R}$ par différentes fonctions réelles h à partir de la convergence faible dans des espaces métriques généraux.

Théorème 1. *Soit $h : S \longrightarrow S'$ mesurable, et soit D_h l'ensemble des points de discontinuité de h.*
Si $P_n \Rightarrow P$ et $P(D_h) = 0$, alors $P_n h^{-1} \Rightarrow P h^{-1}$.

On va maintenant donner une caractérisation pratique de la convergence faible mais pour cela on a préalablement besoin d'introduire les deux définitions suivantes.

Définition 5. *Compacitude relative*
Soit Π une famille de mesures de probabilité sur $(S, \mathcal{S})$. On dit que Π est relativement compacte si toute suite d'éléments de Π contient une sous-suite faiblement convergente.

Définition 6. *Lois de dimension finie*
Soit $S = C(E, F)$ l'ensemble des fonctions continues de E dans F et $\mathcal{S}$ sa tribu borélienne. Soit $\{P_n\}$ une famille de mesures de probabilité sur $\mathcal{S}$. Les lois de dimension finie des P_n sont les mesures $P_n\pi^{-1}_{\boldsymbol{t_1},\ldots,\boldsymbol{t_k}}$, $\forall k = 1, 2, \ldots$, $\forall \boldsymbol{t}_1, \ldots, \boldsymbol{t}_k \in E$ où

$$\begin{aligned} \pi_{\boldsymbol{t_1},\ldots,\boldsymbol{t_k}} : C(E,F) &\longrightarrow F^k \\ x &\mapsto (x(\boldsymbol{t_1}), \ldots, x(\boldsymbol{t_k})). \end{aligned}$$

On dit que les lois de dimension finie des P_n convergent faiblement vers celles de P si $P_n\pi^{-1}_{\boldsymbol{t_1},\ldots,\boldsymbol{t_k}} \Rightarrow P\pi^{-1}_{\boldsymbol{t_1},\ldots,\boldsymbol{t_k}}$. Lorsque P_n et P sont les lois d'éléments aléatoires X_n et X on écrit aussi $X_n \xrightarrow{fd} X$.

Il est facile de montrer que $P_n \Rightarrow P$ si et seulement si les lois de dimension finie des P_n convergent faiblement vers celles de P et $\{P_n\}$ est relativement compacte. Une façon simple de démontrer la compacitude relative d'une famille de mesures est d'utiliser la notion de tension de cette famille.

Définition 7. *Tension d'une famille*
Une famille Π de mesures de probabilité sur un espace métrique général S est dite tendue si

$$\forall \epsilon > 0,\ \exists K \text{ compact tel que } P(K) > 1 - \epsilon,\ \forall P \in \Pi.$$

Une famille d'éléments aléatoires $\{X_n\}$ est dite tendue si la famille des lois des X_n est tendue.

Le résultat liant les deux concepts précédents peut être résumé ainsi.

Théorème 2. *Théorème de Prohorov*
Si Π est tendue, elle est relativement compacte. Si S est séparable et complet et si Π est relativement compact, elle est tendue.

Dans cette optique, un théorème très utilisé, par exemple pour démontrer la convergence en loi de la statistique « classique » de Cramér-von Mises est le suivant.

Théorème 3. *(Théorème 8.1, Billingsley (1968))*
Soient P_n, P des mesures de probabilité sur $(C[0,1], \mathcal{C})$. Si les lois de dimension finie des P_n convergent faiblement vers celles de P, et si $\{P_n\}$ est tendue, alors $P_n \Rightarrow P$.

Dans le cas qui nous préoccupe ici, la fonctionnelle de Cramér-von Mises, basée sur la fonction caractéristique empirique, n'est même pas définie sur $C(\mathbb{R}^{pq}, \mathbb{C})$ et on ne pourra pas utiliser le Théorème 1. Pour résoudre ce problème, on a généralisé le Théorème 3.3 de Kellermeier (1980). Par ailleurs, dans le contexte particulier d'un processus basé sur la fonction caractéristique empirique, qui prend ses valeurs dans l'ensemble $\mathbb{C}$ des nombres complexes et dont l'ensemble des indices est $\mathbb{R}^n$, le problème est bien plus compliqué que dans le cas « classique ». Pour cela, il est utile d'introduire les éléments suivants.

Notons $C(\mathbb{R}^{pq}, \mathbb{C})$ l'ensemble des fonctions continues de $\mathbb{R}^{pq}$ dans $\mathbb{C}$. Schématiquement, la correspondance du processus $R_{n,A}$ de (4) peut se traduire ainsi.

$$\begin{aligned} R_{n,A} : (\Omega, \mathcal{B}, \mathsf{P}) &\longrightarrow C(\mathbb{R}^{pq}, \mathbb{C}) \\ \omega &\mapsto R_{n,A}(\cdot, \omega) : \quad \mathbb{R}^{pq} \longrightarrow \mathbb{C} \\ &\qquad\qquad \boldsymbol{t} \mapsto R_{n,A}(\boldsymbol{t}, \omega) \overset{N}{=} R_{n,A}(\boldsymbol{t}). \end{aligned}$$

On définit une métrique ρ sur l'espace de Fréchet séparable $C(\mathbb{R}^{pq}, \mathbb{C})$ par

$$\forall x, y \in C(\mathbb{R}^{pq}, \mathbb{C}),\ \rho(x, y) = \sum_{j=1}^{\infty} 2^{-j} \frac{\rho_j(x, y)}{1 + \rho_j(x, y)}$$

avec $\rho_j(x, y) = \sup_{\|\boldsymbol{t}\| \leq j} |x(\boldsymbol{t}) - y(\boldsymbol{t})|$, où $|\cdot|$ désigne le module sur $\mathbb{C}$. Notons que ρ_j est bien définie car une fonction continue sur un compact est bornée.

Remarque 2. *Un espace de Fréchet est un espace métrique linéaire complet.*

On note $\mathcal{S}$ la tribu engendrée par les ouverts de $C \equiv C(\mathbb{R}^{pq}, \mathbb{C})$ pour la métrique ρ, $\mathbb{R}^{pq}_j$ la boule fermée de centre $\mathbf{0}$ et de rayon j dans $\mathbb{R}^{pq}$, $C_j \equiv C(\mathbb{R}^{pq}_j, \mathbb{C})$ et $\mathcal{S}_j$ la tribu borélienne sur C_j pour la métrique ρ_j.
Notons aussi

$$\begin{aligned} r_j : C(\mathbb{R}^{pq}, \mathbb{C}) &\longrightarrow C_j \\ x &\mapsto r_j(x) : \quad \mathbb{R}^{pq}_j \longrightarrow \mathbb{C} \\ &\qquad\qquad \boldsymbol{t} \mapsto x(\boldsymbol{t}) = r_j(x(\boldsymbol{t})) \end{aligned}$$

la restriction d'un élément x de $C(\mathbb{R}^{pq}, \mathbb{C})$ à C_j. Puisque $\mathbb{R}^{pq}$ est localement compact, séparable et Hausdorff on a, d'après la Proposition 14.6 de (Kallenberg, 1997, p. 260),

$$R_{n,A} \xrightarrow{L} R_A \text{ si et seulement si } r_j(R_{n,A}) \xrightarrow{L} r_j(R_A),\ \forall j \geq 1.$$

On peut donc restreindre l'étude du processus $R_{n,A}$ aux sous-espaces compacts de C. Maintenant, d'après le Lemme 14.2 (Kallenberg, 1997, p. 256), on a
$r_j(R_{n,A}) \xrightarrow{L} r_j(R_A)$ si et seulement si $r_j(R_{n,A}) \xrightarrow{fd} r_j(R_A)$ et $\{r_j(R_{n,A})\}_n$ est une famille relativement compacte. De plus, d'après le Théorème 14.3 (Kallenberg, 1997, p. 257) et puisque C_j est séparable et complet, on a l'équivalence entre la compacitude relative et la tension de $\{r_j(R_{n,A})\}_n$. Pour prouver la convergence faible de $\{R_{n,A}\}_n$ vers l'élément aléatoire R_A il suffit donc de montrer que les lois de dimension finie des $r_j(R_{n,A})$ convergent faiblement vers celles des $r_j(R_A)$ et que $\forall j \geq 1,\ \{r_j(R_{n,A})\}_n$ est tendue.

Tous ces résultats sont largement exploités et détaillés dans le Chapitre 3 pour construire un test semi-paramétrique d'indépendance, pour des marginales multinormales, qui peut être utile par exemple dans l'étude des données familiales. Ils sont ensuite généralisés au cas sériel.

Pour conclure, il apparait donc que notre problématique repose sur des questions ancrées dans les prémisses de la statistique tandis que notre recherche s'appuie sur des

outils et techniques récentes et innovatrices empruntées à l'analyse multivariée (voir Bilodeau et Brenner (1999)), à la théorie des processus stochastiques (voir Billingsley (1968)) ainsi qu'aux méthodes asymptotiques (voir Ferguson (1996)).

BIBLIOGRAPHIE

[1] Billingsley, P., 1968. Convergence of probability measures. John Wiley & Sons Inc., New York. 19, 20, 22

[2] Bilodeau, M., Brenner, D., 1999. Theory of multivariate statistics. Springer Texts in Statistics. Springer-Verlag, New York.

[3] Box, G. E. P., Jenkins, G. M., 1976. Time series analysis : forecasting and control, revised Edition. Holden-Day, San Francisco, Calif., holden-Day Series in Time Series Analysis.

[4] Ferguson, T. S., 1996. A course in large sample theory. Chapman & Hall, London. 22

[5] Fisher, R. A., 1925. Statistical methods for research workers. VIII + 239 p. with 6 tables. Edinburgh and London, Oliver and Boyd. 14

[6] Funkhauser, H. G., 1936. A Note on a Tenth Century Graph. Osiris, Vol.1. 12, 15

[7] Ghoudi, K., Kulperger, R. J., Rémillard, B., 2001. A nonparametric test of serial independence for time series and residuals. J. Multivariate Anal. 79, 191–218. 18

[8] Kallenberg, O., 1997. Foundations of modern probability. Springer-Verlag, New York. 21

[9] Kellermeier, J., 1980. The empirical characteristic function and large sample hypothesis testing. J. Multivariate Anal. 10 (1), 78–87. 20

[10] Kopecky, K. J., Pierce, D. A., 1979. Efficiency of smooth goodness-of-fit tests. J. Amer. Statist. Assoc. 74, 393–397.

[11] Marsden, J., 1974. Elementary classical analysis. New York : W.H. Freeman and Company. 17

[12] Neyman, J., 1937. Smooth test for goodness of fit. Skand. Aktuar. 20, 149–199. 15, 16, 17, 18

[13] Rao, C., 1947. Large sample tests of statistical hypotheses concerning several parameters with applications to problems of estimation. In : Proceedings of the Cambridge Philosophical Society. Vol. 44. pp. 50–7. 17

[14] Rayner, J., Best, D., 1988. Smooth tests of goodness of fit for regular distributions. Comm. Statist. Theory Methods 17 (10), 3235–67.

[15] Royden, H., 1968. Real analysis. New York : MacMillan. 18

CHAPITRE 2

Goodness-of-fit tests of normality for the innovations in ARMA models

Cet article a été accepté pour publication dans la revue *Journal of Time Series Analysis*.

Comme la coutume dans cette discipline le veut, l'ordre alphabétique des auteurs a été respecté.

Voici la liste des contributions principales de Pierre Lafaye de Micheaux à cet article :

- Recherche bibliographique ayant permis notamment la rédaction de la section 1.
- Démonstration des résultats de la section 2.
- Conception, écriture et validation des programmes Fortran77 pour obtenir la table 3.1.
- Conception, écriture et validation des programmes Fortran77 pour obtenir les résultats de la section 4.
- Conception, écriture et validation de l'exemple.
- Participation à la rédaction.

Goodness-of-fit tests of normality for the innovations in ARMA models

(abbreviated title: Testing the residuals in ARMA)

Gilles R. Ducharme and Pierre Lafaye de Micheaux

Laboratoire de probabilités et statistique, cc051
Université Montpellier II
Place Eugène Bataillon
34095, Montpellier, Cedex 5
France

Abstract
In this paper, we propose a goodness-of-fit test of normality for the innovations of an ARMA(p, q) model with known mean or trend. This test is based on the data driven smooth test approach and is simple to perform. An extensive simulation study is conducted to study the behavior of the test for moderate sample sizes. It is found that our approach is generally more powerful than existing tests while holding its level throughout most of the parameter space and thus, can be recommended. This agrees with theoretical results showing the superiority of the data driven smooth test approach in related contexts.

Key words: ARMA process, Gaussian white noise, Goodness-of-fit test, Normality of residuals, Smooth test.

1. INTRODUCTION

Let (Y_t, $t \in \mathbb{Z}$) be a stationary process. In this paper, we consider the case where $E(Y_t)$ is known or has been estimated using information outside of the data set. Thus, without loss of generality, we set $E(Y_t) = 0$. Consider the framework where (Y_t, $t \in \mathbb{Z}$) obeys the causal and invertible finite order ARMA(p, q) model

$$Y_t - \boldsymbol{\varphi}^{\mathsf{T}} \mathbf{Y}_{t-1}^{(p)} = \boldsymbol{\theta}^{\mathsf{T}} \boldsymbol{\epsilon}_{t-1}^{(q)} + \epsilon_t \tag{1}$$

where (ϵ_t, $t \in \mathbb{Z}$) is an innovation process of random variables with mean 0 and auto-covariance $E(\epsilon_t \epsilon_{t+h}) = \sigma^2 < \infty$ (unknown) if $h = 0$ and 0 otherwise and where

$$\boldsymbol{\varphi} = \begin{bmatrix} \varphi_1 \\ \vdots \\ \varphi_p \end{bmatrix}, \ \boldsymbol{\theta} = \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_q \end{bmatrix}, \ \mathbf{Y}_{t-1}^{(p)} = \begin{bmatrix} Y_{t-1} \\ \vdots \\ Y_{t-p} \end{bmatrix}, \ \boldsymbol{\epsilon}_{t-1}^{(q)} = \begin{bmatrix} \epsilon_{t-1} \\ \vdots \\ \epsilon_{t-q} \end{bmatrix}.$$

A sample $\{Y_1, ..., Y_T\}$ is observed and model (1) is fitted by standard methods, for example the unconditional Gaussian maximum likelihood approach (see Brockwell and Davis (1991), p. 256-257), yielding the estimator $\hat{\boldsymbol{\beta}} = (\hat{\boldsymbol{\varphi}}^\mathsf{T}, \hat{\boldsymbol{\theta}}^\mathsf{T}, \hat{\sigma})^\mathsf{T}$ of $\boldsymbol{\beta} = (\boldsymbol{\varphi}^\mathsf{T}, \boldsymbol{\theta}^\mathsf{T}, \sigma)^\mathsf{T}$.

If it can be safely assumed that the distribution of the (ϵ_t, $t \in \mathbb{Z}$) generating the Y_t's is of a given form, in particular independent identically distributed (*i.i.d.*) normal (Gaussian) random variables, then better inference can be obtained from the fitted model. For example, such an assumption is helpful to get accurate confidence or tolerance bounds for a predicted Y_{T+h}. Moreover, under this Gaussian assumption, $\hat{\boldsymbol{\beta}}$ is asymptotically efficient. It is thus important to have a tool to check the null hypothesis

$$H_0 : \text{ the } \epsilon_t\text{'s are } i.i.d. \sim N(0, \sigma^2). \tag{2}$$

As pointed out by Pierce and Gray (1985) and Brockett et al. (1988), other reasons may motivate a test of (2). One such reason is to check the fit of the structural part of (1). Indeed, the process of fitting a model to data often reduces to finding the model whose residuals behave most like a sample of $i.i.d.$ Gaussian variables. In this context, rejection of (2) may indicate lack-of-fit of the entertained ARMA model. We will not elaborate further here on this possibility and assume, in the sequel, that model (1) is not underspecified. Note however that there exist specific tests for detecting lack-of-fit (for a recent review, see Koul and Stute (1999)).

For the problem of testing (2), the few tests available fall roughly into two groups. Tests of the first group use the fact that for the ARMA (p, q) models, normality of the Y_t's induces normality of the ϵ_t's and vice versa. Thus a test of the hypothesis that a process (Y_t, $t \in \mathbb{Z}$) is Gaussian (Lomnicki (1961); Hinich (1982); Epps (1987)) can serve for problem (2). This presents the advantage of not requiring the values of p and q. But Gasser (1975) and Granger (1976) have shown, and Lutkepohl and Schneider (1989) have confirmed by simulation, that this approach may lose much power. This is because the central limit theorem forces the Y_t's to be close to normality even when (2) is false. Moreover, the adaptation of standard normality tests to dependent data is not an easy task. A small simulation study by Heuts and Rens (1986) has shown that, because of the serial correlation between the Y_t's, the finite null behavior of standard normality tests based on the empirical distribution function (EDF) of the Y_t's is different from what is obtained under $i.i.d.$ data. The same problem appears for tests based on the third or fourth moment of Y_t (see Lomnicki (1961); Lutkepohl and Schneider (1989)) and for Pearson's chi-square test (Moore (1982)).

It thus appears better, when there are reasons to believe that a given ARMA(p, q) model holds, to "inverse filter" the data and compute the residuals $\hat{\epsilon}_t$ of the fitted model. These can then be subjected to some test of normality. Tests of the second group are based on this idea and some examples are listed in Hipel and McLeod (1994). However, these and other authors use such tests in conjunction with critical values for $i.i.d.$ data. Since the residuals of an ARMA model are dependent, the null distribution of standard test statistics may be affected and critical values for $i.i.d.$ data may no longer

be valid. It turns out that for AR models, there is theoretical evidence that this dependence affects only slightly the critical values, at least when T is large. For an AR(p) model with unknown $E(Y_t)$, Pierce and Gray (1985) has shown that the asymptotic null distribution of any test statistic based on the EDF of the residuals coincides with that of the same statistic for $i.i.d.$ data with mean and variance unknown. Thus one can insert the residuals from an AR(p) model into any of the standard EDF-based tests (Kolmogorov-Smirnov, Anderson-Darling) and if T is large, use the critical values given, for example, in Chapter 4 of D'Agostino and Stephens (1986), to obtain an asymptotically valid test strategy. In the same vein, Lee and Na (2002) have recently adapted the Bickel-Rosenblatt test to this AR setting. Beiser (1985) has found that for the AR(1) model, tests based on the skewness or kurtosis coefficient of the residuals (D'Agostino and Stephens (1986), p. 408) in conjunction with the critical points derived for $i.i.d.$ data produce valid levels if T is large and the AR-parameter is not too close to its boundary. This has been confirmed by Lutkepohl and Schneider (1989). See also Anděl (1997).

For the general ARMA model, much less is known. Ojeda et al. (1997) show that tests based on quadratic forms in differences between sample moments and expected values of certain non-linear functions of the sample have the same asymptotic distribution under the ARMA model as under $i.i.d.$ data. This suggests that a generalization of Pierce and Gray (1985) theorem to ARMA models could hold although, to our knowledge, no proof of this has been published. In accordance with this conjecture, the practice recommended in many textbooks (see for example, Brockwell and Davis (1991), p. 314; Hipel and McLeod (1994), p. 241) is to use standard normality tests in conjunction with critical values for $i.i.d.$ data.

In this paper, we develop some tests designed specifically for problem (2) in the ARMA(p, q) context. Our approach is based on the smooth test paradigm introduced by Neyman (1937) and improved by the data driven technology introduced by Ledwina (1994) to select the best order for the test. This approach has been shown in the $i.i.d.$ case to offer many advantages, both theoretically and empirically, over other tests. In particular, the test statistic we recommend for problem (2) is easy to compute with an asymptotic χ^2 distribution that can be corrected in finite samples to yield a close to nominal level. Moreover, as a byproduct of the procedure, diagnostic information is available that helps in understanding which aspects of the null hypothesis are not supported by the data.

Note that we concentrate here on the development of valid tests along this paradigm and do not dwell into their theoretical properties (i.e. local power and asymptotic efficiency). We also stress that the tests proposed here are valid solely for the case where $E(Y_t)$ is assumed known. The case where an unknown trend is present in (1) requires a special treatment and is the object of current research.

The paper is organized as follows. In Section 2, we develop the smooth goodness-of-fit test in the ARMA(p, q) context of (1). In Section 3, we describe the data-driven technology that allows to "fine tune" the test by choosing a good value for its order. In Section 4, a Monte-Carlo study is conducted for some values of (p, q) to study the

behavior of the proposed tests under the null hypothesis and compare their power to some competitors. It emerges that, under the null hypothesis, one of our data driven smooth tests holds its level over most of the parameter space and, under the alternatives studied, is in general more powerful than other methods. It can thus be recommended as a good tool for problem (2). An example concludes the paper.

2. SMOOTH TEST OF NORMALITY IN THE ARMA CONTEXT

Let $\Phi(.)$ be the cumulative distribution function of the $N(0,1)$ distribution with density $\phi(.)$ and let $U_t = 2\Phi(\epsilon_t/\sigma) - 1$ with density $g(.)$. Under H_0 of (2), the U_t's are $i.i.d.$ U[-1, 1] so that (2) reduces to testing $g(u) = 1/2$ on [-1, 1]. The ϵ_t's are unobserved so the test must be based on residuals. Since the process $(Y_t, t \in \mathbb{Z})$ is invertible, we have

$$\epsilon_t = -\sum_{r=0}^{\infty} \delta_r Y_{t-r} \tag{1}$$

where the δ_r's are functions of $\boldsymbol{\theta}$ and $\boldsymbol{\varphi}$ (see (A.2), (A.3) of Appendix A). Let $\hat{\delta}_r$ be the Gaussian maximum likelihood estimator (*m.l.e.*) of δ_r under (2), obtained by plugging in the *m.l.e.* $\hat{\boldsymbol{\theta}}$ and $\hat{\boldsymbol{\varphi}}$ under H_0. We define the residuals of the fitted ARMA model by

$$\hat{\epsilon}_t = -\sum_{r=0}^{\infty} \hat{\delta}_r Y_{t-r}. \tag{2}$$

In practice, some scheme must be used to compute these residuals, for example by taking $Y_t = 0$ if $t < 1$. Note that other residuals can be defined for ARMA models (see Brockwell and Davis (1991), Section 9.4) but the definition above is convenient for the following derivation. Consider $\hat{U}_t = 2\Phi(\hat{\epsilon}_t/\hat{\sigma}) - 1$, $t = 1, ..., T$. Let $\{L_k(.), k \geq 0\}$ be the normalized (over [-1, 1]) Legendre polynomials (Sansone (1959)) with $L_0(.) \equiv 1$ satisfying

$$\frac{1}{2}\int_{-1}^{1} L_k(x)L_j(x)dx = 1 \text{ if } k = j \text{ and } 0 \text{ otherwise.} \tag{3}$$

For some integer $K \geq 1$, consider the density defined on [-1, 1] by

$$g_K(u;\boldsymbol{\omega}) = c(\boldsymbol{\omega}) \exp\left\{\sum_{k=1}^{K} \omega_k L_k(u)\right\}, \tag{4}$$

where $c(\boldsymbol{\omega})$ is a normalizing constant such that $c(\mathbf{0}) = 1/2$. In the smooth test paradigm, (4) is the K-th order alternative with $g_K(.;0)$ being the U[-1, 1] density. Thus, if $g(u)$ can be approximated by (4), (2) reduces to testing H_0: $\boldsymbol{\omega} = \mathbf{0}$. For this, we use the following route. Let $\boldsymbol{L}_t = (L_1(U_t), ..., L_K(U_t))^{\mathsf{T}}$, $\hat{\boldsymbol{L}}_t = (L_1(\hat{U}_t), ..., L_K(\hat{U}_t))^{\mathsf{T}}$ and

$$\overline{\hat{\boldsymbol{L}}} = T^{-1}\sum_{t=1}^{T} \hat{\boldsymbol{L}}_t. \tag{5}$$

Under H_0, $\boldsymbol{L}_t$ has mean $\mathbf{0}$ and covariance matrix $\boldsymbol{I}_K$, the K-th order identity matrix. Under (4), these moments will differ and (5) can be used to capture departures from the U[-1, 1] in the "direction" of $g_K(.;\boldsymbol{\omega})$. This suggests as a test statistic a quadratic form in $\overline{\hat{\boldsymbol{L}}}$. To complete the test, we need the null asymptotic distribution of (5). This is given in the following theorem.

Theorem 1. *Consider the causal and invertible ARMA(p, q) process of (1) where we assume* $1-\varphi_1 z-...-\varphi_p z^p$ *and* $1+\theta_1 z+...+\theta_q z^q$ *have no common zeroes. Under* H_0*, we have*

$$\sqrt{T}\overline{\hat{\boldsymbol{L}}} \xrightarrow{L} N_K\left(\mathbf{0}, \boldsymbol{I}_K - \frac{1}{2}\boldsymbol{b}_K\boldsymbol{b}_K^{\mathsf{T}}\right) \tag{6}$$

where $\boldsymbol{b}_K = (b_1, ..., b_K)^{\mathsf{T}}$*, with* $b_k = \int_{\mathbb{R}} L_k(2\Phi(x)-1)x^2\phi(x)dx$*. Hence, the smooth test statistic*

$$\mathcal{R}_K = T\overline{\hat{\boldsymbol{L}}}^{\mathsf{T}}\left(\boldsymbol{I}_K - \frac{1}{2}\boldsymbol{b}_K\boldsymbol{b}_K^{\mathsf{T}}\right)^{-1}\overline{\hat{\boldsymbol{L}}} \xrightarrow{L} \chi^2_K.$$

PROOF. We present an outline of the argument. More details are given in the appendices and in Ducharme and Lafaye de Micheaux (2002). Let

$$\mathcal{I}_{\boldsymbol{\beta}} = \text{Var}\left[\frac{\partial}{\partial\boldsymbol{\beta}}\text{Log}\left(\frac{1}{\sigma}\phi\left(\frac{\epsilon_t}{\sigma}\right)\right)\right]$$

be Fisher's information matrix for $\boldsymbol{\beta}$. From standard results (see Gouriéroux and Monfort (1995), p.325), we have,

$$\sqrt{T}\left(\hat{\boldsymbol{\beta}}-\boldsymbol{\beta}\right) = \frac{1}{\sqrt{T}}\sum_{t=1}^{T}\mathcal{I}_{\boldsymbol{\beta}}^{-1}\frac{\partial}{\partial\boldsymbol{\beta}}\left[\text{Log}\left(\frac{1}{\sigma}\phi\left(\frac{\epsilon_t}{\sigma}\right)\right)\right] + o_P(1).$$

Since $\left(\hat{\boldsymbol{\beta}}-\boldsymbol{\beta}\right) = O_P(T^{-1/2})$, a Taylor expansion yields

$$\sqrt{T}\overline{\hat{\boldsymbol{L}}} = \frac{1}{\sqrt{T}}\sum_{t=1}^{T}\boldsymbol{L}_t + \left[\frac{1}{T}\sum_{t=1}^{T}\frac{\partial}{\partial\boldsymbol{\beta}}\boldsymbol{L}_t\right]\sqrt{T}\left(\hat{\boldsymbol{\beta}}-\boldsymbol{\beta}\right) + o_P(1). \tag{7}$$

The first term on the right hand side of (7) converges to a $N_K(\mathbf{0}, \boldsymbol{I}_K)$. Moreover, it is shown in Appendix A that

$$\left[\frac{1}{T}\sum_{t=1}^{T}\frac{\partial}{\partial\boldsymbol{\beta}}\boldsymbol{L}_t\right] \xrightarrow{P} \left[\underset{K\times(p+q)}{\mathbf{0}}, -\frac{1}{\sigma}\boldsymbol{b}_K\right] = -\mathcal{J}_K. \tag{8}$$

Hence,

$$\begin{aligned}\sqrt{T}\overline{\hat{\boldsymbol{L}}} &= \frac{1}{\sqrt{T}}\sum_{t=1}^{T}\boldsymbol{L}_t - \frac{1}{\sqrt{T}}\mathcal{J}_K\mathcal{I}_{\boldsymbol{\beta}}^{-1}\sum_{t=1}^{T}\frac{\partial}{\partial\boldsymbol{\beta}}\left[\text{Log}\left(\frac{1}{\sigma}\phi\left(\frac{\epsilon_t}{\sigma}\right)\right)\right] + o_P(1)\\ &= \frac{1}{\sqrt{T}}\sum_{t=1}^{T}\boldsymbol{B}\boldsymbol{V}_t + o_P(1),\end{aligned}$$

where $\boldsymbol{B} = (\boldsymbol{I}_K, -\mathcal{J}_K\mathcal{I}_{\boldsymbol{\beta}}^{-1})$ and

$$\boldsymbol{V}_t = \left(\boldsymbol{L}_t^{\mathsf{T}}, \frac{\partial}{\partial\boldsymbol{\beta}^{\mathsf{T}}}\left[\text{Log}\left(\frac{1}{\sigma}\phi\left(\frac{\epsilon_t}{\sigma}\right)\right)\right]\right)^{\mathsf{T}}.$$

From Appendix B, it follows that, $E(\boldsymbol{V}_t) = \mathbf{0}$ and $\text{Var}(\boldsymbol{B}\boldsymbol{V}_t) = \boldsymbol{I}_K - \boldsymbol{b}_K\boldsymbol{b}_K^{\mathsf{T}}/2$. The central limit theorem yields (6). □

It is possible to write $\mathcal{R}_K$ in a form that makes it easy to use. A Cholesky decomposition of $\left(\boldsymbol{I}_K - \boldsymbol{b}_K\boldsymbol{b}_K^\intercal/2\right)$ yields $\left(\boldsymbol{I}_K - \boldsymbol{b}_K\boldsymbol{b}_K^\intercal/2\right)^{-1} = \boldsymbol{P}\boldsymbol{P}^\intercal$ with $\boldsymbol{P} = (p_{ij})$, an upper triangular matrix. Some algebra gives $p_{ij} = 0$ if $i > j$, while

$$p_{ii} = \sqrt{\frac{2 - \sum_{k=1}^{i-1} b_k^2}{2 - \sum_{k=1}^{i} b_k^2}} \text{ and } p_{ij} = \frac{b_i b_j}{\sqrt{\left(2 - \sum_{k=1}^{j-1} b_k^2\right)\left(2 - \sum_{k=1}^{j} b_k^2\right)}} \text{ if } j > i.$$

Thus

$$\mathcal{R}_K = \sum_{k=1}^{K} \left(\frac{1}{\sqrt{T}} \sum_{t=1}^{T} L_k^*(\hat{U}_t) \right)^2,$$

where

$$L_k^*(\hat{U}_t) = \sum_{l=1}^{k} p_{lk} L_l(\hat{U}_t). \tag{9}$$

Numerical integration gives $(b_2, b_4, ..., b_{10}) = (1.23281, 0.521125, 0.304514, 0.205589, 0.150771)$ with $b_k = 0$ if k is odd. This yields the first ten "modified" Legendre polynomials

$$\begin{aligned}
L_1^*(u) &= 1.73u, \\
L_2^*(u) &= 6.85u^2 - 2.28, \\
L_3^*(u) &= 6.61u^3 - 3.97u, \\
L_4^*(u) &= 19.91u^4 - 10.26u^2 - 0.56, \\
L_5^*(u) &= 26.12u^5 - 29.02u^3 + 6.22u, \\
L_6^*(u) &= 69.84u^6 - 81.84u^4 + 28.36u^2 - 3.06, \\
L_7^*(u) &= 103.84u^7 - 167.75u^5 + 76.25u^3 - 8.47u, \\
L_8^*(u) &= 260.07u^8 - 450.18u^6 + 247.18u^4 - 38.73u^2 - 1.11, \\
L_9^*(u) &= 413.92u^9 - 876.55u^7 + 613.58u^5 - 157.33u^3 + 10.73u, \\
L_{10}^*(u) &= 994.51u^{10} - 2250.43u^8 + 1782.83u^6 - 569.92u^4 + 67.54u^2 - 3.58.
\end{aligned}$$

Remark 2.1. *Theorem 1 shows that we can slightly extend the result of Pierce and Gray (1985) and state that neither the estimation of $\boldsymbol{\varphi}$ and $\boldsymbol{\theta}$ nor the dependence of the Y_t's has any asymptotic impact on a smooth test of (2) in the ARMA context. In pre-asymptotic situations, these elements and the complexity of the model will affect the null distribution of $\mathcal{R}_K$. This will be further explored in simulations of Section 4.*

Remark 2.2. *Each term $T^{-1}\left(\sum_{t=1}^{T} L_k^*(\hat{U}_t)\right)^2$ is a component of the test statistic and has an asymptotic χ_1^2 distribution under H_0. When the null hypothesis is rejected, some of these components will be large. The simple structure of the first few polynomials in (9) helps in understanding what aspects of the normal are not supported by the data. For example, the first component detects departure from symmetry under H_0 in the "direction" of asymmetry. This diagnostic analysis must be undertaken with some care however; see Henze (1997) for details.*

Remark 2.3. *The above methodology can in principle be applied to other distributions than the normal. For location-scale densities, one needs to replace the normal distribution in the definition of U_t and follow the derivation using the new null density.*

The structure of $\mathcal{R}_K$ will be similar to what is obtained above but the modified Legendre polynomials will change. For distributions with a shape parameter, the statistic is more complex since the coefficients of these polynomials will in general depend on this unknown shape parameter that must be estimated.

3. CHOOSING THE ORDER K OF THE ALTERNATIVE

Before applying the test strategy of Section 2, one must choose the value of K. Ideally, this choice should be made so that members of the embedding family $g_K(.;\boldsymbol{\omega})$ of (4) provide a good approximation to any plausible density $g(.)$ of U_t under the alternative. If K is too small, this approximation may be crude and the test loses power. If K is too large, power dilution can occur since $g_K(.;\boldsymbol{\omega})$ encompasses unnecessary "directions".

In practice, the user has only, at best, a qualitative idea of the plausible alternatives and no specific value of K emerges naturally. In the $i.i.d.$ case, some authors (Rayner and Best (1989)) argue that, as a rule of thumb, one can use a trade-off value of K between 2 and 4.

Recently, Ledwina (1994) and Kallenberg and Ledwina (1997a,b) have proposed and explored for $i.i.d.$ data a method to choose adaptively a value for K. At the first step, Schwarz (1978)'s criterion is used to choose the value $\hat{K}$ that seems best in view of the data at hand. The smooth test strategy is then applied using the statistic $\mathcal{R}_{\hat{K}}$. Extensive simulations have shown that, even for small sample sizes, this so-called "data driven smooth test" can yield power close to what could be obtained if one knew the true form of the alternative and had chosen the best value of K accordingly.

So far, this approach has been investigated for $i.i.d.$ data only but it can be extended to the ARMA context. Choose two integers $1 \leq d \leq D$ and consider the set of statistics $(\mathcal{R}_d, ..., \mathcal{R}_D)$. We seek a rule that will select a good $\mathcal{R}_K$ in this set. Write

$$\hat{K} = \min \left[\underset{d \leq s \leq D}{\text{Argmax}} \{\mathcal{R}_s - s\text{Log}(T)\} \right] \tag{1}$$

and denote $\mathcal{R}_{\hat{K}}(d)$, the test statistic $\mathcal{R}_{\hat{K}}$ selected by (1) in $(\mathcal{R}_d, ..., \mathcal{R}_D)$.

Theorem 2. *Under H_0, $\hat{K} \longrightarrow d$ in probability and thus, $\mathcal{R}_{\hat{K}}(d)$ is asymptotically χ^2_d.*

PROOF. Set $e_k = (k-d)\text{Log}T$. For $k \geq d$, $P(\hat{K} = k) \leq P(\mathcal{R}_k > e_k)$. Now, since each $\mathcal{R}_k$ is asymptotically χ^2_k under H_0, as T increases,

$$P(\mathcal{R}_k > e_k) \longrightarrow 0,$$

when k > d. It follows that $P(\hat{K} = d) = 1 - P(\hat{K} \geq d+1) \longrightarrow 1$. □

For finite sample sizes, the asymptotic null distribution of Theorem 2 may not provide a good approximation to that of $\mathcal{R}_{\hat{K}}(d)$ since there is a positive probability that $\hat{K} \geq d+1$. A simple correction has been developed by Janic-Wroblewska and Ledwina (2000) when $d = 1$ ($i.i.d.$ data). Because of the asymptotic independence between the components of $\mathcal{R}_k$, this correction can easily be extended to $d > 1$ and to the present

ARMA context. A direct application of the argument in their Section 4 leads to the following approximation, which can be solved for x by numerical integration

$$P(\mathcal{R}_{\hat{K}}(d) \leq x) \approx P(\chi^2_d \leq x)P(\chi^2_1 \leq \text{Log}(T)) + \int_{\text{Log}(T)}^{x} P(\chi^2_d < x - z)\frac{1}{\sqrt{2\pi z}}e^{-z/2}dz. \tag{2}$$

Some quantiles corrected through (2) are listed in Table 3.1.

TABLE 3.1. Some quantiles obtained from approximation (2)

	T	a = 0.10	a = 0.05	a = 0.01
d = 1	50	3.692	5.410	8.805
	100	3.275	5.201	8.703
	200	3.057	4.751	8.590
d = 2	50	5.466	7.137	10.807
	100	5.262	6.972	10.684
	200	5.043	6.796	10.558

One may have the feeling that this data driven approach replaces the problem of selecting K with that of selecting d and D. To answer this, Kallenberg and Ledwina (1997a,b) have studied a version of the above procedure where D is allowed to increase with T. In the $i.i.d.$ case, they obtain rates connecting these quantities. These rates are theoretically interesting but do not help in practice in selecting a value for D. To get more insight, they have conducted extensive simulations. It turns out that the power levels off rapidly as D increases and there is little to be gained by choosing D much greater than 10. As for the choice of d, again Kallenberg and Ledwina (1997a) briefly discuss this problem where it emerges that in their context $d = 1$ or 2 appears reasonable. In the simulation study of the next section we use both these values of d and take $D = 10$.

In closing this section, note that, by plotting $g_{\hat{K}}(.; \hat{\boldsymbol{\omega}})$ where $\hat{\boldsymbol{\omega}}$ is an estimate of $\boldsymbol{\omega}$, one can get an idea of the true shape of the density when the null hypothesis has been rejected. This can be helpful in finding a more appropriate distribution for the innovations.

4. SIMULATION RESULTS

To get an idea of the behavior of our test statistics as compared to some competitors, a simulation study was conducted. Samples $\{Y_t, t = 1, ..., T\}$ from various ARMA(p, q) models were generated with the innovations arising, in the first part of the simulation, from the normal distribution and, in the second, from various alternatives. For each sample, we estimated the parameters of the model and computed test statistics. From there, we obtained approximations to their level and power. All programs are written in Fortran 77. The subroutines listed below are from the Numerical Algorithms Group (NAG) MARK 16 Fortran library.

4.1. Levels

The first part of the simulation study was designed to see if the critical values obtained from the asymptotic χ^2 or from (2) can be relied upon in finite samples. We took $T = 50$, 100 and 200 and restricted attention to the models MA(2), AR(2), ARMA(1,2), ARMA(2,1) and ARMA(2, 2). To generate ARMA(p,q) samples with Gaussian innovations, we used subroutine G05EGF and G05EWF. These samples were submitted to subroutine G13DCF that returns estimates of the parameters of the model as well as residuals. The definition of these residuals, given at equation (9.4.1) in Brockett et al. (1988), differs from (2) but their numerical values are almost identical. These residuals were then submitted to the various tests. The actual level of each test was computed for nominal level $\alpha = 0.10$ and 0.05.

Regarding the parameter $\boldsymbol{\beta}$, note that our test statistics are in theory invariant to the choice of σ and we took $\sigma = 1$. Numerically, this invariance holds approximately because of the stopping rule in G13DCF. But the finite distribution of our test statistics depends on the values of $\boldsymbol{\theta}$ and $\boldsymbol{\varphi}$. To explore this, we have proceeded as follows. First, causality requires that, if $p = 1$, $\varphi_1 \in]-1, 1[$ while if $p = 2$, $\boldsymbol{\varphi}$ must be in the region $\triangle_{\boldsymbol{\varphi}} = \{(\varphi_1, \varphi_2) | \varphi_1 + \varphi_2 < 1, \varphi_2 - \varphi_1 < 1, |\varphi_2| < 1\}$ (Brockett et al. (1988), p. 110, ex.3.2). Similarly, invertibility implies that if $q = 1$, $\theta_1 \in]-1, 1[$ while if $q = 2$, $\boldsymbol{\theta}$ must be in $\triangledown_{\boldsymbol{\theta}} = \{(-\theta_1, -\theta_2) | \theta_1 + \theta_2 < 1, \theta_2 - \theta_1 < 1$ and $|\theta_2| < 1\}$. In addition, the polynomials $1 - \varphi_1 z$ when $p = 1$ and $1 - \varphi_1 z - \varphi_2 z^2$ when $p = 2$ must have no common zeroes with $1 + \theta_1 z$ when q = 1 and $1 + \theta_1 z + \theta_2 z^2$ when q = 2.

For the AR(2) model, we have taken the values of $\boldsymbol{\varphi}$ in the grid of 64 points $\{(-2.0 + 0.25j, -0.9 + 0.25k) \in \triangle_{\boldsymbol{\varphi}} | j, k \geq 0\}$. A similar grid was used for the MA(2). This makes it possible to see whether the tests maintain the proper critical level over a large section of the parameter space. For the ARMA(1, 2), the grid over $\triangledown_{\boldsymbol{\theta}}$ was reduced to $\{(-2.0 + 0.40j, -0.9 + 0.40k) \in \triangledown_{\boldsymbol{\theta}} | j, k \geq 0\}$ while $\varphi_1 = -0.9 + 0.2j, j = 0, ..., 9$. This gives a set of 250 points on the parameter space of $(\varphi_1, \boldsymbol{\theta})$. For the ARMA(2, 1) model, the same was done with $\boldsymbol{\varphi}$ and θ_1 instead. Finally, for the ARMA (2, 2) model, points $(\boldsymbol{\varphi}, \boldsymbol{\theta})$ satisfying the "no common zeroes" condition were taken in $\{(-1.95 + 0.45j, -0.85 + 0.45k) \in \triangle_{\boldsymbol{\varphi}} | j, k \geq 0\} \cup \{-(-1.95 + 0.45j, -0.95 + 0.45k) \in \triangledown_{\boldsymbol{\theta}} | j, k \geq 0\}$. This yields 294 $(\boldsymbol{\varphi}, \boldsymbol{\theta})$ parameter points. For each of these parameter points, 10000 samples of size T were generated as described above.

To summarize the results, the following approach was adopted. A 95% confidence interval for the true level when $\alpha = 0.10$ is (0.094, 0.106). Similarly, for $\alpha = 0.05$, 95% of the p-values are expected in the interval (0.046, 0.054). Thus the range of possible p-values was divided in 5 sub-intervals. For $\alpha = 10\%$, these are $I_1 = (0, 0.085)$, $I_2 = [0.085, 0.094)$, $I_3 = [0.094, 0.106)$, $I_4 = [0.106, 0.115)$ and $I_5 = [0.115, 1]$. For $\alpha = 0.05$, $I_1 = (0, 0.035)$, $I_2 = [0.035, 0.046)$, $I_3 = [0.046, 0.054)$, $I_4 = [0.054, 0.065)$ and $I_5 = [0.065, 1]$. For each model, the percentage of p-values in each interval was recorded. Table 4.1 reports the results for statistics $\mathcal{R}_3$ and $\mathcal{R}_{\hat{K}}(2)$ which, as discussed in Section 3, are representative of the two schools of thought for the choice of K. The results for the AR(2) and ARMA(2,1) models being similar to those

of the MA(2) and ARMA(1,2) respectively, are omitted for brevity (see Ducharme and Lafaye de Micheaux (2002) for more complete results).

TABLE 4.1. Distribution (in % of the number of parameter points) of the empirical p-values (based on 10000 replications) for the tests based on $\mathcal{R}_3$ and $\mathcal{R}_{\hat{K}}(2)$ among 5 sub-intervals.

$\mathcal{R}_3$			Observed level					Min
Model	T	α	I_1	I_2	I_3	I_4	I_5	p-level
	50	5%	18.8	68.8	12.5	0	0	2.76
	100	5%	1.6	50.0	48.4	0	0	3.41
MA(2)	200	5%	0	9.4	89.1	1.6	0	4.04
	50	10%	23.4	53.1	23.4	0	0	6.49
(64 points)	100	10%	6.3	20.3	73.4	0	0	7.96
	200	10%	0	7.8	90.6	1.6	0	8.92
	50	5%	47.2	46.4	6.4	0	0	2.43
	100	5%	8.0	71.6	20.4	0	0	2.98
ARMA(1,2)	200	5%	0.8	32.4	66.4	0.4	0	3.32
	50	10%	65.6	24.0	10.4	0	0	6.20
(250 points)	100	10%	21.6	35.2	42.8	0.4	0	6.80
	200	10%	4.0	19.6	75.6	0.8	0	7.42
	50	5%	41.2	57.1	1.7	0	0	2.56
	100	5%	5.1	74.1	20.8	0	0	3.09
ARMA(2,2)	200	5%	0.3	27.9	71.8	0	0	3.47
	50	10%	57.8	37.4	4.8	0	0	6.24
(294 points)	100	10%	21.1	33.7	45.2	0	0	6.88
	200	10%	3.1	18.0	78.6	0.3	0	7.86
$\mathcal{R}_{\hat{K}}(2)$			Observed level					Min
Model	T	α	I_1	I_2	I_3	I_4	I_5	p-level
	50	5%	0	9.4	46.9	43.7	0	4.12
	100	5%	0	14.1	68.8	17.2	0	4.17
MA(2)	200	5%	0	7.8	87.5	4.7	0	4.23
	50	10%	6.3	14.1	62.5	17.2	0	7.78
(64 points)	100	10%	6.3	6.3	81.3	6.3	0	8.19
	200	10%	0	6.3	89.1	4.7	0	8.83
	50	5%	0	38.8	46.8	14.4	0	3.53
	100	5%	0	34.8	59.2	6.0	0	3.74
ARMA(1,2)	200	5%	0	23.2	75.2	1.6	0	3.80
	50	10%	24.8	31.6	35.2	8.4	0	7.06
(250 points)	100	10%	13.2	27.2	57.6	2.0	0	7.33
	200	10%	4.4	18.0	76.0	1.6	0	7.61
	50	5%	0	32.0	55.4	12.6	0	3.65
	100	5%	0	31.0	62.9	6.1	0	3.75
ARMA(2,2)	200	5%	0	23.8	75.9	0.3	0	3.89
	50	10%	21.4	30.3	47.0	1.4	0	7.14
(294 points)	100	10%	11.2	24.8	62.6	1.4	0	7.51
	200	10%	2.7	16.7	80.6	0	0	7.62

The actual levels for $\mathcal{R}_3$ are concentrated on I_1, I_2 and I_3. The mode of the distribution is generally located on I_2 for $T = 50$ and is shifted to I_3 as T increases. This lead, at worst, to slightly conservative tests. To appreciate this, the last column of Table 4.1 gives the smallest p-value recorded over the parameter points. For $\mathcal{R}_{\hat{K}}(2)$, the distribution is concentrated on I_2, I_3 and I_4 with, in all cases, a mode centered on I_3. For this statistic, the minimal p-values are also closer to the nominal level (no maximal p-value was very far from the upper bound of I_4). Thus correction (2) works nicely, at least for the cases considered here.

We also investigated what areas of the parameter space give p-values in I_1. Intuitively, one expects these points to be near the boundary. However, the pattern that emerges, which is very similar for both $\mathcal{R}_3$, and $\mathcal{R}_{\hat{K}}(2)$, is more precise. For AR(2) models, these points correspond mainly to positive (φ_1, φ_2) close to the right boundary of $\triangle_{\boldsymbol{\varphi}}$ and, to a lesser degree, to those with positive φ_1 and negative φ_2 but again close to that boundary. For MA(2) models, the situation is reversed, which is not surprising since $\bigtriangledown_{\boldsymbol{\theta}} = -\triangle_{\boldsymbol{\varphi}}$. For ARMA(2, 1), the points giving small p-values correspond to positive (φ_1, φ_2) combined with values of θ_1 close to -1. Again, for ARMA(1, 2) the situation is reversed and small p-values are associated with negative values of (θ_1, θ_2) with a value of φ_1 close to 1. Finally, for the ARMA(2, 2), the points that yield p-values in I_1 are mainly those with positive (φ_1, φ_2) and negative (θ_1, θ_2).

We have also investigated the behavior under H_0 of some other tests that have been recommended in the time series literature for (2). We first considered the Anderson-Darling ($\mathcal{AD}$) test (Pierce and Gray (1985)) for case 2 (known mean) used in conjunction with the quantiles given in D'Agostino and Stephens (1986) p. 122. Our simulations show that, for large T this yields valid critical levels. We also studied a variant of the Shapiro-Wilk test known as the Weisberg and Bingham (1975) ($\mathcal{WB}$) test. To adapt this test to our context where the mean is known, the denominator of equation (9.68) of D'Agostino and Stephens (1986) was replaced by $T\hat{\sigma}^2$, where $\hat{\sigma}^2$ is the estimate of σ^2 returned by subroutine G13DCF. Up to the numerical accuracy of procedure G13DCF, this corresponds to the sum of squares of the residuals. Our simulations show that the quantiles for this test can be approximated by Monte Carlo using $i.i.d.$ data, although we found no theoretical result supporting this. Thus, we simulated 100000 samples from an ARMA(0,0) model and computed the empirical quantiles. For T = 50, 100 and 200, we got, for $\alpha = 10\%$, 0.920, 0.958 and 0.978. For 5%, we found 0.899, 0.947 and 0.973. A third approach, the Jarque and Bera (1987) eq. (5) ($\mathcal{JB}$) test was also investigated. Although developed in the linear regression context, this test has been recommended in the time series literature (see Cromwell et al. (1994); Frances (1998)). A summary of the results for these tests in the ARMA(1, 2) model is given in Table 4.2. Also appearing in this table are the levels of the test based on $\mathcal{R}_{\hat{K}}(1)$ using quantiles derived from (refequation3.2).

Overall, the best tests, according to the criterion of maintaining the proper level throughout the parameter space, are $\mathcal{R}_{\hat{K}}(2)$ followed by $\mathcal{R}_{\hat{K}}(1)$ and then $\mathcal{R}_3$, $\mathcal{AD}$ and $\mathcal{WB}$. In general, the AD test yields distributions of p-values in between those of $\mathcal{R}_3$ and $R_{\hat{K}}(1)$. More troublesome is the fact that this test, as well as the $\mathcal{WB}$ test, may vastly underestimate the intended level, as can be seen by the minimal p-values (last

column of Table 4.2) encountered on the grids. Also, there appears to be a problem with the $\mathcal{JB}$ test as the quantiles, obtained from the χ^2_2 approximation, lead to gross error. Further simulations indicate that the convergence to the χ^2_2 is very slow. The $\mathcal{JB}$ statistic is a version of the Bowman and Shenton test statistic that, for $i.i.d.$ data, has a notoriously slow convergence. The simulation results in Lutkepohl and Schneider (1989) tend to show that this is also the case for AR(1) and AR(2) models. In view of this problem, we choose to drop from further investigations the $\mathcal{JB}$ test.

TABLE 4.2. Distribution (in % of the number of parameter points) of the empirical p-values (based on 10000 replications) of various tests for the ARMA(1,2) model. $\mathcal{AD}$=Anderson-Darling, $\mathcal{WB}$=Weisberg-Bingham, $\mathcal{JB}$=Jarque-Bera and $\mathcal{R}_{\hat{K}}(1) = \mathcal{R}_{\hat{K}}$ with $d = 1$.

Test			Observed level					Min
Model	T	α	I_1	I_2	I_3	I_4	I_5	p-level
$\mathcal{AD}$	50	5%	43.8	24.4	20.0	6.8	0	0.54
	100	5%	32.0	34.8	33.2	0	0	0.92
	200	5%	11.6	38.0	49.6	0.8	0	1.50
	50	10%	41.2	16.0	22.8	18.4	1.6	3.38
	100	10%	23.2	24.4	48.0	3.6	0.8	3.93
	200	10%	9.6	13.2	70.0	6.8	0.4	4.65
$\mathcal{WB}$	50	5%	61.2	23.6	15.2	0	0	0.57
	100	5%	39.2	46.0	14.4	0.4	0	0.93
	200	5%	10.8	30.4	56.0	2.8	0	1.60
	50	10%	56.8	16.4	25.6	1.2	0	2.96
	100	10%	37.2	47.2	15.6	0	0	3.55
	200	10%	15.6	36.0	46.0	2.4	0	4.71
$\mathcal{JB}$	50	5%	71.2	28.8	0	0	0	3.13
	100	5%	0.4	99.2	0.4	0	0	3.13
	200	5%	0	85.2	14.4	0.4	0	4.18
	50	10%	100	0	0	0	0	4.96
	100	10%	100	0	0	0	0	5.88
	200	10%	98.8	1.2	0	0	0	7.23
$\mathcal{R}_{\hat{K}}(1)$	50	5%	0	58.0	27.6	14.4	0	3.52
	100	5%	10.0	48.8	39.2	2.0	0	3.32
	200	5%	8.4	33.6	56.4	1.6	0	3.39
	50	10%	43.6	20.8	26.0	9.6	0	4.69
	100	10%	26.0	24.8	48.0	1.2	0	4.81
	200	10%	8.8	16.4	70.4	4.4	0	5.79

4.2. Power

The second part of the simulation was designed to study the power of our tests and allow comparison with the competitors mentioned above. We restricted attention to $i.i.d.$ innovations. We generated samples $\{Y_t, t = 1, ..., T\}$ according to model (1) from various alternatives to the normal distribution. These alternatives were taken as the centered version of the densities listed in Table V of Kallenberg and Ledwina (1997b). They comprise a large range of departure from the normal distribution both in skewness, kurtosis and shape.

To generate ARMA(p, q) samples $\{Y_t, t = 1, ..., T\}$ according to model (1) with non-Gaussian innovations, we used the random shock method (algorithms IA 1 with $m = 50$ and SA 1 with $M = 200$) of Burn (1987). To allow a proper comparison of the various tests, we used for each model a set of parameters for which the p-values computed in the first part of the simulation were in I_3 for all tests. More precisely we took: ARMA(2, 1): $(\boldsymbol{\varphi}, \theta_1) = (-0.8, -0.1, 0.7)$, ARMA(1, 2): $(\varphi_1, \boldsymbol{\theta}) = (-0.7, 0.4, 0.5)$ and ARMA(2, 2): $(\boldsymbol{\varphi}, \boldsymbol{\theta}) = (-1.05, -0.4, 0.15, 0.85)$. Also we took $T = 50$ (more complete simulations appear in Ducharme and Lafaye de Micheaux (2002)). For each combination of model and alternative distribution, we generated 10000 samples and performed the various tests. From there, empirical powers were computed.

Table 4.3 presents these empirical powers for the tests $\mathcal{R}_3$, $\mathcal{R}_{\hat{K}}(2)$ and $\mathcal{WB}$ when $\alpha = 10\%$. Similar results were obtained for $\alpha = 5\%$. The tests $\mathcal{R}_3$ and $\mathcal{R}_{\hat{K}}(2)$ behave similarly with, overall, $\mathcal{R}_{\hat{K}}(2)$ being slightly better. Both these tests generally dominate the others. The $\mathcal{AD}$ approach, not shown here, often yields a power that is much lower than these two tests whereas $\mathcal{WB}$ generally lies somewhere in between. For $i.i.d.$ data, the $\mathcal{WB}$ test, as a variant of the Shapiro-Wilk test, is considered among the best omnibus tests of normality. In ARMA situations, this does not seem to hold at the same degree.

We have also computed the power of the test based on $\mathcal{R}_{\hat{K}}(1)$. The tabulated results are not presented here for brevity. We found that, for $T = 50$ and symmetric alternatives, the test based on $\mathcal{R}_{\hat{K}}(1)$ yields slightly better power that $\mathcal{R}_{\hat{K}}(2)$. For asymmetric alternatives, the situation is reversed. But for $T = 100$, $\mathcal{R}_{\hat{K}}(2)$ is more powerful almost everywhere. This behavior of $\mathcal{R}_{\hat{K}}(1)$ is explained by the fact that for asymmetric alternatives, $\mathcal{R}_1$ yields little, sometimes trivial, power. Moreover, power as a function of K usually levels off at $K = 3$, and not infrequently at $K = 2$. This empirical observation is behind the rule of thumb stated in Section 3. Thus to have good power, the selection rule with $d = 1$ must give $\hat{K} \geq 3$, which may be difficult. Starting at $d = 2$ gives a better chance that $\hat{K} \geq 3$ when necessary.

In view of the results of these simulations, we recommend the use of $\mathcal{R}_{\hat{K}}(2)$ for testing (2) when $E(Y_t)$ is known. The levels are stable over most of the parameter points and close to nominal for moderate samples. Moreover, the power is generally better than that of other tests that have been recommended in the time series literature. Finally, the test is very easy to apply.

TABLE 4.3. Empirical power (based on 10000 replications with $\alpha = 10\%$) of various tests when $T = 50$. The part above the line in the middle of the table corresponds to symmetric alternatives while those below are skewed. The distributions are ordered according to increasing kurtosis. The ARMA(2,1) model has parameter $(\boldsymbol{\varphi}, \theta_1) = (-0.8, -0.1, 0.7)$, the ARMA(1,2) model has parameter $(\varphi_1, \boldsymbol{\theta}) = (-0.7, 0.4, 0.5)$ while the ARMA(2,2) model has parameter $(\boldsymbol{\varphi}, \boldsymbol{\theta}) = (-1.05, -0.4, 0.15, 0.85)$.

$T = 50$	ARMA(2,1)			ARMA(1,2)			ARMA(2,2)		
Alternatives	$\mathcal{R}_3$	$\mathcal{R}_{\hat{K}}(2)$	$\mathcal{WB}$	$\mathcal{R}_3$	$\mathcal{R}_{\hat{K}}(2)$	$\mathcal{WB}$	$\mathcal{R}_3$	$\mathcal{R}_{\hat{K}}(2)$	$\mathcal{WB}$
SB(0;0.5)	83.19	84.76	28.93	73.16	74.90	22.47	63.85	65.44	17.57
TU(1.5)	66.94	67.96	18.27	57.86	59.43	15.07	49.17	50.50	13.62
TU(0.7)	44.47	45.64	12.42	38.69	39.44	11.02	32.58	33.88	10.76
Logistic(1)	20.74	22.75	19.02	18.97	20.87	17.10	18.59	20.36	17.60
TU(10)	94.64	96.64	83.60	89.57	91.62	74.31	85.14	87.26	65.78
SC(0.05;3)	33.65	37.38	35.98	32.82	35.65	34.40	30.81	34.69	32.37
SC(0.2;5)	96.36	96.77	92.84	94.21	94.72	89.12	92.28	92.96	85.62
SC(0.05;5)	62.33	65.22	63.63	61.43	63.86	61.81	58.90	62.20	59.77
SC(0.05;7)	74.05	76.12	75.32	73.00	75.22	73.89	72.28	74.04	72.50
SU(0;1)	75.96	76.49	66.57	71.73	72.44	62.20	68.60	69.79	59.99
SB(0.533;0.5)	91.09	89.76	59.41	83.62	82.09	48.17	76.29	74.04	36.87
SB(1;1)	53.75	56.94	32.60	45.94	48.41	26.18	42.03	44.17	23.84
LC(0.2;3)	55.58	57.79	29.72	49.58	52.11	26.19	43.88	46.16	22.62
Weibull(2)	28.10	30.72	21.25	25.63	28.66	18.52	24.32	26.85	18.68
LC(0.1;3)	44.10	43.85	35.21	40.62	40.54	31.38	37.00	37.25	27.97
χ^2 (df.=10)	41.41	45.84	34.72	37.80	41.98	31.09	35.04	38.91	29.66
LC(0.05;3)	29.50	31.25	28.28	28.05	29.86	26.18	25.91	28.32	24.17
LC(0.1;5)	96.10	96.00	95.07	93.40	93.03	90.44	91.04	89.90	86.06
SU(-1;2)	37.88	38.92	33.93	34.38	36.29	31.12	33.54	65.40	29.87
χ^2 (df.=4)	76.13	80.54	69.78	71.19	75.76	63.30	65.96	70.30	57.70
LC(0.05;5)	81.48	83.98	84.41	78.07	80.38	80.80	75.48	77.71	77.58
LC(0.05;7)	94.26	94.74	96.28	94.05	94.73	96.39	93.44	94.24	95.88
SU(1;1)	96.26	96.15	93.98	94.18	94.17	91.39	93.14	93.17	90.14
LN(0;1)	99.52	99.68	99.24	98.74	98.85	98.02	97.89	98.16	96.83

5. AN EXAMPLE

In the course of a study to forecast the amount of daily gas required, Shea (1987) has studied a bivariate time series of $T = 366$ points. The first component of this time series pertains to differences in daily temperature between successive days ($\bigtriangledown\tau_t$) and he found, after an iteration process of fitting and diagnostic checking, that the following MA(4) model could be entertained:

$$\bigtriangledown\tau_t = \epsilon_t + 0.07\epsilon_{t-1} - 0.30\epsilon_{t-2} - 0.15\epsilon_{t-3} - 0.20\epsilon_{t-4}.$$

The residual variance is 2.475. All these parameters are obtained by maximizing the Gaussian likelihood so that problem (2) is of some importance. Shea does not discuss the normality of the innovations in assessing the fit of this model but rather goes on to find a good model for the bivariate series based on an analysis of the residuals' cross correlation matrix.

An application of our tests yields $\mathcal{R}_3 = 22.85$, with a p-value of 0.00004 while $\mathcal{R}_{\hat{K}}(2) = 22.77$ ($\hat{K} = 2$) yielding a p-value of 0.00003 according to (2). Thus, both tests strongly reject the null hypothesis (2). A complementary analysis helps understanding what aspect of the Gaussian is not supported by the data. We found $\mathcal{R}_1 = 0.15$ ($p = 0.69$) with a skewness coefficient of 0.13. Thus there is no reason to suspect an asymmetrical distribution for the innovations. On the other hand, we can notice that 9.3% of the absolute standardized residuals are greater than 2.5 and the kurtosis is 4.33. Thus, if the model entertained above is correct, the conclusion that emerges from the present analysis is that the $\bigtriangledown\tau_t$ series could have been generated from innovations with a symmetric distribution having fatter tails than the Gaussian.

ACKNOWLEDGMENTS
The authors would like to thank Dr. B.L. Shea for some insight on subroutine G13DCF of the NAG library and for providing them with the data set used in Section 5.

APPENDIX A

We show that(8) holds under H_0. Assume p and $q > 0$. It suffices to show that

$$\frac{1}{T}\sum_{t=1}^{T}\frac{\partial}{\partial\sigma}L_k(U_t) \xrightarrow{P} E\left[\frac{\partial}{\partial\sigma}L_k(U_t)\right] = -\frac{1}{\sigma}b_k, \tag{A.1.a}$$

$$\frac{1}{T}\sum_{t=1}^{T}\frac{\partial}{\partial\varphi_1}L_k(U_t) \xrightarrow{P} E\left[\frac{\partial}{\partial\varphi_1}L_k(U_t)\right] = 0, \tag{A.1.b}$$

$$\frac{1}{T}\sum_{t=1}^{T}\frac{\partial}{\partial\theta_1}L_k(U_t) \xrightarrow{P} E\left[\frac{\partial}{\partial\theta_1}L_k(U_t)\right] = 0. \tag{A.1.c}$$

First,

$$\frac{\partial}{\partial\sigma}L_k(U_t) = -\frac{2\epsilon_t}{\sigma^2}\phi\left(\frac{\epsilon_t}{\sigma}\right)L_k'(x)|_{x=2\Phi\left(\frac{\epsilon_t}{\sigma}\right)-1} = -\frac{\epsilon_t}{\sigma^2}w\left(\frac{\epsilon_t}{\sigma}\right) \text{ say.}$$

The law of large numbers yields (A.1.a). For (A.1.b), define for $r \geq 0$,

$$B_{r-1} = \frac{\partial}{\partial \varphi_1} \delta_r(\boldsymbol{\theta}, \boldsymbol{\varphi}),$$

where, setting $\varphi_0 = -1$, $\gamma_0 = \theta_0 = 1$, we have

$$\delta_r(\boldsymbol{\theta}, \boldsymbol{\varphi}) = \delta_r = \sum_{i=0}^{\min(r,p)} \varphi_i \gamma_{r-i} \;\; r \geq 0, \tag{A.2}$$

$$\gamma_r = -\sum_{i=1}^{\min(r,q)} \gamma_{r-i} \theta_i \;\; r \geq 1. \tag{A.3}$$

Obviously $B_{r-1} = \gamma_{r-1}$ when $r \geq 1$. For $r \geq q$, from Brockwell and Davis (1991), p.107,

$$\gamma_r = \sum_{i=1}^{j} \sum_{n=0}^{r_i - 1} c_{in} r^n \alpha_i^{-r}$$

for some constants c_{in} and where the α_i's are the j distinct roots of $1 + \theta_1 z + ... + \theta_q z^q$ and r_i is the multiplicity of α_i, $i = 1, ..., j$. Thus, when $r \geq q + 1$,

$$B_{r-1} = \sum_{i=1}^{j} \sum_{n=0}^{r_i - 1} c_{in} (r-1)^n \alpha_i^{-r+1}. \tag{A.4}$$

If $(X_t, t \in \mathbb{Z})$ is a weak stationary process such that $\text{Cov}(X_t, X_{t+h}) \to 0$ as $h \to \infty$, then $\overline{X}_T \xrightarrow{P} E(X_t)$. We apply this result with $X_t = \partial L_k(U_t)/\partial \varphi_1$. From (1) and (1), we have

$$X_t = \frac{1}{\sigma} w\left(\frac{\epsilon_t}{\sigma}\right) \frac{\partial}{\partial \varphi_1} \epsilon_t = -\frac{1}{\sigma} w\left(\frac{\epsilon_t}{\sigma}\right) \left(Y_{t-1} - \sum_{r=0}^{\infty} \left(\frac{\partial}{\partial \varphi_1} \delta_r \right) \boldsymbol{\theta}^{\mathsf{T}} \boldsymbol{Y}_{t-1-r}^{(q)} \right). \tag{A.5}$$

Thus $E(X_t) = 0$. Moreover, $\text{Var}(X_t) < \infty$ as shown in Appendix C and it is seen by Lemma C.1 that $\text{Cov}(X_t, X_{t+h})$ depends only on h. Thus $(X_t, t \in \mathbb{Z})$ is stationary. We show that $\text{Cov}(X_t, X_{t+h}) \to 0$ as $h \to \infty$. From (A.5), for h large, $|Cov(X_t, X_{t+h})| = |d_1| E|w(\epsilon_{t+h}/\sigma)|/\sigma$, where

$$d_1 = E\left[\frac{1}{\sigma} w\left(\frac{\epsilon_t}{\sigma}\right) \left\{ Y_{t-1} - \sum_{r=0}^{\infty} B_{r-1} \boldsymbol{\theta}^{\mathsf{T}} \boldsymbol{Y}_{t-1-r}^{(q)} \right\} \left\{ Y_{t+h-1} - \sum_{r=0}^{\infty} B_{r-1} \boldsymbol{\theta}^{\mathsf{T}} \boldsymbol{Y}_{t+h-1-r}^{(q)} \right\} \right]. \tag{A.6}$$

But, $|d_1| \leq d_2 + \sum_{j=1}^{q} |\theta_j| (d_{3j} + d_{4j}) + \sum_{i=1}^{q} \sum_{j=1}^{q} |\theta_i \theta_j| d_{5ij}$ where

$$d_2 = \left| E\left[\frac{1}{\sigma} w\left(\frac{\epsilon_t}{\sigma}\right) Y_{t-1} Y_{t+h-1} \right] \right|, \; d_{3j} = \sum_{r=0}^{\infty} \left| B_{r-1} E\left[\frac{1}{\sigma} w\left(\frac{\epsilon_t}{\sigma}\right) Y_{t-r-j} Y_{t+h-1} \right] \right|,$$

$$d_{4j} = \sum_{r=0}^{\infty} \left| B_{r-1} E\left[\frac{1}{\sigma} w\left(\frac{\epsilon_t}{\sigma}\right) Y_{t-1} Y_{t+h-r-j} \right] \right|$$

and

$$d_{5ij} = \sum_{r=0}^{\infty} \sum_{r'=1}^{\infty} \left| B_{r-1} B_{r'-1} E\left[\frac{1}{\sigma} w\left(\frac{\epsilon_t}{\sigma}\right) Y_{t-r-i} Y_{t+h-r'-j} \right] \right|.$$

It can be shown that d_2, d_{3j}, d_{4j} and $d_{5ij} \to 0$ when $h \to \infty$. Proof for d_{4j}, which is typical, is sketched in Appendix D. This yields (A.1.b).

As for (A.1.c), let $A_r = \frac{\partial}{\partial\theta_1}\delta_r(\boldsymbol{\theta}, \boldsymbol{\varphi})$. From (A.3), we obtain, for $r \geq q$, the system

$$\begin{cases} \gamma'_r + \theta_1\gamma'_{r-1} + ... + \theta_q\gamma'_{r-q} = -\gamma_{r-1} \\ \gamma_r + \theta_1\gamma_{r-1} + ... + \theta_q\gamma_{r-q} = 0 \end{cases}$$

from which we find

$$0 = \sum_{j=0}^{q}\theta_j\left(-\sum_{i=0}^{q}\theta_i\gamma'_{r-j-i+1}\right) = \sum_{h=0}^{2q} a_h\gamma'_{r-h+1} \text{ where } a_h = \sum_{\substack{i+j=h \\ 0\leq i,j\leq q}}\theta_i\theta_j,\ \forall r \geq 2q-1.$$

Again from Brockwell and Davis (1991), p.107, we have, for some constants d_{in}

$$\gamma'_r = \sum_{i=1}^{j}\sum_{n=0}^{s_i-1} d_{in}r^n\beta_i^{-r}$$

where the β_i's are the j distinct roots (with multiplicity s_i) of $1+a_1z+a_2z^2+...+a_{2q}z^{2q}$. Now

$$\left(\sum_{i=0}^{q}\theta_i z\right)^2 = \sum_{h=0}^{2q}\left(\sum_{\substack{i+j=h \\ 0\leq i,j\leq q}}\theta_i\theta_j\right)z^h = \sum_{h=0}^{2q} a_h z^h$$

where $a_0 = \theta_0^2 = 1$. This shows that the roots of $1 + a_1z + a_2z^2 + ... + a_{2q}z^{2q}$ are exactly the same than that of $1 + \theta_1z + \theta_2z^2 + ... + \theta_qz^q$, apart from the multiplicity. Thus, we obtain

$$A_r = \sum_{l=0}^{p}\sum_{i=1}^{j}\sum_{n=0}^{s_i-1} d_{in}(r-l)^n\alpha_i^{-(r-l)}\varphi_l, \text{ for all } r \geq \max(2q, p). \tag{A.7}$$

By the same argument, using A_r of (A.7) instead of B_{r-1} of (A.4), we get (A.1.c).

APPENDIX B

We show that $E(\boldsymbol{V}_t) = 0$ and $\text{Var}(\boldsymbol{B}\boldsymbol{V}_t) = \boldsymbol{I}_K - \boldsymbol{b}_K\boldsymbol{b}_K^\intercal/2$. In view of (1) and (1),

$$\frac{\partial}{\partial\boldsymbol{\varphi}}\text{Log}\left(\frac{1}{\sigma}\phi\left(\frac{\epsilon_t}{\sigma}\right)\right) = \frac{\epsilon_t}{\sigma^2}\left[\boldsymbol{Y}_{t-1}^{(p)} - \sum_{r=0}^{\infty}\left(\frac{\partial}{\partial\boldsymbol{\varphi}}\delta_r\right)\boldsymbol{\theta}^\intercal\boldsymbol{Y}_{t-1-r}^{(q)}\right],$$

$$\frac{\partial}{\partial\boldsymbol{\theta}}\text{Log}\left(\frac{1}{\sigma}\phi\left(\frac{\epsilon_t}{\sigma}\right)\right) = \frac{\epsilon_t}{\sigma^2}\left[\boldsymbol{\epsilon}_{t-1}^{(q)} - \sum_{r=0}^{\infty}\left(\frac{\partial}{\partial\boldsymbol{\theta}}\delta_r\right)\boldsymbol{\theta}^\intercal\boldsymbol{Y}_{t-1-r}^{(q)}\right]$$

and

$$\frac{\partial}{\partial\sigma}\text{Log}\left(\frac{1}{\sigma}\phi\left(\frac{\epsilon_t}{\sigma}\right)\right) = \frac{1}{\sigma}\left(\left(\frac{\epsilon_t}{\sigma}\right)^2 - 1\right).$$

It follows that $E(\boldsymbol{V}_t) = 0$ under H_0. Moreover, under H_0, $\text{Var}(\boldsymbol{L}_t) = \boldsymbol{I}_K$. Thus,

$$\text{Cov}\left(\boldsymbol{L}_t, \frac{\partial}{\partial\boldsymbol{\beta}}\text{Log}\left(\frac{1}{\sigma}\phi\left(\frac{\epsilon_t}{\sigma}\right)\right)\right)^\intercal = \begin{bmatrix} \boldsymbol{0} \\ 0 \\ \frac{1}{\sigma}\boldsymbol{b}_K^\intercal \end{bmatrix} = \mathcal{J}_K^\intercal.$$

Finally, Var $\left(\frac{\partial}{\partial \boldsymbol{\beta}} \text{Log}\left(\frac{1}{\sigma}\phi\left(\frac{\epsilon_t}{\sigma}\right)\right)\right) = \mathcal{I}_\beta = \begin{bmatrix} \boldsymbol{C} & \mathbf{0} \\ \mathbf{0} & \frac{2}{\sigma^2} \end{bmatrix}$, for some matrix $\boldsymbol{C}$ whose exact expression is not needed. Thus

$$\text{Var}(\boldsymbol{V}_t) = \begin{bmatrix} \boldsymbol{I}_K & \mathcal{I}_K \\ \mathcal{I}_K^\intercal & \mathcal{I}_\beta \end{bmatrix}.$$

APPENDIX C

We show that $\text{Var}(X_t) < \infty$. Without loss of generality, set $\sigma = 1$. This will be assumed here and in the next appendix. Since Y_t is causal, we can write $Y_t = \sum_{j=0}^{\infty} \psi_j \epsilon_{t-j}$ and from (A.5)

$$\text{Var}(X_t) = E(w(\epsilon_t))^2 E\left(Y_{t-1} - \sum_{r=0}^{\infty} B_{r-1} \boldsymbol{\theta}^\intercal \boldsymbol{Y}_{t-1-r}^{(q)}\right)^2 = E(w(\epsilon_t))^2 E\left(\sum_{h=1}^{\infty} d_h \epsilon_{t-h}\right)^2$$

where $d_h = \psi_{h-1} - \sum_{\substack{r+j+l=h \\ 1\le j\le q \\ 0\le r,l\le h-1}} \psi_l \gamma_{r-1} \theta_j$. We now need the following lemma.

Lemma 1. *If the ARMA process (1) is causal and invertible, then* $\sum_{h=1}^{\infty} |d_h| < \infty$.

PROOF.

From (A.3), $\sum_{\substack{r+j+l=h \\ 1\le j\le q \\ 0\le r,l\le h-1}} \psi_l \gamma_{r-1} \theta_j = \sum_{k=1}^{h} \psi_{h-k} \left(\sum_{\substack{r+j=k \\ 1\le j\le q \\ 0\le r\le h-1}} \gamma_{r-1} \theta_j \right) = -\sum_{k=1}^{h} \psi_{h-k} \gamma_{k-1}$.
We have also $\sum_{h=1}^{\infty} \sum_{k=1}^{h} |\psi_{h-k}\gamma_{k-1}| = \sum_{h=0}^{\infty} \sum_{k=0}^{h} |\psi_{h-k}\gamma_k| = \sum_{k=0}^{\infty} |\gamma_k| \sum_{h=0}^{\infty} |\psi_h|$.
Thus, $\sum_{h=1}^{\infty} |d_h| \le \sum_{h=1}^{\infty} |\psi_{h-1}| + \sum_{h=1}^{\infty} \sum_{k=1}^{h} |\psi_{h-k}\gamma_{k-1}| = \sum_{h=0}^{\infty} |\psi_h| \left(\sum_{k=0}^{\infty} |\gamma_k| + 1\right)$.
But from Brockwell and Davis (1991), p.87, $\sum_{k=0}^{\infty} |\gamma_k|$ is finite. Since under the assumption of Theorem 1, $\sum_{j=0}^{\infty} |\psi_j| < \infty$ the lemma follows. □

From this lemma, we conclude that $E\left[\sum_{h=1}^{\infty} d_h \epsilon_{t-h}\right]^2 = \sum_{h=1}^{\infty} d_h^2 < \infty$. Since

$$E(w(\epsilon_t))^2 = 4 \int (L_k'(2\Phi(x) - 1))^2 \phi^3(x) dx < \infty,$$

the result follows.

APPENDIX D

Here we sketch the proof that the typical element d_{4j} of inequality (A.6) vanishes. From $Y_t = \sum_{j=0}^{\infty} \psi_j \epsilon_{t-j}$ and the fact that the remainder of a convergent series converges toward 0, we have

$$\lim_{h\to\infty} d_{4j} = \lim_{h\to\infty} \sum_{r=0}^{\infty} |B_{r-1} E\left(w(\epsilon_t) Y_{t-1} Y_{t+h-r-j}\right)| \leq$$

$$\lim_{h\to\infty} |E(w(\epsilon_t))| \sum_{r=0}^{h-j} \left| B_{r-1} \sum_{a=0}^{\infty} \psi_a \psi_{a+h-j-r+1} \right|$$

$$\leq |E(w(\epsilon_t))| \lim_{h\to\infty} \left[\sum_{a=0}^{m-1} |\psi_a| \sum_{r=0}^{a+h-j-m} |B_{r-1}\psi_{a+h-j-r+1}| + \sum_{r=a+h-j-m+1}^{h-j} |B_{r-1}\psi_{a+h-j-r+1}| \right.$$

$$\left. + \sum_{r=0}^{h-j} |B_{r-1}| \sum_{a=m}^{\infty} |\psi_a \psi_{a+h-j-r+1}| \right] \quad \text{(D.1)}$$

where $m = \max\{p, q+1\} - p$. For the first term in the limit of (D.1), using the expression for B_{r-1} in (A.4) and that of $\psi_{a+h-j-r+1}$ given in Brockwell and Davis (1991) eq. (3.3.6), we have

$$\sum_{r=q+1}^{a+h-j-m} |B_{r-1}\psi_{a+h-j-r+1}| =$$

$$\sum_{r=q+1}^{a+h-j-m} \left| \sum_{b=1}^{k} \sum_{l=0}^{r_b-1} c_{bl} r^l \alpha_a^{-r} \sum_{b'=1}^{k'} \sum_{l'=0}^{r_{b'}-1} \alpha_{b'l'} (a+h-j-r+1)^{l'} \xi_{b'}^{-(a+h-j-r+1)} \right|$$

$$\leq \sum_{b=1}^{k} \sum_{l=0}^{r_b-1} \sum_{b'=1}^{k'} \sum_{l'=0}^{r_{b'}-1} \sum_{d=0}^{l'} \binom{l'}{d} \left\{ |c_{bl}\alpha_{b'l'}| |\xi_{b'}^{-(a+h-j+1)} (a+h-j+1)^{l'-d}| \right.$$

$$\left. \times \sum_{r=q+1}^{a+h-j-m} r^{l+d} |\alpha_a|^{-r} |\xi_{b'}|^r \right\}.$$

If $|\xi_{b'}| < |\alpha_a|$, the term in braces $\to 0$ as $h \to \infty$. Let $|\alpha_a| = 1 + \epsilon_1 < |\xi_{b'}| = 1 + \epsilon_2$ with $\epsilon_1, \epsilon_2 > 0$.

$$\left| \frac{(a+h-j+1)^{l'-d}}{\xi_{b'}^{(a+h-j+1)}} \right| \sum_{r=q+1}^{a+h-j-m} r^{l+d} |\alpha_a|^{-r} |\xi_{b'}|^r \leq \frac{|a+h-j+1|^{l'-d}}{|\xi_{b'}|^{a+h-j+1}} \sum_{r=0}^{a+h-j+1} r^{l+d} \left(\frac{|\xi_{b'}|}{|\alpha_a|} \right)^r. \quad \text{(D.2)}$$

For all $\epsilon > 0$, there exist a C, C' such that the left-hand side of (D.2) is bounded above by

$$C \frac{|a+h-j+1|^{l'-d}}{|\xi_{b'}|^{a+h-j+1}} \frac{\left(\frac{|\xi_{b'}|}{|\alpha_a|} + \epsilon \right)^{h+a-j+2} - 1}{\frac{|\xi_{b'}|}{|\alpha_a|} + \epsilon - 1} \leq C' |a+h-j+1|^{l'-d} \left(\frac{\frac{|\xi_{b'}|}{|\alpha_a|} + \epsilon}{|\xi_{b'}|} \right)^{a+h-j+2}. \quad \text{(D.3)}$$

In (D.3), take $\epsilon > 0$ smaller than $\epsilon_1(1+\epsilon_2)/(1+\epsilon_1)$. Then the right hand side of (D.3) converges to 0 as $h \to \infty$. This shows that the first term in the limit of (D.1) converges to 0. It follows that the second term also converges toward 0. As for the last term in the limit, a similar argument yields that all terms on the right hand side of (D.1) converge to 0 so that $d_{4j} \to 0$.

BIBLIOGRAPHY

[1] Anděl, J., 1997. On residual analysis for time series models. Kybernetika 33 (2), 161–170. 27

[2] Beiser, A., 1985. Distributions of $\sqrt{b_1}$ and b_2 for autoregressive errors. Ph.D. thesis, Boston University. 27

[3] Brockett, P. L., Hinich, M. J., Patterson, D., 1988. Bispectral-based tests for the detection of gaussianity and linearity in time series. J. Amer. Statist. Assoc. 83, 657–664. 26, 33

[4] Brockwell, P. J., Davis, R. A., 1991. Time series: Theory and Methods, 2nd Edition. Springer-Verlag New York. 26, 27, 28, 42, 43, 44, 45

[5] Burn, D., 1987. Simulation of stationary time series. Proceedings of the 1987 Winter Simulation Conference , 289–294. 39

[6] Cromwell, J. B., Labys, W. C., Terraza, M., 1994. Univariate tests for time-series models. Sage Publications Inc, Thousand Oaks, California. 36

[7] D'Agostino, R. B., Stephens, M. A., 1986. Goodness-of-fit techniques. Statistics: Text-BOOKs and Monographs, 68, New York: Marcel Dekker. 27, 36

[8] Ducharme, G., Lafaye de Micheaux, P., 2002. Goodness-of-fit tests of normality for the innovations in arma models. Tech. rep., Technical report #02-02, Université Montpellier II. 29, 34, 39

[9] Epps, T. W., 1987. Testing that a stationnary time series is gaussian. Ann. Statist. 15 (4), 1683–1698. 26

[10] Frances, P., 1998. Time series models for business and economic forecasting. Cambridge University Press, Cambridge. 36

[11] Gasser, T., 1975. Goodness-of-fit tests for correlated data. Biometrika 62, 563–570. 26

[12] Gouriéroux, C., Monfort, A., 1995. Séries temporelles et modèles dynamiques, 2nd Edition. Economica. 29

[13] Granger, C. W. J., 1976. Tendency towards normality of linear combinations of random variables. Metrika 23 (4), 237–248. 26

[14] Henze, N., 1997. Do components of smooth tests of fit have diagnostic properties? Metrika 45, 121–130. 30

[15] Heuts, R., Rens, S., 1986. Testing normality when observations satisfy a certain low order arma-scheme. Computat. Statist. Quarterly 1, 49–60. 26

[16] Hinich, M. J., 1982. Testing for gaussianity and linearity of a stationary time series. J. Time Ser. Anal. 3 (3), 169–176. 26

[17] Hipel, K. W., McLeod, A. I., 1994. Time series modelling of water resources and environmental systems. [Elsevier Science Publishing Co., New York; North-Holland Publishing Co., Amsterdam] (New York; Amsterdam). 26, 27

[18] Janic-Wroblewska, A., Ledwina, T., 2000. Data driven rank test for two-sample problem. Scand. J. Statist. 27, 281–298. 31

[19] Jarque, C., Bera, A., 1987. A test for normality of observations and regression residuals. Internat. Statist. Review 55 (2), 163–172. 36

[20] Kallenberg, W., Ledwina, T., 1997a. Data driven smooth tests for composite hypotheses: comparison of powers. J. Statist. Comput. Simul. 59 (2), 101–121. 31, 32

[21] Kallenberg, W., Ledwina, T., 1997b. Data-driven smooth tests when the hypothesis is composite. J. Amer. Statist. Assoc. 92 (439), 1094–1104. 31, 32, 39

[22] Koul, H. L., Stute, W., 1999. Nonparametric model checks for time series. Ann. Statist. 27 (1), 204–236. 26
[23] Ledwina, T., 1994. Data-driven version of Neyman's smooth test of fit. J. Amer. Statist. Assoc. 89 (427), 1000–1005. 27, 31
[24] Lee, S., Na, S., 2002. On the Bickel-Rosenblatt test for first-order autoregressive models. Statist. Probab. Lett. 56 (1), 23–35. 27
[25] Lomnicki, Z., 1961. Tests for departure from normality in the case of linear stochastic processes. Metrika 4, 37–62. 26
[26] Lutkepohl, H., Schneider, W., 1989. Testing for normality of autoregressive time series. Comput. Statist. Quaterly 2, 151–168. 26, 27, 37
[27] Moore, D. S., 1982. The effect of dependence on chi squared tests of fit. Ann. Statist. 10 (4), 1163–1171. 26
[28] Neyman, J., 1937. Smooth test for goodness of fit. Skand. Aktuar. 20, 149–199. 27
[29] Ojeda, R., Cardoso, J., Moulines, E., 1997. Asymptotically invariant gaussianity test for causal invertible time series. Proc. of IEEE international conference on Acoustics, Speech and Signal Processing 5, 3713–3716. 27
[30] Pierce, D. A., Gray, R. J., 1985. Goodness-of-fit tests for censored survival data. Ann. Statist. 13 (2), 552–563. 26, 27, 30, 36
[31] Rayner, J., Best, D., 1989. Smooth Tests of Goodness-of-Fit. Oxford: Oxford University Press. 31
[32] Sansone, G., 1959. Orthogonal functions. New York: Interscience. 28
[33] Schwarz, G., 1978. Estimating the dimension of a model. Ann. Statist. 6 (2), 461–464. 31
[34] Shea, B. L., 1987. Estimation of multivariate time series. J. Time Ser. Anal. 8 (1), 95–109. 41
[35] Weisberg, S., Bingham, C., 1975. An approximate analysis of variance test for non-normality suitable for machine calculation. Technometrics 17, 133–134. 36

JAVASCRIPT APPLICATION

This little Javascript program permits to compute the last test statistic in Theorem 1, and the p-value of the test. For this, you have to input the values of K , $\hat{\sigma}$ and then the $\hat{\epsilon}$'s separated by spaces.

Leave no space after K: K

$\hat{\sigma}$ followed by a single space: sigchap

The $\hat{\epsilon}$'s separated from each other by a single space. Leave no space after the last one:
epschaps

Click to Submit: Submit

Statistic value:

p-value:

CHAPITRE 3

A multivariate empirical characteristic function test of independence with normal marginals

Cet article a été soumis à la revue *Journal of Multivariate Analysis*.

Comme la coutume dans cette discipline le veut, l'ordre alphabétique des auteurs a été respecté.

Voici la liste des contributions principales de Pierre Lafaye de Micheaux à cet article :

- Conduite d'une importante recherche bibliographique.
- Démonstration des Théorèmes 2.1, 2.3, 3.1, 3.2 et 3.3.
- Démonstration d'une partie de la preuve du Théorème 2.2.
- Démonstration de l'essentiel de la preuve du Théorème 2.4.
- Conduite générale de la recherche pour l'intégralité de la section 4.
- Conception, écriture et validation des programmes C++ et simulations.
- Rédaction d'une bonne partie de l'article en anglais.

A multivariate empirical characteristic function test of independence with normal marginals

Martin Bilodeau and Pierre Lafaye de Micheaux

Département de mathématiques et de statistique, Université de Montréal, Canada

Abstract
This paper proposes a semi-parametric test of independence (or serial independence) between marginal vectors each of which is normally distributed but without assuming the joint normality of these marginal vectors. The test statistic is a Cramér-von Mises functional of a process defined from the empirical characteristic function. This process is defined similarly as the process of Ghoudi et al. (2001) built from the empirical distribution function and used to test for independence between univariate marginal variables. The test statistic can be represented as a V statistic. It is consistent to detect any form of dependence. The weak convergence of the process is derived. The asymptotic distribution of the Cramér-von Mises functionals are approximated by the Cornish-Fisher expansion using a recursive formula for cumulants and by the numerical evaluations of the eigenvalues in the inversion formula. The test statistic is finally compared with Wilks' statistic for testing the parametric hypothesis of independence in the one-way MANOVA model with random effects.

Key words: Characteristic function, Independence, Multivariate Analysis, Serial independence, Stochastic processes
1991 MSC: 62H15, 62M99

1. INTRODUCTION

Different characterizations have led to various tests of independence. Let $p \geq 1$ be a fixed integer. Consider a partitioned random vector $\boldsymbol{\epsilon} = \left(\boldsymbol{\epsilon}^{(1)}, \ldots, \boldsymbol{\epsilon}^{(p)}\right)$ made up of p q-dimensional subvectors and a corresponding partitioned $\boldsymbol{t} = \left(\boldsymbol{t}^{(1)}, \ldots, \boldsymbol{t}^{(p)}\right)$, for any fixed vector $\boldsymbol{t}$. Independence of the subvectors may be characterized with the joint distribution function or characteristic function as

$$K_p(\boldsymbol{t}) = \prod_{k=1}^{p} K^{(k)}(\boldsymbol{t}^{(k)}), \tag{1.1}$$

$$C_p(\boldsymbol{t}) = \prod_{k=1}^{p} C^{(k)}(\boldsymbol{t}^{(k)}), \tag{1.2}$$

where K_p and C_p are, respectively, the joint distribution function and joint characteristic function. The marginal versions are $K^{(k)}$ and $C^{(k)}$ for $k = 1, \ldots, p$. In the univariate setting ($q = 1$) Blum et al. (1961) proposed an empirical process based on

(1.1), whereas Csörgő (1981a) defined a similar process based on (1.2). Feuerverger (1993) proposes an empirical characteristic function version of the Blum et al. (1961) test statistic. He points out difficulties with dimensions above 2.

Recently, in the univariate setting, Ghoudi et al. (2001) introduced a new process based on their novel characterization of independence which is now presented. This characterization for $p = 3$ is implicit in the paper of Blum et al. (1961). For any $A \subset I_p = \{1, \ldots, p\}$ and any $\boldsymbol{t} \in \mathbb{R}^p$, let

$$\mu_A(\boldsymbol{t}) = \sum_{B \subset A} (-1)^{|A \setminus B|} K_p(\boldsymbol{t}^B) \prod_{j \in A \setminus B} K^{(j)}(\boldsymbol{t}^{(j)}).$$

The notation $|A|$ stands for cardinality of the set A and the convention $\prod_{\emptyset} = 1$ is adopted. The vector $\boldsymbol{t}^B$ is used to make a selection of components of $\boldsymbol{t}$ according to the set B,

$$(\boldsymbol{t}^B)^{(i)} = \begin{cases} \boldsymbol{t}^{(i)}, & i \in B; \\ \infty, & i \in I_p \setminus B. \end{cases}$$

Independence can be characterized: $\boldsymbol{\epsilon}^{(1)}, \ldots, \boldsymbol{\epsilon}^{(p)}$ are independent if and only if $\mu_A \equiv 0$, for all $A \subset I_p$ satisfying $|A| > 1$. Cramér-von Mises type functional of an associated process then leads them to a non-parametric test of independence in the non-serial or serial situation. The interest in their process resides in the simple form of the covariance which is expressed as a product of covariance functions of Brownian bridge.

In the multivariate setting ($q \geq 1$), the present paper proposes tests of independence, when subvectors or marginals are normally distributed, built from a process relying on a similar independence characterization based on characteristic functions. Note that the subvectors are not assumed to be jointly multinormal in which case independence can be tested parametrically with covariances using likelihood ratio tests. Namely, the marginals $\boldsymbol{\epsilon}^{(1)}, \ldots, \boldsymbol{\epsilon}^{(p)}$ are independent if and only if $\mu_A \equiv 0$, for all $A \subset I_p$, $|A| > 1$. Here,

$$\mu_A(\boldsymbol{t}) = \sum_{B \subset A} (-1)^{|A \setminus B|} C_p(\boldsymbol{t}^B) \prod_{j \in A \setminus B} C^{(j)}(\boldsymbol{t}^{(j)}),$$

where

$$(\boldsymbol{t}^B)^{(i)} = \begin{cases} \boldsymbol{t}^{(i)}, & i \in B; \\ \mathbf{0}, & i \in I_p \setminus B. \end{cases}$$

Note that the subvectors are not assumed to be jointly multinormal in which case independence can be tested parametrically with covariances using likelihood ratio tests. Normality of the marginals will often be approximately satisfied. For example, data analysis of regression models on which Box-Cox transformations are done in the first stage are common in practice. One should bear in mind, however, that the asymptotic of the tests of independence proposed here would not consider the Box-Cox transformation as data dependent but as a fixed transformation. Goodness-of-fit tests of normality after a data dependent Box-Cox transformation by Chen et al. (2002) is a result in this direction; it will not be pursued here for tests of independence.

The non-serial and serial problems are considered. It is shown that the asymptotic distribution of the proposed process is the same in both cases under the null hypothesis

of independence. Moreover, it is established that the estimation of the unknown mean vector and positive definite covariance matrix of the normal marginals does not affect the asymptotic distribution of the process. The proposed Cramér-von Mises type of test statistic is related to V-statistics for which de Wet and Randles (1987) studied the effect of estimating unknown parameters. Norm on Euclidian spaces $\mathbb{R}^m$ will be denoted $\|\cdot\|$, whereas $|\cdot|$ will be the norm on the complex field $\mathbb{C}$. We also let $C(\mathbb{R}^{pq}, \mathbb{C})$ be the space of continuous functions from $\mathbb{R}^{pq}$ to $\mathbb{C}$.

2. TESTING INDEPENDENCE: THE NON-SERIAL SITUATION

2.1. The case of known parameters

Let $\boldsymbol{\epsilon} = \left(\boldsymbol{\epsilon}^{(1)}, \ldots, \boldsymbol{\epsilon}^{(p)}\right) \in \mathbb{R}^{pq}$ denote a partition into p subvectors and $\boldsymbol{\epsilon}_1, \ldots, \boldsymbol{\epsilon}_n$ be an i.i.d. sample of such (pq)-dimensional random vectors. Suppose that the subvectors of the random vectors $\boldsymbol{\epsilon}_i$ all have the same $N_q(\mathbf{0}, \boldsymbol{I})$ normal distribution, with characteristic function ϕ. The problem is that of testing the independence of the marginals, that is the independence of $\boldsymbol{\epsilon}^{(1)}, \ldots, \boldsymbol{\epsilon}^{(p)}$. This non-serial problem with known parameters is of very limited practical importance. However, it serves as a prototype on which subsequent results are based.

Following Ghoudi et al. (2001), for any $A \subset I_p = \{1, \ldots, p\}$ and any $\boldsymbol{t} = (\boldsymbol{t}^{(1)}, \ldots, \boldsymbol{t}^{(p)}) \in \mathbb{R}^{pq}$, let

$$R_{n,A}(\boldsymbol{t}) = \sqrt{n} \sum_{B \subset A} (-1)^{|A \setminus B|} \phi_{n,p}(\boldsymbol{t}^B) \prod_{i \in A \setminus B} \phi(\boldsymbol{t}^{(i)}) \tag{2.1}$$

where

$$\phi_{n,p}(\boldsymbol{t}) = \frac{1}{n} \sum_{j=1}^{n} \exp(\mathrm{i}\langle \boldsymbol{t}, \boldsymbol{\epsilon}_j \rangle) \tag{2.2}$$

is the empirical characteristic function of the sample. The notation $\langle \cdot, \cdot \rangle$ is for the usual inner product between vectors.

The asymptotic behaviour of these processes is stated next. It states that for different subsets A the associated processes are asymptotically independent, each process being asymptotically Gaussian with a covariance function of a particularly simple form. Specifically, the covariance function is a product of covariance functions of the type encountered by Feuerverger and Mureika (1977), Csörgő (1981a) or Marcus (1981), for the empirical characteristic function process. Another process defined by Csörgő (1985) has a covariance of a more complicated structure.

Theorem 2.1. *If $\boldsymbol{\epsilon}_1^{(1)}, \ldots, \boldsymbol{\epsilon}_1^{(p)}$ are independent, the collection of processes $\{R_{n,A} : |A| > 1\}$ converge in $C(\mathbb{R}^{pq}, \mathbb{C})$ to independent zero mean complex Gaussian processes $\{R_A : |A| > 1\}$ having covariance function given by*

$$C_A(\boldsymbol{s}, \boldsymbol{t}) = \prod_{k \in A} [\phi(\boldsymbol{t}^{(k)} - \boldsymbol{s}^{(k)}) - \phi(\boldsymbol{t}^{(k)}) \phi(\boldsymbol{s}^{(k)})] \tag{2.3}$$

and pseudo-covariance function given by

$$\overline{C}_A(\boldsymbol{s}, \boldsymbol{t}) = E[R_A(\boldsymbol{s}) R_A(\boldsymbol{t})] = C_A(-\boldsymbol{s}, \boldsymbol{t}). \tag{2.4}$$

The multinomial formula (Ghoudi et al. (2001)) yields the equivalent representation

$$R_{n,A}(\boldsymbol{t}) = \frac{1}{\sqrt{n}} \sum_{j=1}^{n} \prod_{k \in A} [\exp(\mathrm{i}\langle \boldsymbol{t}^{(k)}, \boldsymbol{\epsilon}_j^{(k)} \rangle) - \phi(\boldsymbol{t}^{(k)})]. \tag{2.5}$$

This i.i.d. average representation is used in the proof of Theorem 2.1.

2.2. The case of unknown parameters

The context is the same as in the preceeding subsection except that the components of the random vectors $\boldsymbol{\epsilon}_i$ in the sample now have all the same $N_q(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ normal distribution, where $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$, positive definite, are unknown. The problem again is that of testing for independence of the marginals, that is the independence of the components $\boldsymbol{\epsilon}^{(1)}, \ldots, \boldsymbol{\epsilon}^{(p)}$.

First, define the standardized residual vectors $\boldsymbol{e}_j^{(k)} = \boldsymbol{S}^{-\frac{1}{2}}(\boldsymbol{\epsilon}_j^{(k)} - \overline{\boldsymbol{\epsilon}})$, where $\boldsymbol{S} = \frac{1}{np} \sum_{j=1}^{n} \sum_{k=1}^{p} (\boldsymbol{\epsilon}_j^{(k)} - \overline{\boldsymbol{\epsilon}})(\boldsymbol{\epsilon}_j^{(k)} - \overline{\boldsymbol{\epsilon}})^{\mathsf{T}}$ and $\overline{\boldsymbol{\epsilon}} = \frac{1}{np} \sum_{j=1}^{n} \sum_{k=1}^{p} \boldsymbol{\epsilon}_j^{(k)}$ are, respectively, the sample covariance matrix and the sample mean. Also, let $\overline{\boldsymbol{\epsilon}}^{(k)} = \frac{1}{n} \sum_{j=1}^{n} \boldsymbol{\epsilon}_j^{(k)}$ be the sample mean of the kth subvectors.

The underlying process is the same apart from the unknown parameters which are replaced by their sample estimates. The plug-in process is thus

$$\hat{R}_{n,A}(\boldsymbol{t}) = \sqrt{n} \sum_{B \subset A} (-1)^{|A \setminus B|} \hat{\phi}_{n,p}(\boldsymbol{t}^B) \prod_{i \in A \setminus B} \phi(\boldsymbol{t}^{(i)}) \tag{2.6}$$

where

$$\hat{\phi}_{n,p}(\boldsymbol{t}) = \frac{1}{n} \sum_{j=1}^{n} \exp(\mathrm{i}\langle \boldsymbol{t}, \boldsymbol{e}_j \rangle) \tag{2.7}$$

is the empirical characteristic function based on standardized residuals $\boldsymbol{e}_j = \left(\boldsymbol{e}_j^{(1)}, \ldots, \boldsymbol{e}_j^{(p)}\right) \in \mathbb{R}^{pq}$. The asymptotic behaviour of these processes is stated next, the main conclusion being that the estimation of the unknown parameters does not affect the asymptotic distribution.

Theorem 2.2. *If $\boldsymbol{\epsilon}_1^{(1)}, \ldots, \boldsymbol{\epsilon}_1^{(p)}$ are independent, the processes $\{\hat{R}_{n,A} : |A| > 1\}$ converge in $C(\mathbb{R}^{pq}, \mathbb{C})$ to independent zero mean complex Gaussian processes $\{R_A : |A| > 1\}$ having covariance and pseudo-covariance functions respectively given by $C_A(\boldsymbol{s}, \boldsymbol{t})$ and $\overline{C}_A(\boldsymbol{s}, \boldsymbol{t})$ in (2.3) and (2.4).*

The same multinomial formula (Ghoudi et al. (2001)) yields the representation

$$\hat{R}_{n,A}(\boldsymbol{t}) = \frac{1}{\sqrt{n}} \sum_{j=1}^{n} \prod_{k \in A} [\exp(\mathrm{i}\langle \boldsymbol{t}^{(k)}, \boldsymbol{e}_j^{(k)} \rangle) - \phi(\boldsymbol{t}^{(k)})]. \tag{2.8}$$

The Cramér-von Mises test statistic proposed is $nT_{n,b,A}$, where for a given subset A

$$T_{n,b,A} = \frac{1}{n} \int |\hat{R}_{n,A}(\boldsymbol{t})|^2 \varphi_b(\boldsymbol{t}) \mathrm{d}\boldsymbol{t}, \tag{2.9}$$

where φ_b is the $N_{pq}(\mathbf{0}, b^2\boldsymbol{I})$ density which acts as a weighting function. The multinomial representation and this appropriate weighting allow this test statistic to be computed explicitely as

$$\frac{1}{n^2}\sum_{l=1}^{n}\sum_{l'=1}^{n}\prod_{k\in A}\left\{ \exp\left[-\frac{b^2}{2}\|\boldsymbol{e}_l^{(k)}-\boldsymbol{e}_{l'}^{(k)}\|^2\right]\right. \tag{2.10}$$

$$-(b^2+1)^{-\frac{q}{2}} \exp\left[-\frac{1}{2}\frac{b^2}{b^2+1}\|\boldsymbol{e}_l^{(k)}\|^2\right] \tag{2.11}$$

$$\left.-(b^2+1)^{-\frac{q}{2}} \exp\left[-\frac{1}{2}\frac{b^2}{b^2+1}\|\boldsymbol{e}_{l'}^{(k)}\|^2\right]+(2b^2+1)^{-\frac{q}{2}}\right\}. \tag{2.12}$$

Since squared Mahalanobis type statistics are affine invariant it follows that $T_{n,b,A}$ is affine invariant. Thus, the asymptotic distribution of this statistic does not depend on unknown parameters.

It should be noted that the functional (2.9) defining this test statistic is not continuous; it is not even defined on $C(\mathbb{R}^{pq},\mathbb{C})$ but only on the subset of squared-integrable functions with respect to the measure $\varphi_b(\boldsymbol{t})d\boldsymbol{t}$. Thus, the continuity theorem as in Billingsley (1968) can not be invoked. In order to obtain the asymptotic distribution of this functional, the following generalization of Theorem 3.3 of Kellermeier (1980) on a uniform integrability condition is proposed. Let $\mathbb{R}_j^{pq}$ be the ball of radius j centered at zero in $\mathbb{R}^{pq}$.

Theorem 2.3. *Let $\boldsymbol{y}_n$ and $\boldsymbol{y}$ be random elements of $C(\mathbb{R}^{pq},\mathbb{C})^2$ such that $\boldsymbol{y}_n \xrightarrow{\mathcal{D}} \boldsymbol{y}$ on all compact balls. Let $f:\mathbb{C}^2\to\mathbb{R}$ be a continuous function and let G be a probability measure on $\mathbb{R}^{pq}$. Define $w_n=\int f(\boldsymbol{y}_n(\boldsymbol{t}))dG(\boldsymbol{t})$ and $w=\int f(\boldsymbol{y}(\boldsymbol{t}))dG(\boldsymbol{t})$. Suppose that w_n and w are well defined with probability one. Moreover, suppose that there exists $\alpha\geq 1$ such that for all $\epsilon>0$,*

$$\lim_{j\to\infty}\limsup_{n\to\infty}\int_{\mathbb{R}^{pq}\setminus\mathbb{R}_j^{pq}} E|f(\boldsymbol{y}_n(\boldsymbol{t}))|^{\alpha}dG(\boldsymbol{t})=0. \tag{2.13}$$

Then $w_n \xrightarrow{\mathcal{D}} w$ as $n\to\infty$.

Using Theorem 2.3, the joint convergence of the Cramér-von Mises functionals can be established.

Theorem 2.4.

$$\int\left(|\hat{R}_{n,A}(\boldsymbol{t})|^2,|\hat{R}_{n,B}(\boldsymbol{t})|^2\right)\varphi_b(\boldsymbol{t})d\boldsymbol{t}\xrightarrow{\mathcal{D}}\int\left(|R_A(\boldsymbol{t})|^2,|R_B(\boldsymbol{t})|^2\right)\varphi_b(\boldsymbol{t})d\boldsymbol{t},$$

where integrals are computed componentwise.

All possible subsets A can then be simultaneously accounted for by combining the test statistics as in

$$S_n=n\sum_{|A|>1}T_{n,b,A} \tag{2.14}$$

or

$$M_n = n \max_{|A|>1} T_{n,b,A}. \tag{2.15}$$

2.3. Relation to V-statistics

The statistic $T_{n,b,A}$ is in fact a V-statistic as in de Wet and Randles (1987). It can be represented as

$$T_{n,b,A} = \frac{1}{n^2} \sum_{l=1}^{n} \sum_{l'=1}^{n} h(\boldsymbol{\epsilon}_l, \boldsymbol{\epsilon}_{l'}; \hat{\boldsymbol{\lambda}}_n), \tag{2.16}$$

where $\hat{\boldsymbol{\lambda}}_n = (\bar{\boldsymbol{\epsilon}}, \boldsymbol{S})$ consistently estimates the true parameter $\boldsymbol{\lambda} = (\mathbf{0}, \boldsymbol{I})$. The function h at an arbitrary $\boldsymbol{\gamma} = (\boldsymbol{\mu}, \boldsymbol{\Sigma})$ is defined as

$$h(\boldsymbol{\epsilon}_l, \boldsymbol{\epsilon}_{l'}; \boldsymbol{\gamma}) = \int g(\boldsymbol{\epsilon}_l, \boldsymbol{t}; \boldsymbol{\gamma}) g(\boldsymbol{\epsilon}_{l'}, \boldsymbol{t}; \boldsymbol{\gamma}) \varphi_b(\boldsymbol{t}) \mathrm{d}\boldsymbol{t},$$

where, from elementary properties of integrals of odd functions, the function g can be taken real-valued

$$\begin{aligned} g(\boldsymbol{\epsilon}_l, \boldsymbol{t}; \boldsymbol{\gamma}) \;=\; & \prod_{k \in A} \Big[\cos(\langle \boldsymbol{t}^{(k)}, \boldsymbol{\Sigma}^{-1/2} (\boldsymbol{\epsilon}_l^{(k)} - \boldsymbol{\mu}) \rangle) \\ & + \sin(\langle \boldsymbol{t}^{(k)}, \boldsymbol{\Sigma}^{-1/2} (\boldsymbol{\epsilon}_l^{(k)} - \boldsymbol{\mu}) \rangle) - \phi(\boldsymbol{t}^{(k)}) \Big]. \end{aligned}$$

Letting $\mu(\boldsymbol{t}; \boldsymbol{\gamma}) = E_{(\mathbf{0}, \boldsymbol{I})} g(\boldsymbol{\epsilon}_l, \boldsymbol{t}; \boldsymbol{\gamma})$, it is seen that $T_{n,b,A}$ is a V-statistic which falls into case I situation. In de Wet and Randles (1987), they refer to case I when all first order partial derivatives of $\mu(\boldsymbol{t}; \boldsymbol{\gamma})$ evaluated at the true parameter $\boldsymbol{\gamma} = \boldsymbol{\lambda}$ vanish. Otherwise, they refer to case II. This is case I here since only A's such that $|A| > 1$ are considered. Thus, the asymptotic distribution of $T_{n,b,A}$ is the same whether one uses $\hat{\boldsymbol{\lambda}}_n$ or $\boldsymbol{\lambda}$ in (2.16). It is not clear, however, how this argument would apply to the joint distribution of $T_{n,b,A}$ and $T_{n,b,B}$. The proof of Theorem 2.4 in the Appendix does not use de Wet and Randles (1987).

For subsets A, $|A| = 1$, the statistic $T_{n,b,A}$ reduces to the statistic used by Baringhaus and Henze (1988) and Henze and Zirkler (1990) to test normality of a given marginal. They showed that the asymptotic distribution is affected by the estimation of the unknown parameters by establishing case II of de Wet and Randles (1987). Henze and Wagner (1997) treated the same problem with an approach based on stochastic processes.

2.4. Consistency

Consider the alternatives whereby $\boldsymbol{\epsilon}^{(1)}, \ldots, \boldsymbol{\epsilon}^{(p)}$ are distributed as $N_q(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, but are not independent. Then, $S_n \to \infty$ and $M_n \to \infty$. Thus, the test statistics S_n and M_n in (2.14)-(2.15) are consistent against any such alternatives.

The argument to establish consistency is rather trivial as in Baringhaus and Henze (1988). Recall that $C_p(\cdot)$ is the joint characteristic function of $\boldsymbol{\epsilon}^{(1)}, \ldots, \boldsymbol{\epsilon}^{(p)}$. This

argument consists of the following almost sure convergence:

$$\begin{aligned} T_{n,b,A} &= \int |\sum_{B\subset A}(-1)^{|A\setminus B|}\exp(-\mathrm{i}\sum_{k\in B}\langle \boldsymbol{t}^{(k)}, \boldsymbol{S}^{-\frac{1}{2}}\overline{\boldsymbol{\epsilon}}\rangle) \\ &\quad \times\frac{1}{n}\sum_{j=1}^{n}\exp(\mathrm{i}\sum_{k\in B}\langle \boldsymbol{t}^{(k)}, \boldsymbol{S}^{-\frac{1}{2}}\boldsymbol{\epsilon}_j^{(k)}\rangle)\prod_{i\in A\setminus B}\phi(\boldsymbol{t}^{(i)})|^2\varphi_b(\boldsymbol{t})\mathrm{d}\boldsymbol{t} \\ &\overset{as}{\to} \int |\sum_{B\subset A}(-1)^{|A\setminus B|}\exp(-\mathrm{i}\sum_{k\in B}\langle \boldsymbol{t}^{(k)}, \boldsymbol{\Sigma}^{-\frac{1}{2}}\boldsymbol{\mu}\rangle) \\ &\quad \times C_p((\boldsymbol{I}_p\otimes\boldsymbol{\Sigma}^{-\frac{1}{2}})\boldsymbol{t}^B)\prod_{i\in A\setminus B}\phi(\boldsymbol{t}^{(i)})|^2\varphi_b(\boldsymbol{t})\mathrm{d}\boldsymbol{t} \end{aligned}$$

which equals 0 for all A, $|A|>1$, if and only if $\boldsymbol{\epsilon}^{(1)},\ldots,\boldsymbol{\epsilon}^{(p)}$ are independent $N_q(\boldsymbol{\mu},\boldsymbol{\Sigma})$. Therefore, if $\boldsymbol{\epsilon}^{(1)},\ldots,\boldsymbol{\epsilon}^{(p)}$ are dependent $N_q(\boldsymbol{\mu},\boldsymbol{\Sigma})$ then, there exists an A such that $nT_{n,b,A}\to\infty$ which suffices to have $S_n\to\infty$ and $M_n\to\infty$.

3. TESTING INDEPENDENCE: THE SERIAL SITUATION

3.1. The case of known parameters

Let $\boldsymbol{u}_1,\boldsymbol{u}_2,\ldots$ be a stationary sequence of random vectors $\boldsymbol{u}_i$ distributed as $N_q(\boldsymbol{0},\boldsymbol{I})$. It is desired to verify whether the $\boldsymbol{u}_i$'s are independent. Introduce the partitioned random vectors $\boldsymbol{\epsilon}_i=(\boldsymbol{u}_i,\ldots,\boldsymbol{u}_{i+p-1})\in\mathbb{R}^{pq}$, $i=1,\ldots,n-p+1$. Also, let $R_{n,A}(\boldsymbol{t})$ be as in (2.1) with the slight modification $\phi_{n,p}(\boldsymbol{t})=\frac{1}{n}\sum_{j=1}^{n-p+1}\exp(\mathrm{i}\langle\boldsymbol{t},\boldsymbol{\epsilon}_j\rangle)$. The main result related to the asymptotic distribution is that the m-dependence introduced by the overlapping of the $\boldsymbol{u}_i$'s does not affect the asymptotic distribution. It is the same as in the non-serial case.

Theorem 3.1. *If the $\boldsymbol{u}_i$'s are independent, the collection of processes $\{R_{n,A}:|A|>1\}$ converge in $C(\mathbb{R}^{pq},\mathbb{C})$ to independent zero mean complex Gaussian processes $\{R_A:|A|>1\}$ having covariance and pseudo-covariance functions respectively given by $C_A(\boldsymbol{s},\boldsymbol{t})$ and $\overline{C}_A(\boldsymbol{s},\boldsymbol{t})$ in (2.3) and (2.4).*

As in (2.5), the multinomial formula (Ghoudi et al. (2001)) yields

$$R_{n,A}(\boldsymbol{t})=\frac{1}{\sqrt{n}}\sum_{j=1}^{n-p+1}\prod_{k\in A}[\exp(\mathrm{i}\langle\boldsymbol{t}^{(k)},\boldsymbol{\epsilon}_j^{(k)}\rangle)-\phi(\boldsymbol{t}^{(k)})]. \tag{3.1}$$

which is useful in the proof of Theorem 3.1.

3.2. The case of unknown parameters

The context is the same as in the preceeding section but here the $\boldsymbol{u}_i$'s all have the same $N_q(\boldsymbol{\mu},\boldsymbol{\Sigma})$ normal distribution, where $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$, positive definite, are assumed unknown. Again, we want to test whether the $\boldsymbol{u}_i$'s are independent. To this aim, define the random vectors $\boldsymbol{\epsilon}_i=(\boldsymbol{u}_i,\ldots,\boldsymbol{u}_{i+p-1})\in\mathbb{R}^{pq}$ and $\boldsymbol{e}_i=(\hat{\boldsymbol{u}}_i,\ldots,\hat{\boldsymbol{u}}_{i+p-1})\in\mathbb{R}^{pq}$, $i=1,\ldots,n-p+1$. Also, define the standardized residuals $\hat{\boldsymbol{u}}_i=\boldsymbol{S}^{-\frac{1}{2}}(\boldsymbol{u}_i-\overline{\boldsymbol{u}})$ with

the sample covariance matrix $\boldsymbol{S} = \frac{1}{n}\sum_{j=1}^{n}(\boldsymbol{u}_j - \overline{\boldsymbol{u}})(\boldsymbol{u}_j - \overline{\boldsymbol{u}})^{\mathsf{T}}$ and the sample mean $\overline{\boldsymbol{u}} = \frac{1}{n}\sum_{j=1}^{n}\boldsymbol{u}_j$. Now, let

$$\hat{R}_{n,A}(\boldsymbol{t}) = \sqrt{n}\sum_{B\subset A}(-1)^{|A\setminus B|}\hat{\phi}_{n,p}(\boldsymbol{t}^B)\prod_{i\in A\setminus B}\phi(\boldsymbol{t}^{(i)}), \tag{3.2}$$

where $\hat{\phi}_{n,p}(\boldsymbol{t}) = \frac{1}{n}\sum_{j=1}^{n-p+1}\exp(\mathrm{i}\langle \boldsymbol{t}, \boldsymbol{e}_j\rangle)$. The asymptotic behaviour of these processes is stated next. The main conclusion is that the estimation of the unknown parameters does not affect the asymptotic distribution.

Theorem 3.2. *If the $\boldsymbol{u}_i$'s are independent, the processes $\{\hat{R}_{n,A} : |A| > 1\}$ converge in $C(\mathbb{R}^{pq}, \mathbb{C})$ to independent zero mean complex Gaussian processes $\{R_A : |A| > 1\}$ having covariance and pseudo-covariance functions respectively given by $C_A(\boldsymbol{s}, \boldsymbol{t})$ and $\overline{C}_A(\boldsymbol{s}, \boldsymbol{t})$ in (2.3) and (2.4).*

Note that the multinomial formula yields

$$\hat{R}_{n,A}(\boldsymbol{t}) = \frac{1}{\sqrt{n}}\sum_{j=1}^{n-p+1}\prod_{k\in A}[\exp(\mathrm{i}\langle \boldsymbol{t}^{(k)}, \boldsymbol{e}_j^{(k)}\rangle) - \phi(\boldsymbol{t}^{(k)})] \tag{3.3}$$

and so the Cramér-von Mises test statistic

$$T_{n,b,A} = \frac{1}{n}\int |\hat{R}_{n,A}(\boldsymbol{t})|^2\varphi_b(\boldsymbol{t})\mathrm{d}\boldsymbol{t}$$

can be computed as

$$\frac{1}{n^2}\sum_{l=1}^{n-p+1}\sum_{l'=1}^{n-p+1}\prod_{k\in A}\Bigg\{ \exp\left[-\frac{b^2}{2}\|\boldsymbol{e}_l^{(k)} - \boldsymbol{e}_{l'}^{(k)}\|^2\right] \tag{3.4}$$

$$-(b^2+1)^{-\frac{q}{2}}\exp\left[-\frac{1}{2}\frac{b^2}{b^2+1}\|\boldsymbol{e}_l^{(k)}\|^2\right] \tag{3.5}$$

$$-(b^2+1)^{-\frac{q}{2}}\exp\left[-\frac{1}{2}\frac{b^2}{b^2+1}\|\boldsymbol{e}_{l'}^{(k)}\|^2\right] + (2b^2+1)^{-\frac{q}{2}}\Bigg\}. \tag{3.6}$$

This representation shows that $T_{n,b,A}$ is affine invariant. Here again we can use theorem 2.3 to obtain

Theorem 3.3.

$$\int \left(|\hat{R}_{n,A}(\boldsymbol{t})|^2, |\hat{R}_{n,B}(\boldsymbol{t})|^2\right)\varphi_b(\boldsymbol{t})d\boldsymbol{t} \xrightarrow{\mathcal{D}} \int \left(|R_A(\boldsymbol{t})|^2, |R_B(\boldsymbol{t})|^2\right)\varphi_b(\boldsymbol{t})d\boldsymbol{t}, \tag{3.7}$$

where integrals are computed componentwise.

In the serial situation, a subset A and its translate $A+k$ essentially lead to the same statistic $T_{n,b,A}$. Hence, when considering these statistics, only A's such that $1 \in A$ can be considered. The same statistics (2.14) or (2.15) can be used to perform the statistical test:

$$S_n = n\sum_{|A|>1, 1\in A} T_{n,b,A}, \qquad M_n = n\max_{|A|>1, 1\in A} T_{n,b,A}.$$

4. PROPERTIES OF THE LIMITING PROCESSES

This section shows how to compute the critical values of the Cramér-von Mises variable $T_{b,A} \equiv \int |R_A(\boldsymbol{t})|^2 \varphi_b(\boldsymbol{t}) \mathrm{d}\boldsymbol{t}$. This can be achieved either by computing its cumulants and then applying the Cornish-Fisher asymptotic expansion (see Lee and Lin (1992); Lee and Lee (1992)) or by inversion of the characteristic function (see Imhof (1961) or an improved version of this algorithm introduced by Davies (1973, 1980) or Deheuvels and Martynov (1996)) after evaluation of the eigenvalues of C_A.

The Cramér-von Mises test statistic in (2.9)can also be written in terms of a real process:

$$T_{n,b,A} = \frac{1}{n} \int W_{n,A}^2(\boldsymbol{t}) \varphi_b(\boldsymbol{t}) \mathrm{d}\boldsymbol{t}$$

where $W_{n,A}(\boldsymbol{t}) = \frac{1}{\sqrt{n}} \sum_{j=1}^{n} \prod_{k \in A} \left[\cos(\langle \boldsymbol{t}^{(k)}, \boldsymbol{e}_j^{(k)} \rangle) + \sin(\langle \boldsymbol{t}^{(k)}, \boldsymbol{e}_j^{(k)} \rangle) - \phi(\boldsymbol{t}^{(k)}) \right]$ is a real process which converges to a real Gaussian process with the same covariance function $C_A(\boldsymbol{s}, \boldsymbol{t})$ as in (2.3). Thus the usual Karhunen-Loève expansion holds.

Let $k = |A|$. It is well known that

$$\int_{\mathbb{R}^{pq}} |R_A(\boldsymbol{t})|^2 \varphi_b(\boldsymbol{t}) \mathrm{d}\boldsymbol{t} \sim T_{b,A} = \sum_{(i_1,\ldots,i_k) \in \mathbb{N}^{*k}} \lambda_{(i_1,\ldots,i_k)} |Z_{(i_1,\ldots,i_k)}|^2 \tag{4.1}$$

where $Z_{(i_1,\ldots,i_k)}$ are standard i.i.d. $CN(0,1)$ complex random variables. Also, it is easy to show that $\lambda_{(i_1,\ldots,i_k)} = \prod_{l=1}^{k} \lambda_{i_l}$ where the λ_j's are the eigenvalues of the integral operator O defined by

$$O(f)(\boldsymbol{s}) = \int_{\mathbb{R}^q} f(\boldsymbol{t}) K(\boldsymbol{s}, \boldsymbol{t}) \varphi_b(\boldsymbol{t}) \mathrm{d}\boldsymbol{t} \tag{4.2}$$

with

$$K(\boldsymbol{s}, \boldsymbol{t}) = \exp(-\frac{1}{2} \|\boldsymbol{s} - \boldsymbol{t}\|^2) - \exp(-\frac{1}{2}(\|\boldsymbol{s}\|^2 + \|\boldsymbol{t}\|^2)). \tag{4.3}$$

That is to say the problem is to solve, in λ (and f), the linear second order homogeneous Fredholm integral equation

$$\lambda f(\boldsymbol{s}) = \int_{\mathbb{R}^q} f(\boldsymbol{t}) K(\boldsymbol{s}, \boldsymbol{t}) \varphi_b(\boldsymbol{t}) \mathrm{d}\boldsymbol{t}. \tag{4.4}$$

See Conway (1985) for an introduction to integral operators.

It does not seem possible to solve (4.4) explicitely, but one can compute its eigenvalues using a relation as

$$\int_{\mathbb{R}^q} f(\boldsymbol{\nu}) e^{-\|\boldsymbol{\nu}\|^2} \mathrm{d}\boldsymbol{\nu} = \sum_{j=1}^{N} B_j f(\boldsymbol{\nu}_j) \tag{4.5}$$

where the parameters B_j and $\boldsymbol{\nu}_j = (\nu_{j,1}, \ldots, \nu_{j,q})$, $j = 1, \ldots, N$ could be respectively the coefficients and the points of a cubature formula (CF), or also could be obtained by a Monte-Carlo experiment, in which case $B_j = \frac{1}{N}$ and $\boldsymbol{\nu}_j \sim N(\boldsymbol{0}, \boldsymbol{I})$, $j = 1, \ldots, N$. A good rule of the thumb is to use a cubature formula when b is small, for example less than one, otherwise use Monte Carlo method.

We used the following formulas: the nth degree Gauss quadrature formula when $q = 1$, the fifteenth degree CF $E_2^{r^2} : 15 - 1$ (see (Stroud, 1971, p. 326)) when $q = 2$

and the seventh degree CF $E_q^{r^2} : 7-2$ appearing in (Stroud, 1971, p. 319) for $q \geq 3$. This last formula contains an error, see Stroud (1967) for details.

Moreover one can notice that all the cumulants of $T_{b,A}$ can be computed explicitly. In fact, the m-th cumulant $\kappa_{m,A}(b)$ of $T_{b,A}$ in (4.1) is given by

$$\kappa_{m,A}(b) = (m-1)! \left[\int_{\mathbb{R}^q} K^{(m)}(\boldsymbol{x}, \boldsymbol{x}) \varphi_b(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} \right]^{|A|} \tag{4.6}$$

where

$$\varphi_b(\boldsymbol{x}) = (2\pi)^{-q/2} b^{-q} \exp(-\frac{1}{2}\|\boldsymbol{x}\|^2/b^2) \tag{4.7}$$

and where $K^{(1)}(\boldsymbol{x}, \boldsymbol{y}) = K(\boldsymbol{x}, \boldsymbol{y})$ and

$$K^{(m)}(\boldsymbol{x}, \boldsymbol{y}) = \int_{\mathbb{R}^q} K^{(m-1)}(\boldsymbol{x}, \boldsymbol{z}) K(\boldsymbol{z}, \boldsymbol{y}) \varphi_b(\boldsymbol{z}) \mathrm{d}\boldsymbol{z},\ m \geq 2.$$

Define

$$\begin{aligned} I^{(1)}_{\boldsymbol{x},\boldsymbol{y}}(\alpha, \beta, \gamma) &= \exp(\alpha\|\boldsymbol{x}\|^2 + \beta\|\boldsymbol{y}\|^2 + \gamma\langle \boldsymbol{x}, \boldsymbol{y}\rangle) \\ I^{(m)}_{\boldsymbol{x},\boldsymbol{y}}(\alpha, \beta, \gamma) &= \int_{\mathbb{R}^q} I^{(m-1)}_{\boldsymbol{x},\boldsymbol{z}}(\alpha, \beta, \gamma) K(\boldsymbol{z}, \boldsymbol{y}) \varphi_b(\boldsymbol{z}) \mathrm{d}\boldsymbol{z},\ m \geq 2. \end{aligned}$$

One can show

$$K^{(m)}(\boldsymbol{x}, \boldsymbol{x}) = I^{(m)}_{\boldsymbol{x},\boldsymbol{x}}\left(-\frac{1}{2}, -\frac{1}{2}, 1\right) - I^{(m)}_{\boldsymbol{x},\boldsymbol{x}}\left(-\frac{1}{2}, -\frac{1}{2}, 0\right) \tag{4.8}$$

and the recurrence relation

$$I^{(m)}_{\boldsymbol{x},\boldsymbol{x}}(\alpha, \beta, \gamma) = \chi_\beta \left[I^{(m-1)}_{\boldsymbol{x},\boldsymbol{x}}(\alpha - \frac{\gamma^2}{4c}, -\frac{1}{4c} - \frac{1}{2}, -\frac{2\gamma}{4c}) - I^{(m-1)}_{\boldsymbol{x},\boldsymbol{x}}(\alpha - \frac{\gamma^2}{4c}, -\frac{1}{2}, 0) \right],$$

where $\chi_\beta = (2\pi)^{-q/2} b^{-q} \int_{\mathbb{R}^q} e^{c\|\boldsymbol{z}\|^2} \mathrm{d}\boldsymbol{z} = (1 + b^2 - 2b^2\beta)^{-q/2}$ and $-\frac{1}{4c} = \frac{b^2}{2+2b^2-4b^2\beta}$. Thus, one can express $K^{(m)}(\boldsymbol{x}, \boldsymbol{x})$ in terms of $I^{(1)}_{\boldsymbol{x},\boldsymbol{x}}$ and then use the relation

$$\int_{\mathbb{R}^q} I^{(1)}_{\boldsymbol{x},\boldsymbol{x}}(\alpha, \beta, \gamma) \varphi_b(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} = [1 - 2b^2(\alpha + \beta + \gamma)]^{-q/2} \tag{4.9}$$

to obtain all the cumulants recursively. Note that this permits to double-check into the preceding computation of the eigenvalues through the following relation

$$\kappa_{m,A}(b) = (m-1)! \left[\sum_{j=1}^{\infty} \lambda_j^m \right]^{|A|}. \tag{4.10}$$

Note also that one only needs to compute the cumulants $\kappa_m(b)$'s for the case $|A| = 1$ since

$$\kappa_{m,A}(b) = [\kappa_m(b)]^{|A|} [(m-1)!]^{1-|A|}. \tag{4.11}$$

The CF's used here are not the only one available to obtain estimates $\hat{\lambda}_j$ of the λ_j's. A good choice is one that minimises

$$|\kappa_m(b) - (m-1)! \sum_{j=1}^{\infty} \hat{\lambda}_j^m|. \tag{4.12}$$

See Cools (1999) and Cools and Rabinowitz (1993) for a comprehensive list of such formulas.
Table 4.1 provides an approximation of the cut-off values obtained from the Cornish Fisher asymptotic expansion based on the first six cumulants, for $b = 0.1$.

TABLE 4.1. Critical Values of the Distribution of $T_{b,A}$ for $b = 0.1$.

$1-\alpha$	q=2			q=3		
	k=2	k=3	k=4	k=2	k=3	k=4
0.900	0.000733	1.230e-05	2.122e-07	0.00137	3.347e-05	8.707e-07
0.905	0.000745	1.244e-05	2.140e-07	0.00138	3.370e-05	8.741e-07
0.910	0.000758	1.260e-05	2.159e-07	0.00140	3.393e-05	8.777e-07
0.915	0.000771	1.276e-05	2.179e-07	0.00142	3.418e-05	8.815e-07
0.920	0.000785	1.293e-05	2.200e-07	0.00143	3.444e-05	8.854e-07
0.925	0.000800	1.311e-05	2.222e-07	0.00145	3.472e-05	8.896e-07
0.930	0.000815	1.330e-05	2.246e-07	0.00147	3.501e-05	8.939e-07
0.935	0.000832	1.350e-05	2.271e-07	0.00149	3.531e-05	8.986e-07
0.940	0.000850	1.372e-05	2.298e-07	0.00152	3.564e-05	9.035e-07
0.945	0.000870	1.395e-05	2.326e-07	0.00154	3.600e-05	9.088e-07
0.950	0.000891	1.421e-05	2.357e-07	0.00157	3.638e-05	9.145e-07
0.955	0.000915	1.449e-05	2.392e-07	0.00160	3.680e-05	9.207e-07
0.960	0.000941	1.480e-05	2.429e-07	0.00163	3.726e-05	9.275e-07
0.965	0.000971	1.514e-05	2.471e-07	0.00167	3.777e-05	9.351e-07
0.970	0.001005	1.554e-05	2.519e-07	0.00171	3.836e-05	9.437e-07
0.975	0.001045	1.601e-05	2.575e-07	0.00176	3.904e-05	9.537e-07
0.980	0.001093	1.657e-05	2.643e-07	0.00182	3.985e-05	9.656e-07
0.985	0.001156	1.729e-05	2.728e-07	0.00190	4.088e-05	9.805e-07
0.990	0.001243	1.828e-05	2.845e-07	0.00200	4.229e-05	1.000e-06
0.995	0.001390	1.994e-05	3.039e-07	0.00217	4.460e-05	1.033e-06

TABLE 4.2. Distribution of $T^*_{b,A}$ for $b = 0.1$.

x	q=2			q=3		
	$P[T_{b,2} \leq x]$	$P[T_{b,3} \leq x]$	$P[T_{b,4} \leq x]$	$P[T_{b,2} \leq x]$	$P[T_{b,3} \leq x]$	$P[T_{b,4} \leq x]$
0.0	0.593	0.566	0.546	0.562	0.536	0.520
0.2	0.665	0.640	0.622	0.637	0.612	0.598
0.4	0.725	0.705	0.691	0.702	0.683	0.671
0.6	0.777	0.761	0.751	0.759	0.745	0.737
0.8	0.819	0.809	0.802	0.807	0.798	0.794
1.0	0.854	0.848	0.845	0.848	0.843	0.842
1.2	0.883	0.881	0.880	0.880	0.880	0.881
1.4	0.906	0.907	0.908	0.907	0.910	0.913
1.6	0.925	0.928	0.931	0.928	0.933	0.937
1.8	0.941	0.944	0.948	0.945	0.951	0.955
2.0	0.953	0.957	0.961	0.958	0.964	0.969
2.2	0.963	0.967	0.972	0.968	0.974	0.979
2.4	0.970	0.975	0.979	0.976	0.982	0.986
2.6	0.977	0.981	0.985	0.982	0.987	0.990
2.8	0.982	0.986	0.989	0.986	0.991	0.994
3.0	0.985	0.989	0.992	0.990	0.994	0.996
3.2	0.988	0.992	0.994	0.992	0.996	0.997
3.4	0.991	0.994	0.996	0.994	0.997	0.998
3.6	0.993	0.995	0.997	0.996	0.998	0.999
3.8	0.994	0.996	0.998	0.997	0.998	0.999
4.0	0.995	0.997	0.998	0.997	0.999	0.999

As in Ghoudi et al. (2001), define $T^*_{b,A}$ to be the standardised version of $T_{b,A}$. Then Table 4.2 provides the distribution function of this statistic for some values of $|A|$ and q, with $b = 0.1$ as approximated by Davies technique. Besides, a C++ program is available from the authors which permits to compute any cut-off value given the nominal level and vice-versa.

In Table 4.3, one can find the empirical percentage points of $nT_{n,b,A}$ $(n = 20, 50, 100; b = 0.1, 0.5, 1.0, 3.0; \alpha = 0.1, 0.05)$ based on $N = 10000$ Monte Carlo replications, in the non serial case.

TABLE 4.3. Empirical Percentage Points of $nT_{n,b,A}$ based on $N = 10000$ Monte Carlo replications: non-serial case.

$1-\alpha$	b	n	q=2			q=3		
			$k=2$	$k=3$	$k=4$	$k=2$	$k=3$	$k=4$
0.9	0.1	20	0.000718	1.249e-05	2.371e-07	0.00134	3.460e-05	1.033e-06
		50	0.000718	1.253e-05	2.358e-07	0.00136	3.420e-05	9.748e-07
		100	0.000733	1.229e-05	2.242e-07	0.00137	3.389e-05	9.430e-07
	0.5	20	0.170	0.049	0.016	0.269	0.110	0.050
		50	0.167	0.0481	0.0156	0.266	0.107	0.0484
		100	0.168	0.0471	0.0149	0.266	0.106	0.0472
	1.0	20	0.561	0.333	0.221	0.727	0.555	0.450
		50	0.558	0.327	0.213	0.724	0.547	0.440
		100	0.559	0.326	0.209	0.723	0.544	0.436
	3.0	20	0.938	0.864	0.820	0.981	0.969	0.958
		50	0.940	0.860	0.814	0.983	0.967	0.955
		100	0.938	0.858	0.811	0.983	0.966	0.954
0.95	0.1	20	0.000854	1.485e-05	3.036e-07	0.00151	3.849e-05	1.212e-06
		50	0.000883	1.465e-05	2.840e-07	0.00156	3.751e-05	1.103e-06
		100	0.000874	1.429e-05	2.653e-07	0.00155	3.724e-05	1.027e-06
	0.5	20	0.191	0.0539	0.0184	0.289	0.114	0.053
		50	0.192	0.0520	0.0168	0.289	0.111	0.0505
		100	0.191	0.0513	0.0158	0.289	0.110	0.0486
	1.0	20	0.606	0.345	0.228	0.749	0.562	0.456
		50	0.603	0.338	0.218	0.748	0.553	0.445
		100	0.593	0.336	0.213	0.745	0.549	0.439
	3.0	20	0.955	0.868	0.823	0.984	0.970	0.959
		50	0.954	0.863	0.816	0.987	0.967	0.956
		100	0.953	0.861	0.813	0.986	0.966	0.955

In Table 4.4, one can find the empirical percentage points of $nT_{n,b,A}$ $(n = 20, 50, 100; b = 0.1, 0.5, 1.0, 3.0; \alpha = 0.1, 0.05)$ based on $N = 10000$ Monte Carlo replications, in the serial case, for $p = 4$.

TABLE 4.4. Empirical Percentage Points of $nT_{n,b,A}$ based on $N = 10000$ Monte Carlo replications for $p = 4$: serial case.

$1-\alpha$	b	n	q=2			q=3		
			$k=2$	$k=3$	$k=4$	$k=2$	$k=3$	$k=4$
0.9	0.1	20	0.000712	1.174e-05	2.139e-07	0.00132	3.384e-05	9.931e-07
		50	0.000721	1.205e-05	2.288e-07	0.00135	3.357e-05	9.867e-07
		100	0.000716	1.217e-05	2.261e-07	0.00138	3.335e-05	9.530e-07
	0.5	20	0.173	0.0504	0.0171	0.276	0.1160	0.0547
		50	0.168	0.0484	0.0162	0.269	0.109	0.050
		100	0.170	0.0476	0.0155	0.268	0.107	0.048
	1.0	20	0.567	0.353	0.241	0.737	0.586	0.486
		50	0.561	0.334	0.223	0.728	0.559	0.458
		100	0.561	0.328	0.216	0.727	0.55	0.447
	3.0	20	0.937	0.880	0.842	0.984	0.975	0.966
		50	0.939	0.867	0.825	0.983	0.970	0.960
		100	0.939	0.862	0.818	0.983	0.968	0.957
0.95	0.1	20	0.000841	1.414e-05	2.734e-07	0.00148	3.804e-05	1.176e-06
		50	0.000897	1.412e-05	2.780e-07	0.00154	3.772e-05	1.128e-06
		100	0.000861	1.435e-05	2.737e-07	0.00154	3.688e-05	1.071e-06
	0.5	20	0.196	0.0556	0.0193	0.296	0.122	0.0587
		50	0.192	0.0528	0.0179	0.292	0.114	0.0539
		100	0.189	0.0517	0.0168	0.290	0.111	0.0509
	1.0	20	0.615	0.367	0.255	0.759	0.598	0.498
		50	0.604	0.346	0.231	0.753	0.567	0.466
		100	0.602	0.340	0.222	0.751	0.557	0.452
	3.0	20	0.955	0.886	0.849	0.985	0.976	0.968
		50	0.953	0.871	0.829	0.987	0.971	0.961
		100	0.952	0.865	0.822	0.987	0.969	0.958

5. ONE-WAY MANOVA MODEL WITH RANDOM EFFECTS

The one way linear model with random effects

$$\boldsymbol{\epsilon}_i^{(j)} = \boldsymbol{\mu} + \boldsymbol{\alpha}_i + \boldsymbol{\delta}_i^{(j)},\ i = 1, \ldots, n; j = 1, \ldots, p,$$

where $\boldsymbol{\alpha}_i \sim N_q(\mathbf{0}, \boldsymbol{\psi})$ and $\boldsymbol{\delta}_i^{(j)} \sim N_q(\mathbf{0}, \boldsymbol{\Sigma})$ are all mutually independant, provides a joint normal model for the non-serial case. This means that in this variance component model

$$\boldsymbol{\epsilon}_i = (\boldsymbol{\epsilon}_i^{(1)}, \ldots, \boldsymbol{\epsilon}_i^{(p)}) \sim N_{pq}(\mathbf{1}_p \otimes \boldsymbol{\mu}, (\boldsymbol{I}_p \otimes \boldsymbol{\Sigma}) + (\mathbf{1}_p \mathbf{1}_p^{\mathsf{T}}) \otimes \boldsymbol{\psi}), i = 1, \ldots, n,$$

are i.i.d. The test of independance amounts to the parametric test of the hypothesis $H_0 : \boldsymbol{\psi} = \mathbf{0}$. The MANOVA decomposition

$$\sum_{i=1}^{n}\sum_{j=1}^{p}(\boldsymbol{\epsilon}_i^{(j)} - \overline{\boldsymbol{\epsilon}}_{\bullet}^{(\bullet)})(\boldsymbol{\epsilon}_i^{(j)} - \overline{\boldsymbol{\epsilon}}_{\bullet}^{(\bullet)})^{\mathsf{T}} = \sum_{i=1}^{n}\sum_{j=1}^{p}(\boldsymbol{\epsilon}_i^{(j)} - \overline{\boldsymbol{\epsilon}}_i^{(\bullet)})(\boldsymbol{\epsilon}_i^{(j)} - \overline{\boldsymbol{\epsilon}}_i^{(\bullet)})^{\mathsf{T}} \quad (5.1)$$

$$+ \ p\sum_{i=1}^{n}(\overline{\boldsymbol{\epsilon}}_i^{(\bullet)} - \overline{\boldsymbol{\epsilon}}_{\bullet}^{(\bullet)})(\overline{\boldsymbol{\epsilon}}_i^{(\bullet)} - \overline{\boldsymbol{\epsilon}}_{\bullet}^{(\bullet)})^{\mathsf{T}} \quad (5.2)$$

$$= E + H, \quad (5.3)$$

(see Rencher (2002)), leads to the usual MANOVA table. A dot means averaging over the corresponding index. The test of $H_0 : \boldsymbol{\psi} = \mathbf{0}$ is usually done with Wilks statistic

TABLE 5.1. MANOVA table

Source	Sum of squares	Degrees of freedom	Expected sum of squares	Null distribution
Between	H	$\nu_H = n - 1$	$(n-1)\boldsymbol{\Sigma} + p(n-1)\boldsymbol{\psi}$	$W_q(n-1, \boldsymbol{\Sigma})$
Within	E	$\nu_E = n(p-1)$	$n(p-1)\boldsymbol{\Sigma}$	$W_q(n(p-1), \boldsymbol{\Sigma})$
Total	$E + H$			

$$\Lambda = \frac{\det E}{\det (E+H)} = \prod_{k=1}^{q} \left[1 + \lambda_k(E^{-1}H)\right]^{-1},$$

where $\lambda_k(E^{-1}H)$ are the eigenvalues of $E^{-1}H$. The null distribution of Λ is the Λ_{q,ν_H,ν_E} distribution. Tables of exact critical points for Λ are available but for $q = 2$ the relation

$$\frac{(\nu_E - 1)}{\nu_H} \frac{(1 - \Lambda^{\frac{1}{2}})}{\Lambda^{\frac{1}{2}}} \sim F_{2\nu_H, 2(\nu_E - 1)}$$

holds.

TABLE 5.2. Empirical power of $nT_{n,b,A}$ and Wilks test based on $N = 10000$ Monte Carlo replications for $p = 2$, $q = 2$, $\boldsymbol{\mu} = \mathbf{0}$, $\boldsymbol{\Sigma} = \gamma \boldsymbol{I}_2$ and $\boldsymbol{\Psi} = \theta \boldsymbol{I}_2$.

$1-\alpha$	$\frac{\theta}{\gamma}$	n	$nT_{n,b,A}$					Wilks	$\frac{\lvert\hat{R}_{n,A}(\boldsymbol{t})\rvert^2}{0.5 C_A(\boldsymbol{t},\boldsymbol{t})}$
			$b = 0.01$	$b = 0.1$	$b = 0.5$	$b = 1.0$	$b = 3.0$		$\boldsymbol{t} = (0.6, 0.6, 0.6, 0.6)$
0.90	0.0	20	10.14	10.27	10.06	10.21	10.33	9.96	10.64
		50	9.46	10.14	10.15	10.20	9.94	10.34	9.99
	0.2	20	14.27	14.39	13.78	12.64	11.12	39.15	22.23
		50	28.61	29.87	27.50	19.00	10.50	63.63	29.44
	0.4	20	28.43	28.57	26.61	19.28	11.81	67.69	35.55
		50	68.31	69.70	63.43	40.63	13.67	94.72	49.78
	1.0	20	77.09	77.36	72.66	50.48	16.49	97.86	51.98
		50	99.80	99.81	99.43	92.68	27.66	99.99	60.26
0.95	0.0	20	4.67	4.91	5.52	4.88	5.16	5.22	5.39
		50	4.84	4.58	4.98	4.82	5.04	5.22	5.16
	0.2	20	7.35	7.63	7.94	6.39	5.53	26.10	13.86
		50	18.96	18.26	17.11	10.60	5.14	49.60	20.06
	0.4	20	16.77	17.20	16.93	10.80	6.03	53.44	25.04
		50	56.83	55.72	50.23	28.08	7.37	89.24	38.59
	1.0	20	64.57	65.61	61.12	36.17	8.98	94.91	42.47
		50	99.45	99.42	98.61	86.81	17.72	99.98	50.93

Wilks test has power superior to the test proposed in this paper. This is not surprising since Wilks test is specifically designed for the parametric hypothesis $\boldsymbol{\Psi} = \mathbf{0}$ in the linear model which holds in the simulation. However, our test is more generally applicable and will yield reasonable power for reasonably large sample sizes. Moreover, it

is not difficult to construct another model where Wilks test would fail. For example, Wilks test would be unable to detect the dependance in the normal mixture model

$$\begin{pmatrix} \boldsymbol{\epsilon}^{(1)} \\ \boldsymbol{\epsilon}^{(2)} \end{pmatrix} \sim 0.5N_4\left(\begin{pmatrix} \mathbf{0} \\ \mathbf{0} \end{pmatrix}, \begin{pmatrix} \boldsymbol{I}_2 & \boldsymbol{\Psi} \\ \boldsymbol{\Psi} & \boldsymbol{I}_2 \end{pmatrix}\right) + 0.5N_4\left(\begin{pmatrix} \mathbf{0} \\ \mathbf{0} \end{pmatrix}, \begin{pmatrix} \boldsymbol{I}_2 & -\boldsymbol{\Psi} \\ -\boldsymbol{\Psi} & \boldsymbol{I}_2 \end{pmatrix}\right)$$

contrary to our test.

6. PROOFS

Define the metric ρ on $C := C(\mathbb{R}^{pq}, \mathbb{C})$ by

$$\rho(x, y) = \sum_{j=1}^{\infty} 2^{-j} \frac{\rho_j(x, y)}{1 + \rho_j(x, y)}, \tag{6.1}$$

where

$$\rho_j(x, y) = \sup_{\|\boldsymbol{t}\| \le j} |x(\boldsymbol{t}) - y(\boldsymbol{t})|$$

is the usual sup norm. Endowed with this metric, $C(\mathbb{R}^p, \mathbb{C})$ is a separable Fréchet space, that is a linear complete and separable metric space. Moreover, convergence in this metric corresponds to the uniform convergence on all compact sets, $\lim_{n\to\infty} \rho(x_n, x) = 0$ if and only if $\forall j \ge 1, \lim_{n\to\infty} \rho_j(x_n, x) = 0$. For random elements, it also holds $\rho(X_n, Y_n) \xrightarrow{P} 0$ if and only if $\forall j \ge 1, \rho_j(X_n, Y_n) \xrightarrow{P} 0$. For all mappings $x \in C$, let $r_j(x)$ be the restriction of x to the ball $\mathbb{R}_j^{pq}$ of radius j.

6.1. Proof of Theorem 2.1

By Propositions 14.6, 14.2 and Theorem 14.3 of Kallenberg (1997), it suffices to show that $r_j(R_{n,A}) \xrightarrow{fd} r_j(R_A)$ and $\{r_j(R_{n,A})\}$ is a tight family in order to show the convergence $R_{n,A} \xrightarrow{\mathcal{D}} R_A$. Convergence of the finite dimensional distributions to Gaussian limit is a direct consequence of the multivariate central limit theorem and the representation (2.5) of the process. The covariance function C_A of (2.3) is also easy to obtain, as is the independence of the processes R_A. Then, one can write

$$R_{n,A}(\boldsymbol{t}) = \sum_{B \subset A} (-1)^{|A \setminus B|} \phi_{A,B}(\boldsymbol{t}) Y_{n,B}(\boldsymbol{t}), \tag{6.2}$$

where

$$Y_{n,B}(\boldsymbol{t}) = \frac{1}{\sqrt{n}} \sum_{j=1}^{n} \left\{ \exp(\mathrm{i}\langle \boldsymbol{t}^B, \boldsymbol{\epsilon}_j \rangle) - E\left[\exp(\mathrm{i}\langle \boldsymbol{t}^B, \boldsymbol{\epsilon}_1 \rangle)\right] \right\} \tag{6.3}$$

and $\phi_{A,B}(\boldsymbol{t}) = \prod_{l \in A \setminus B} \phi(\boldsymbol{t}^{(l)})$. The process $Y_{n,B}$ on a compact is an empirical characteristic function process which was shown by Csörgő (1981b) to be tight if the sufficient condition in (Csörgő, 1985, p. 294) is satisfied. Hence, $r_j(Y_{n,B})$ is tight. Since there is only a finite number of B's in (6.2), it follows that $\{r_j(R_{n,A})\}$ is also tight.

6.2. Proof of Theorem 2.2

By invariance assume $\boldsymbol{\mu} = \mathbf{0}$ and $\boldsymbol{\Sigma} = \boldsymbol{I}$. Let $\boldsymbol{e}_j^{(k)} = \boldsymbol{S}^{-\frac{1}{2}}(\boldsymbol{\epsilon}_j^{(k)} - \overline{\boldsymbol{\epsilon}}) = \boldsymbol{\epsilon}_j^{(k)} + \boldsymbol{\Delta}_j^{(k)}$, with $\boldsymbol{\Delta}_j^{(k)} = (\boldsymbol{S}^{-\frac{1}{2}} - \boldsymbol{I})\boldsymbol{\epsilon}_j^{(k)} - \boldsymbol{S}^{-\frac{1}{2}}\overline{\boldsymbol{\epsilon}}$. Two intermediate processes are now defined. Firstly,

$$\tilde{R}_{n,A}(\boldsymbol{t}) = \frac{1}{\sqrt{n}}\sum_{j=1}^{n}\left\{\prod_{k\in A}(\exp(\mathrm{i}\langle \boldsymbol{t}^{(k)}, \boldsymbol{\epsilon}_j^{(k)}\rangle) - \phi(\boldsymbol{t}^{(k)})\right. \tag{6.4}$$

$$\left. +\mathrm{i}\sum_{\alpha\in A} a_{\alpha,j} \prod_{k\in A; k\neq\alpha}\left[\exp(\mathrm{i}\langle \boldsymbol{t}^{(k)}, \boldsymbol{\epsilon}_j^{(k)}\rangle) - \phi(\boldsymbol{t}^{(k)})\right]\exp(\mathrm{i}\langle \boldsymbol{t}^{(\alpha)}, \boldsymbol{\epsilon}_j^{(\alpha)}\rangle)\right\} \tag{6.5}$$

$$\equiv \frac{1}{\sqrt{n}}\sum_{j=1}^{n}\prod_{k\in A}[\exp(\mathrm{i}\langle \boldsymbol{t}^{(k)}, \boldsymbol{\epsilon}_j^{(k)}\rangle) - \phi(\boldsymbol{t}^{(k)})] + \mathrm{i}U_n(\boldsymbol{t}) \tag{6.6}$$

where $a_{\alpha,j} = \langle \boldsymbol{t}^{(\alpha)}, \boldsymbol{\Delta}_j^{(\alpha)}\rangle$. Secondly, define also

$$\breve{R}_{n,A}(\boldsymbol{t}) = \frac{1}{\sqrt{n}}\sum_{j=1}^{n}\prod_{k\in A}\left[\exp(\mathrm{i}\langle \boldsymbol{t}^{(k)}, \boldsymbol{\epsilon}_j^{(k)}\rangle) - \phi(\boldsymbol{t}^{(k)})\right]. \tag{6.7}$$

The proof proceeds with the following steps

$$\rho(\hat{R}_{n,A}, \tilde{R}_{n,A}) \xrightarrow{P} 0, \tag{6.8}$$

$$\rho(\tilde{R}_{n,A}, \breve{R}_{n,A}) \xrightarrow{P} 0, \tag{6.9}$$

$$\{\breve{R}_{n,A} : \ |A| > 1\} \Rightarrow \{R_A : \ |A| > 1\}. \tag{6.10}$$

The last step was proven in Theorem 2.1. A Taylor expansion of

$$\prod_{k\in A}\left[\exp(\mathrm{i}\langle \boldsymbol{t}^{(k)}, \boldsymbol{\epsilon}_j^{(k)}\rangle + a_{k,j}) - \phi(\boldsymbol{t}^{(k)})\right]$$

around $(a_{k,j})_{k\in A} = \mathbf{0}$ and Schwarz's inequality yield

$$\left|\hat{R}_{n,A}(\boldsymbol{t}) - \tilde{R}_{n,A}(\boldsymbol{t})\right| \leq 2^{|A|-2}|A|^2\|\boldsymbol{t}\|^2\frac{1}{\sqrt{n}}\sum_{j=1}^{n}\|\boldsymbol{\Delta}_j\|^2, \tag{6.11}$$

where $\boldsymbol{\Delta}_j = (\boldsymbol{\Delta}_j^{(1)}, \ldots, \boldsymbol{\Delta}_j^{(p)})$. Now

$$\frac{1}{\sqrt{n}}\sum_{j=1}^{n}\|\boldsymbol{\Delta}_j\|^2 = \sum_{k=1}^{p}\frac{1}{\sqrt{n}}\sum_{j=1}^{n}\|\boldsymbol{\Delta}_j^{(k)}\|^2, \tag{6.12}$$

where each of the p terms is expressed as

$$\frac{1}{\sqrt{n}}\sum_{j=1}^{n}\|\boldsymbol{\Delta}_j^{(k)}\|^2 = \sqrt{n}\mathrm{tr}\left[(\boldsymbol{S}^{-\frac{1}{2}} - \boldsymbol{I})^2\frac{1}{n}\sum_{j=1}^{n}\boldsymbol{\epsilon}_j^{(k)}\boldsymbol{\epsilon}_j^{(k)\intercal}\right] \tag{6.13}$$

$$- \ 2\overline{\boldsymbol{\epsilon}}^{(k)\intercal}\boldsymbol{S}^{-\frac{1}{2}}\sqrt{n}(\boldsymbol{S}^{-\frac{1}{2}} - \boldsymbol{I})\overline{\boldsymbol{\epsilon}} + \sqrt{n}\overline{\boldsymbol{\epsilon}}^{\intercal}\boldsymbol{S}^{-1}\overline{\boldsymbol{\epsilon}}. \tag{6.14}$$

In view of $\overline{\boldsymbol{\epsilon}}^{(k)} = O_P(n^{-\frac{1}{2}})$, $\overline{\boldsymbol{\epsilon}} = O_P(n^{-\frac{1}{2}})$,

$$\sqrt{np}(\boldsymbol{S}^{-\frac{1}{2}} - \boldsymbol{I}) = -\frac{1}{2\sqrt{np}} \sum_{j=1}^{n} \sum_{k=1}^{p} \left(\boldsymbol{\epsilon}_j^{(k)} \boldsymbol{\epsilon}_j^{(k)^\intercal} - \boldsymbol{I} \right) + O_P(n^{-\frac{1}{2}}) \tag{6.15}$$

and

$$\frac{1}{n} \sum_{j=1}^{n} \boldsymbol{\epsilon}_j^{(k)} \boldsymbol{\epsilon}_j^{(k)^\intercal} = \boldsymbol{I} + O_P(n^{-\frac{1}{2}}) \tag{6.16}$$

(see Henze and Wagner (1997), p.9), we obtain $\frac{1}{\sqrt{n}} \sum_{j=1}^{n} \|\boldsymbol{\Delta}_j\|^2 \xrightarrow{P} 0$. Using (6.11), $\rho_k(\hat{R}_{n,A}, \tilde{R}_{n,A}) \xrightarrow{P} 0$ and so (6.8) is proved. To establish (6.9) consider $\rho_k(\tilde{R}_{n,A}(\boldsymbol{t}), \breve{R}_{n,A}(\boldsymbol{t})) = \sup_{\|\boldsymbol{t}\| \le k} |U_n(\boldsymbol{t})|$, where

$$U_n(\boldsymbol{t}) = \sum_{\alpha \in A} \left\{ \boldsymbol{t}^{(\alpha)^\intercal} \sqrt{n} (\boldsymbol{S}^{-\frac{1}{2}} - \boldsymbol{I}) \left[A_{n,\alpha}(\boldsymbol{t}^{(A)}) - \overline{\boldsymbol{\epsilon}} B_{n,\alpha}(\boldsymbol{t}^{(A)}) \right] \right. \tag{6.17}$$

$$\left. - \boldsymbol{t}^{(\alpha)^\intercal} \sqrt{n} \overline{\boldsymbol{\epsilon}} B_{n,\alpha}(\boldsymbol{t}^{(A)}) \right\} \tag{6.18}$$

with

$$A_{n,\alpha}(\boldsymbol{t}^{(A)}) = \frac{1}{n} \sum_{j=1}^{n} \left\{ \boldsymbol{\epsilon}_j^{(\alpha)} \exp(\mathrm{i} \langle \boldsymbol{t}^{(\alpha)}, \boldsymbol{\epsilon}_j^{(\alpha)} \rangle) \prod_{k \in A; k \neq \alpha} \left[\exp(\mathrm{i} \langle \boldsymbol{t}^{(k)}, \boldsymbol{\epsilon}_j^{(k)} \rangle) - \phi(\boldsymbol{t}^{(k)}) \right] \right\} \tag{6.19}$$

and

$$B_{n,\alpha}(\boldsymbol{t}^{(A)}) = \frac{1}{n} \sum_{j=1}^{n} \left\{ \exp(\mathrm{i} \langle \boldsymbol{t}^{(\alpha)}, \boldsymbol{\epsilon}_j^{(\alpha)} \rangle) \prod_{k \in A; k \neq \alpha} \left[\exp(\mathrm{i} \langle \boldsymbol{t}^{(k)}, \boldsymbol{\epsilon}_j^{(k)} \rangle) - \phi(\boldsymbol{t}^{(k)}) \right] \right\}. \tag{6.20}$$

Note that both expressions $A_{n,\alpha}$ and $B_{n,\alpha}$ are i.i.d. averages of zero mean variables. By the uniform strong law of large numbers on compact sets (see (Ferguson, 1996, p. 108)), one can show that

$$\max_{\|\boldsymbol{t}^{(A)}\| \le k} \|A_{n,\alpha}(\boldsymbol{t}^{(A)})\| \xrightarrow{as} 0 \tag{6.21}$$

and

$$\max_{\|\boldsymbol{t}^{(A)}\| \le k} |B_{n,\alpha}(\boldsymbol{t}^{(A)})| \xrightarrow{as} 0, \tag{6.22}$$

so that (6.9) is proved.

6.3. Proof of Theorem 2.3

Let $x_{n,j} = \int_{\mathbb{R}_j^{pq}} f(\boldsymbol{y}_n(\boldsymbol{t})) \mathrm{d}G(\boldsymbol{t})$ and $x_j = \int_{\mathbb{R}_j^{pq}} f(\boldsymbol{y}(\boldsymbol{t})) \mathrm{d}G(\boldsymbol{t})$. Define the mapping $h_j : C(\mathbb{R}_j^{pq}, \mathbb{C})^2 \to \mathbb{R}$ by $h_j(\boldsymbol{y}) = \int_{\mathbb{R}_j^{pq}} f(\boldsymbol{y}(\boldsymbol{t})) \mathrm{d}G(\boldsymbol{t})$. Then, $x_{n,j} = h_j(r_j(\boldsymbol{y}_n))$ and $x_j = h_j(r_j(\boldsymbol{y}))$. By assumption, for all j, $P_{\boldsymbol{y}_n} r_j^{-1} \Rightarrow P_{\boldsymbol{y}} r_j^{-1}$. Thus, Theorem 5.1 of Billingsley (1968) and the continuity of h_j imply $P_{x_{n,j}} \Rightarrow P_{x_j}$ as $n \to \infty$, for all j. Also, the dominated convergence theorem yields $x_j \xrightarrow{\mathcal{D}} w$, as $j \to \infty$. Using Theorem 4.2 of Billingsley (1968), in order to establish $w_n \xrightarrow{\mathcal{D}} w$ as $n \to \infty$, it suffices to show that for all $\epsilon > 0$,

$$\lim_{j \to \infty} \limsup_{n \to \infty} P\{|x_{n,j} - w_n| \ge \epsilon\} = 0.$$

By Markov's inequality it suffices to show that for all $\epsilon > 0$,

$$\lim_{j\to\infty} \limsup_{n\to\infty} E|x_{n,j} - w_n|^\alpha = 0 \tag{6.23}$$

for some $\alpha > 0$. But

$$\begin{aligned} E|x_{n,j} - w_n|^\alpha &= E\left|\int_{\mathbb{R}^{pq}\setminus\mathbb{R}_j^{pq}} f(\boldsymbol{y}_n(\boldsymbol{t}))\mathrm{d}G(\boldsymbol{t})\right|^\alpha \\ &\leq E\left[\int_{\mathbb{R}^{pq}\setminus\mathbb{R}_j^{pq}} |f(\boldsymbol{y}_n(\boldsymbol{t}))|\mathrm{d}G(\boldsymbol{t})\right]^\alpha \\ &= E_{\boldsymbol{y}_n}\left(E_G\left[\mathbb{1}\{\|\boldsymbol{t}\| > j\}|f(\boldsymbol{y}_n(\boldsymbol{t}))|\right]\right)^\alpha. \end{aligned}$$

Then, using Jensen's inequality with $\alpha \geq 1$ and Fubini's theorem,

$$\begin{aligned} E|x_{n,j} - w_n|^\alpha &\leq E_{\boldsymbol{y}_n}E_G[\mathbb{1}\{\|\boldsymbol{t}\| > j\}|f(\boldsymbol{y}_n(\boldsymbol{t}))|^\alpha] \\ &= E_G E_{\boldsymbol{y}_n}[\mathbb{1}\{\|\boldsymbol{t}\| > j\}|f(\boldsymbol{y}_n(\boldsymbol{t}))|^\alpha] \\ &= \int_{\mathbb{R}^{pq}\setminus\mathbb{R}_j^{pq}} E|f(\boldsymbol{y}_n(\boldsymbol{t}))|^\alpha \mathrm{d}G(\boldsymbol{t}). \end{aligned}$$

The theorem is proved.

6.4. Proof of Theorem 2.4

Define the functional (norm) $|[x]|^2 = \int |x(\boldsymbol{t})|^2\varphi_b(\boldsymbol{t})\mathrm{d}\boldsymbol{t}$ on the subset of squared-integrable functions with respect to $\varphi_b(\boldsymbol{t})d\boldsymbol{t}$. Following (Henze and Wagner, 1997, pp. 10-11) , in order to prove the theorem, it suffices to show

$$\left(|[\breve{R}_{n,A}]|^2, |[\breve{R}_{n,B}]|^2\right) \xrightarrow{\mathcal{D}} \left(|[R_A]|^2, |[R_B]|^2\right) \tag{6.24}$$

and

$$|[\tilde{R}_{n,A} - \breve{R}_{n,A}]|^2 \xrightarrow{P} 0. \tag{6.25}$$

This is sufficient since then the triangle inequality produces

$$\left||[\breve{R}_{n,A}]| - |[\tilde{R}_{n,A}]|\right| \leq |[\breve{R}_{n,A} - \tilde{R}_{n,A}]| \xrightarrow{P} 0.$$

This implies

$$\left(|[\tilde{R}_{n,A}]|^2, |[\tilde{R}_{n,B}]|^2\right) \xrightarrow{\mathcal{D}} \left(|[R_A]|^2, |[R_B]|^2\right). \tag{6.26}$$

But from (6.11), $\left|\hat{R}_{n,A}(\boldsymbol{t}) - \tilde{R}_{n,A}(\boldsymbol{t})\right| = \|\boldsymbol{t}\|^2 o_P(1)$ which yields

$$|[\hat{R}_{n,A} - \tilde{R}_{n,A}]|^2 \xrightarrow{P} 0. \tag{6.27}$$

Again, the triangle inequality produces

$$\left||[\hat{R}_{n,A}]| - |[\tilde{R}_{n,A}]|\right| \leq |[\hat{R}_{n,A} - \tilde{R}_{n,A}]| \xrightarrow{P} 0.$$

Then, the desired result

$$\left(|[\hat{R}_{n,A}]|^2, |[\hat{R}_{n,B}]|^2\right) \xrightarrow{\mathcal{D}} \left(|[R_A]|^2, |[R_B]|^2\right) \tag{6.28}$$

could be concluded.

The convergence in (6.24) is now proved by means of arbitrary linear combinations with the use of Theorem 2.3 with $\boldsymbol{y}_n = (\breve{R}_{n,A}, \breve{R}_{n,B})$, and the continuous function $f : \mathbb{C}^2 \to \mathbb{R}$ defined by $f(x_1, x_2) = a_1|x_1|^2 + a_2|x_2|^2$ for arbitrary constants a_1 and a_2 . Theorem 2.1 states that $(\breve{R}_{n,A}, \breve{R}_{n,B}) \xrightarrow{\mathcal{D}} (R_A, R_B)$. From Proposition 14.6 of Kallenberg (1997), this remains true on all the closed balls. Note that

$$E|f(\boldsymbol{y}_n(\boldsymbol{t}))| \leq E|a_1||\breve{R}_{n,A}(\boldsymbol{t})|^2 + E|a_2||\breve{R}_{n,B}(\boldsymbol{t})|^2.$$

It can be readily shown that $E|\breve{R}_{n,A}(\boldsymbol{t})|^2 = \prod_{k\in A}(1-\phi^2(\boldsymbol{t}^{(k)}))$, which is independent of n and integrable with respect to $\varphi_b(\boldsymbol{t})d\boldsymbol{t}$. Hence, condition (2.13) is satisfied with $\alpha = 1$. The convergence in (6.24) thus holds.

To prove (6.25), proceed as follows. From (6.4), (6.7), (6.17) and the inequality $(\sum_{j=1}^s a_j)^2 \leq s\sum_{j=1}^s a_j^2$, it follows

$$|\tilde{R}_{n,A}(\boldsymbol{t}) - \breve{R}_{n,A}(\boldsymbol{t})|^2 \leq \tag{6.29}$$

$$|A|\sum_{\alpha\in A}\left[\frac{\|\boldsymbol{t}\|^2}{4}\left\|\frac{1}{\sqrt{np}}\sum_{j=1}^{n}\sum_{k=1}^{p}(\boldsymbol{\epsilon}_j^{(k)}\boldsymbol{\epsilon}_j^{(k)\intercal} - \boldsymbol{I})\right\|^2 \left\|A_{n,\alpha}(\boldsymbol{t}^{(A)})\right\|^2\right. \tag{6.30}$$

$$+\|\boldsymbol{t}\|^2\left\|A_{n,\alpha}(\boldsymbol{t}^{(A)})\right\|^2 o_P(1) + \|\boldsymbol{t}\|^2 n\left\|\overline{\boldsymbol{\epsilon}}\right\|^2\left|B_{n,\alpha}(\boldsymbol{t}^{(A)})\right|^2 \tag{6.31}$$

$$\left.+\|\boldsymbol{t}\|^2\left\|\sqrt{n}(\boldsymbol{S}^{-\frac{1}{2}} - \boldsymbol{I})\right\|^2\left\|\overline{\boldsymbol{\epsilon}}\right\|^2\left|B_{n,\alpha}(\boldsymbol{t}^{(A)})\right|^2\right]. \tag{6.32}$$

Let

$$\begin{aligned} W_n &= \int_{\mathbb{R}^p}\left|B_{n,\alpha}(\boldsymbol{t}^{(A)})\right|^2\|\boldsymbol{t}\|^2\varphi_b(\boldsymbol{t})\mathrm{d}\boldsymbol{t}, \\ \tilde{W}_n &= \int_{\mathbb{R}^p}\left\|A_{n,\alpha}(\boldsymbol{t}^{(A)})\right\|^2\|\boldsymbol{t}\|^2\varphi_b(\boldsymbol{t})\mathrm{d}\boldsymbol{t}. \end{aligned}$$

By Tonelli's theorem,

$$EW_n = \int E\left|B_{n,\alpha}(\boldsymbol{t}^{(A)})\right|^2\|\boldsymbol{t}\|^2\varphi_b(\boldsymbol{t})\mathrm{d}\boldsymbol{t}, \tag{6.33}$$

where by direct calculation

$$E\left|B_{n,\alpha}(\boldsymbol{t}^{(A)})\right|^2 = \frac{1}{n}\prod_{k\in A;k\neq\alpha}(1-\phi^2(\boldsymbol{t}^{(k)})), \tag{6.34}$$

so that $W_n = o_P(1)$. Similarly, since

$$E\left\|A_{n,\alpha}(\boldsymbol{t}^{(A)})\right\|^2 = \frac{p}{n}\prod_{k\in A;k\neq\alpha}(1-\phi^2(\boldsymbol{t}^{(k)})), \tag{6.35}$$

it follows that $\tilde{W}_n = o_P(1)$. Thus,

$$|[\tilde{R}_{n,A} - \breve{R}_{n,A}]|^2 \leq o_P(1). \tag{6.36}$$

6.5. Proof of Theorem 3.1

As in the proof of Theorem 2.1, it suffices to show that $r_j(R_{n,A}) \xrightarrow{fd} r_j(R_A)$ and $\{r_j(R_{n,A})\}$ is a tight family. We have

$$\begin{aligned} R_{n,A}(\boldsymbol{t}) &= \frac{1}{\sqrt{n}} \sum_{l=1}^{pc_n} \prod_{k\in A} \left[\exp\left(\mathrm{i}\langle \boldsymbol{t}^{(k)}, \boldsymbol{\epsilon}_l^{(k)} \rangle \right) - \phi(\boldsymbol{t}^{(k)}) \right] \\ &\quad + \frac{1}{\sqrt{n}} \sum_{l=pc_n+1}^{n-p+1} \prod_{k\in A} \left[\exp\left(\mathrm{i}\langle \boldsymbol{t}^{(k)}, \boldsymbol{\epsilon}_l^{(k)} \rangle \right) - \phi(\boldsymbol{t}^{(k)}) \right] \\ &\equiv \tilde{R}_{n,A}(\boldsymbol{t}) + r_n(\boldsymbol{t}), \end{aligned}$$

where $c_n = \left[\frac{n-p+1}{p} \right]$ is the integer part. The process $R_{n,A}$ is asymptotically equivalent to $\tilde{R}_{n,A}$ since

$$\rho_j(R_{n,A}, \tilde{R}_{n,A}) \le \frac{1}{\sqrt{n}} \sum_{l=pc_n+1}^{n-p+1} 2^{|A|} \xrightarrow{P} 0. \tag{6.37}$$

Let

$$Y_l^A(\boldsymbol{t}) = \prod_{k\in A} \left[\exp\left(\mathrm{i}\langle \boldsymbol{t}^{(k)}, \boldsymbol{\epsilon}_l^{(k)} \rangle \right) - \phi(\boldsymbol{t}^{(k)}) \right], \; l = 1, 2, \ldots, \tag{6.38}$$

which is an m-dependent sequence. Thus, using a multivariate central limit theorem for m-dependent complex random vectors obtained by applying the Cramér-Wold device in connection with a complex variant of Theorem 11 in Ferguson (1996), the finite dimensional convergence to Gaussian limit and the covariance function are obtained. It remains to show that $\{\tilde{R}_{n,A}\}$ is a tight sequence. Since $\{1, 2, \ldots, pc_n\} = \cup_{h=1}^{p} \{pl + h;\, 0 \le l < c_n\}$, we have $r_j(\tilde{R}_{n,A})(\boldsymbol{t}) = \sum_{h=1}^{p} \sum_{B\subset A} (-1)^{|A\setminus B|} \phi_{A,B}(\boldsymbol{t}) Y_{n,h,B}^j(\boldsymbol{t})$, where $\phi_{A,B}(\boldsymbol{t}) = \prod_{l\in A\setminus B} \phi(\boldsymbol{t}^{(l)})$ and

$$Y_{n,h,B}^j(\boldsymbol{t}) = \frac{1}{\sqrt{n}} \sum_{l=0}^{c_n-1} \left[\prod_{j\in B} \exp\left(\mathrm{i}\langle \boldsymbol{t}^{(j)}, \boldsymbol{\epsilon}_{pl+h}^{(j)} \rangle \right) - \prod_{j\in B} \phi(\boldsymbol{t}^{(j)}) \right]. \tag{6.39}$$

To show that $\{Y_{n,h,B}^j\}$ is tight, proceed as in proof of Theorem 2.1. Since there is only a finite number of h's and B's, tightness follows and the theorem is proved.

6.6. Proof of Theorem 3.2

Let

$$\tilde{R}_{n,A}(\boldsymbol{t}) = \frac{1}{\sqrt{n}} \sum_{j=1}^{n-p+1} \prod_{k\in A} [\exp(\mathrm{i}\langle \boldsymbol{t}^{(k)}, \boldsymbol{\epsilon}_j^{(k)} \rangle) - \phi(\boldsymbol{t}^{(k)})] + \mathrm{i} U_n(\boldsymbol{t}), \tag{6.40}$$

where $U_n(\boldsymbol{t})$ is defined as in (6.4) and where $\boldsymbol{\Delta}_j^{(k)} = (\boldsymbol{S}^{-\frac{1}{2}} - \boldsymbol{I})\boldsymbol{\epsilon}_j^{(k)} - \boldsymbol{S}^{-\frac{1}{2}} \bar{\boldsymbol{u}}$. Let also

$$\breve{R}_{n,A}(\boldsymbol{t}) = \frac{1}{\sqrt{n}} \sum_{j=1}^{n-p+1} \prod_{k\in A} \left[\exp(\mathrm{i}\langle \boldsymbol{t}^{(k)}, \boldsymbol{\epsilon}_j^{(k)} \rangle) - \phi(\boldsymbol{t}^{(k)}) \right]. \tag{6.41}$$

Proceed as in the proof of Theorem 2.2 to establish (6.8), (6.9) and (6.10). The proofs of (6.8) and (6.9) are established along the lines in the proof of Theorem 2.2. Step (6.10) was proven in Theorem 3.1.

6.7. Proof of Theorem 3.3

Follow the lines in the proof of Theorem 2.4 but use Theorem 3.1 instead of Theorem 2.1, and replace (6.29) with

$$|\tilde{R}_{n,A}(\boldsymbol{t}) - \breve{R}_{n,A}(\boldsymbol{t})|^2 \leq \tag{6.42}$$

$$|A| \sum_{\alpha \in A} \left[\frac{\|\boldsymbol{t}\|^2}{4} \left\| \frac{1}{\sqrt{n}} \sum_{j=1}^{n} (\boldsymbol{u}_j \boldsymbol{u}_j^{\mathsf{T}} - \boldsymbol{I}) \right\|^2 \left\| A_{n,\alpha}(\boldsymbol{t}^{(A)}) \right\|^2 \right. \tag{6.43}$$

$$+ \|\boldsymbol{t}\|^2 \left\| A_{n,\alpha}(\boldsymbol{t}^{(A)}) \right\|^2 o_P(1) + \|\boldsymbol{t}\|^2 n \|\overline{\boldsymbol{u}}\|^2 \left| B_{n,\alpha}(\boldsymbol{t}^{(A)}) \right|^2 \tag{6.44}$$

$$\left. + \|\boldsymbol{t}\|^2 \left\| \sqrt{n} (\boldsymbol{S}^{-\frac{1}{2}} - \boldsymbol{I}) \right\|^2 \|\overline{\boldsymbol{u}}\|^2 \left| B_{n,\alpha}(\boldsymbol{t}^{(A)}) \right|^2 \right]. \tag{6.45}$$

Replace also n with $n - p + 1$ in (6.34) and (6.35).

ACKNOWLEDGEMENTS

The first author would like to thank the National Sciences and Engineering Research Council of Canada for financial support through grant 97303-99. This paper is part of the second author's Ph.D. thesis.

BIBLIOGRAPHY

[1] Baringhaus, L., Henze, N., 1988. A consistent test for multivariate normality based on the empirical characteristic function. Metrika 35 (6), 339–348. 56

[2] Billingsley, P., 1968. Convergence of probability measures. John Wiley & Sons Inc., New York. 55, 68

[3] Blum, J. R., Kiefer, J., Rosenblatt, M., 1961. Distribution free tests of independence based on the sample distribution function. Ann. Math. Statist. 32, 485–498. 51, 52

[4] Chen, G., Lockhart, R. A., Stephens, M. A., 2002. Box-cox transformations in linear models: Large sample theory and tests of normality. Canad. J. Statist. 30 (2), 177–234. 52

[5] Conway, J. B., 1985. A course in functional analysis. Springer-Verlag, New York. 59

[6] Cools, R., 1999. Monomial cubature rules since "Stroud": a compilation. II. J. Comput. Appl. Math. 112 (1-2), 21–27, numerical evaluation of integrals. 61

[7] Cools, R., Rabinowitz, P., 1993. Monomial cubature rules since "Stroud": a compilation. J. Comput. Appl. Math. 48 (3), 309–326. 61

[8] Csörgő, S., 1981a. Limit behaviour of the empirical characteristic function. Ann. Probab. 9 (1), 130–144. 52, 53

[9] Csörgő, S., 1981b. Multivariate empirical characteristic functions. Z. Wahrsch. Verw. Gebiete 55 (2), 203–229. 66

[10] Csörgő, S., 1985. Testing for independence by the empirical characteristic function. J. Multivariate Anal. 16 (3), 290–299. 53, 66

[11] Davies, R. B., 1973. Numerical inversion of a characteristic function. Biometrika 60, 415–417. 59

[12] Davies, R. B., 1980. [Algorithm AS 155] The distribution of a linear combination of χ^2 random variables (AS R53: 84V33 p366- 369). Appl. Statistics 29, 323–333. 59

[13] de Wet, T., Randles, R. H., 1987. On the effect of substituting parameter estimators in limiting χ^2 U and V statistics. Ann. Statist. 15 (1), 398–412. 53, 56

[14] Deheuvels, P., Martynov, G. V., 1996. Cramér-von Mises-type tests with applications to tests of independence for multivariate extreme-value distributions. Comm. Statist. Theory Methods 25 (4), 871–908. 59

[15] Ferguson, T. S., 1996. A course in large sample theory. Chapman & Hall, London. 68, 71

[16] Feuerverger, A., 1993. A consistent test for bivariate dependence. Internat. Statist. Rev. 61 (2), 419–433. 52

[17] Feuerverger, A., Mureika, R. A., 1977. The empirical characteristic function and its applications. Ann. Statist. 5 (1), 88–97. 53

[18] Ghoudi, K., Kulperger, R. J., Rémillard, B., 2001. A nonparametric test of serial independence for time series and residuals. J. Multivariate Anal. 79, 191–218. 52, 53, 54, 57, 62

[19] Henze, N., Wagner, T., 1997. A new approach to the BHEP tests for multivariate normality. J. Multivariate Anal. 62, 1–23. 56, 68, 69

[20] Henze, N., Zirkler, B., 1990. A class of invariant consistent tests for multivariate normality. Comm. Statist. Theory Methods 19, 3595–3617. 56

[21] Imhof, J. P., 1961. Computing the distribution of quadratic forms in normal variables. Biometrika 48, 419–426. 59

[22] Kallenberg, O., 1997. Foundations of modern probability. Springer-Verlag, New York. 66, 70

[23] Kellermeier, J., 1980. The empirical characteristic function and large sample hypothesis testing. J. Multivariate Anal. 10 (1), 78–87. 55

[24] Lee, Y.-S., Lee, M. C., 1992. On the derivation and computation of the Cornish-Fisher expansion. Austral. J. Statist. 34 (3), 443–450. 59

[25] Lee, Y.-S., Lin, T.-K., 1992. [Algorithm AS 269] High order Cornish-Fisher expansion. Appl. Statistics 41, 233–240. 59

[26] Marcus, M. B., 1981. Weak convergence of the empirical characteristic function. Ann. Probab. 9 (2), 194–201. 53

[27] Rencher, A. C., 2002. Methods of multivariate analysis, 2nd Edition. Wiley Series in Probability and Statistics. Wiley-Interscience [John Wiley & Sons], New York. 65

[28] Stroud, A., 1967. Some seventh degree integration formulas for symmetric regions. SIAM J. Numer. Anal. 4, 37–44. 60

[29] Stroud, A. H., 1971. Approximate calculation of multiple integrals. Prentice-Hall Inc., Englewood Cliffs, N.J., prentice-Hall Series in Automatic Computation. 59, 60

CHAPITRE 4

Conclusion

La question initiale de recherche consistait à explorer différentes pistes pouvant permettre de valider certaines hypothèses qui sont souvent faites dans le contexte de séries stochastiques dépendantes, et en particulier l'hypothèse de normalité et d'indépendance du bruit blanc pour lequel peu de travaux ont été publiés. Dans le Chapitre 2, nous nous sommes attaqués à la première partie de ce problème en considérant la mise au point d'un test de normalité pour les innovations d'un modèle ARMA univarié de moyenne connue. Les modèles ARMA univariés sont en effet très connus et utilisés dans la pratique. Cette construction utilise la stratégie des tests lisses de Neyman (1937) enrichie par la technologie « data-driven » de Ledwina (1994) qui sont, selon les travaux de Ledwina et son équipe, théoriquement plus performants que les autres approches. Le test obtenu autorise une bonne puissance en comparaison des rares tests existants et son niveau est atteint et tenu assez rapidement. Une extension de ce travail est en cours pour considérer le cas où la moyenne est inconnue, estimée par la méthode du maximum de vraisemblance. Il sera ensuite possible de généraliser les résultats obtenus au cas d'un ARMA multivarié, et probablement aussi à celui d'un ARIMA multivarié.

Le chapitre suivant aborde le problème consistant à tester l'indépendance de données sérielles. L'approche des tests lisses n'étant pas encore développée pour le cas de séries multivariées, des résultats théoriques sur la convergence faible de processus stochastiques à valeurs dans un espace de Fréchet ont été employés. En utilisant la notion de convergence des lois de dimension finie et la tension d'une famille, la convergence faible d'un certain processus empirique basé sur la fonction caractéristique a été démontrée. L'application d'une fonctionnelle de type Cramér-von Mises a alors permis l'obtention d'un test d'indépendance multivarié pour des marginales Gaussiennes fixées. Ce résultat est d'un intérêt non négligeable dans le domaine des données familiales. Il a ensuite été étendu au cas de données dépendantes afin de produire un test multivarié d'indépendance sérielle, qui pourrait possiblement être appliqué pour vérifier l'hypothèse de bruit blanc dans le cas de la régression multivariée et celui d'innovations Gaussiennes d'un modèle ARMA. En considérant le cas d'ensembles A de cardinal unité, il sera possible de bâtir un test de l'hypothèse composée simultanément de la multinormalité et de l'indépendance. Pour cela, certains problèmes doivent encore être résolus comme la caractérisation de la loi de deux statistiques de Cramér-von

Mises dépendantes à partir de la donnée de leur covariance. Une autre extension importante consiste à se débarasser de l'hypothèse de multinormalité des marginales afin d'obtenir un test d'indépendance complètement non-paramétrique. Un théorème de Csörgő (1981) de type Glivenko-Cantelli pour la fonction caractéristique donne espoir d'atteindre cet objectif. Des développements récents sur les tests de normalité basés sur un processus, tels ceux exposés dans Chen et al. (2002), montrent l'intérêt actuel de nombreux chercheurs dans ce domaine.

BIBLIOGRAPHIE

[1] Chen, G., Lockhart, R. A., Stephens, M. A., 2002. Box-cox transformations in linear models : Large sample theory and tests of normality. Canad. J. Statist. 30 (2), 177–234. 76

[2] Csörgő, S., 1981. Multivariate empirical characteristic functions. Z. Wahrsch. Verw. Gebiete 55 (2), 203–229. 76

[3] Ledwina, T., 1994. Data-driven version of Neyman's smooth test of fit. J. Amer. Statist. Assoc. 89 (427), 1000–1005. 75

[4] Neyman, J., 1937. Smooth test for goodness of fit. Skand. Aktuar. 20, 149–199. 75

INDEX DES AUTEURS

Anděl (1997), 27
Baringhaus and Henze (1988), 56
Beiser (1985), 27
Billingsley (1968), 19, 20, 22, 55, 68
Blum et al. (1961), 51, 52
Brockett et al. (1988), 26, 33
Brockwell and Davis (1991), 26–28, 42–45
Burn (1987), 39
Chen et al. (2002), 52, 76
Conway (1985), 59
Cools and Rabinowitz (1993), 61
Cools (1999), 61
Cromwell et al. (1994), 36
Csörgő (1981), 76
Csörgő (1981a), 52, 53
Csörgő (1981b), 66
Csörgő (1985), 53, 66
D'Agostino and Stephens (1986), 27, 36
Davies (1973), 59
Davies (1980), 59
Deheuvels and Martynov (1996), 59
Ducharme and Lafaye de Micheaux (2002), 29, 34, 39
Epps (1987), 26
Ferguson (1996), 22, 68, 71
Feuerverger and Mureika (1977), 53
Feuerverger (1993), 52
Fisher (1925), 14
Frances (1998), 36
Funkhauser (1936), 12, 15
Gasser (1975), 26
Ghoudi et al. (2001), 18, 52–54, 57, 62
Gouriéroux and Monfort (1995), 29
Granger (1976), 26
Henze and Wagner (1997), 56, 68, 69
Henze and Zirkler (1990), 56
Henze (1997), 30
Heuts and Rens (1986), 26
Hinich (1982), 26

Hipel and McLeod (1994), 26, 27
Imhof (1961), 59
Janic-Wroblewska and Ledwina (2000), 31
Jarque and Bera (1987), 36
Kallenberg and Ledwina (1997a), 31, 32
Kallenberg and Ledwina (1997b), 31, 32, 39
Kallenberg (1997), 21, 66, 70
Kellermeier (1980), 20, 55
Koul and Stute (1999), 26
Ledwina (1994), 27, 31, 75
Lee and Lee (1992), 59
Lee and Lin (1992), 59
Lee and Na (2002), 27
Lomnicki (1961), 26
Lutkepohl and Schneider (1989), 26, 27, 37
Marcus (1981), 53
Marsden (1974), 17
Moore (1982), 26
Neyman (1937), 15–18, 27, 75
Ojeda et al. (1997), 27
Pierce and Gray (1985), 26, 27, 30, 36
Rao (1947), 17
Rayner and Best (1989), 31
Rencher (2002), 65
Royden (1968), 18
Sansone (1959), 28
Schwarz (1978), 31
Shea (1987), 41
Stroud (1967), 60
Stroud (1971), 59, 60
Weisberg and Bingham (1975), 36
de Wet and Randles (1987), 53, 56

ANNEXE A

Les Programmes Fortran 77 du premier article

Vu l'ampleur des programmes (plus de 13000 lignes de code !), le listing des différents programmes n'est fourni que dans la version électronique de ce document sur le site Internet http://www.theses.umontreal.ca.
Tous ces programmes ont été utilisés sur un PC (gracieusement mis à disposition par Gilles Ducharme) disposant d'un processeur de 1 mégaHz et pourvu, par mes soins, du système d'exploitation Linux Redhat 7.2.

Mode d'utilisation de mon programme de simulation :

Il faut commencer par changer les valeurs des paramètres souhaitées dans les programmes *big_prog_AR1.f, big_prog_AR2.f, big_prog_MA1.f, big_prog_MA2.f, big_prog_ARMA11.f, big_prog_ARMA12.f, big_prog_ARMA21.f, big_prog_ARMA22.f.* Les paramètres où des changements peuvent être nécessaires sont marqués, dans le code source, par le signe **.

Ensuite, il faut remplir le fichier nommé *paramentree* de la façon suivante : sur chaque ligne du fichier, il doit figurer en séquence les valeurs de $p\ q\ loi\ \varphi_1\ \varphi_2\ \theta_1\ \theta_2$ avec $p, q = 0, 1$ ou 2, $loi \in \{0, 1, 2, 3, 4, ..., 19\}$ et φ et θ dans la région assurant l'inversibilité et la causalité du processus.

Remarque 1. *Même si p (resp. q) =0, il faut donner des valeurs aux θ (resp. aux φ), mais ces valeurs ne seront pas prises en compte par le programme.*

Certains de ces fichiers, qui correspondent aux points de la grille que l'on a choisi avec Gilles Ducharme pour être dans les régions d'inversibilité, sont déjà créés dans les répertoires *FICHIERS_paramentree*. Il suffit de les mettre dans le dossier principal et de les renommer en *paramentree*.

Ensuite, on tape :
```
./compile
```

Puis :
`./a.out`

Les résultats sont alors stockés dans le dossier :
`./SIMUL/`
sous la forme de différents fichiers *para.i* et *resultat.i.xc*. Les fichiers *para.i* contiennent différents paramètres de la simulation. Les fichiers *resultat.i.xc* contiennent de nombreux résultats de la simulation et sont faciles à lire avec le logiciel Excel. Il est aussi créé un fichier *tableau.txt* qui résume cette information.

Attention
Si la simulation est interrompue, des fichiers qui sont normalement effacés à la fin de chaque simulation (*data.txt* et *ftemp*) vont subsister dans le dossier
`./SIMUL/../.`
Avant de relancer une autre série de simulations, il faut effacer ces deux fichiers. Il faut aussi déplacer (ou effacer si on ne veut plus les conserver) les fichiers *para.i*, *resultat.i.xc* et *tableau.txt* du dossier
`SIMUL.`

Déroulement des divers programmes :

Voilà comment se déroule l'exécution de mes programmes.
Le programme principal est le programme *test.f* (qui sera compilé en *a.out*).

1) Dans le programme **TEST** :
On crée le fichier *tableau.txt* qui contiendra sur chaque ligne :
Nbcle T p q loi α $\boldsymbol{\varphi}_1$ $\boldsymbol{\varphi}_2$ $\boldsymbol{\theta}_1$ $\boldsymbol{\theta}_2$ $PuissR1$ $PuissR2$ $PuissR3$ $PuissR4$ $PuissR5$ $PuissLed1$ $PuissLed2$ $PuissBD$ $PuissAD$ $PuissJB$.
Le programme **TEST** lit, dans le fichier *paramentree*, les valeurs de p, q, loi, $phi(1)$, $phi(2)$, $theta(1)$, $theta(2)$ et les stocke dans la matrice $temp$, cela permet aussi de calculer le nombre de lignes (*nbline*) du fichier *paramentree*.
Pour $i = 1$ à $nbline$:
On crée le fichier *ftemp* qui contient la ligne i du fichier *paramentree* en ne gardant bien sûr que ce qui est nécessaire. Ensuite on appelle, suivant les valeurs dans le fichier *ftemp*, le bon programme **.out* (–>2) puis on efface *ftemp* et on recommence à $i + 1$.
À chaque boucle, le programme **TEST** appelle donc l'un des programmes **ARMA22.out**, **ARMA21.out**, A**RMA12.out**, **ARMA11.out**, **MA2.out**, **MA1.out**, **AR2.out**, **AR1.out** ou **ARMA00.out**.

2) Prenons l'exemple où on appelle **ARMA22.out** :
Dans **ARMA22.out**, on va lire les paramètres nécessaires dans *ftemp* : j, loi, $phi(1)$, $phi(2)$, $theta(1)$, $theta(2)$.
On appelle le programme **creerdat_ARMA** (–>3) qui crée le fichier de données *data.txt* qui contient sur chaque ligne $sigchap2$ et $epschap$.
On appelle le programme **calcstat** (–>4) qui crée le fichier des statistiques *resultat.i.xc* (qui contient sur chaque ligne $numero - simul$ $R1$ $R2$ $R3$ $R4$ $R5$ $R6$ $R7$ $R8$ $R9$ $R10$ $Kchap1$ $Ledwi1$ $Kchap2$ $Ledwi2$ BD AD JB) et rajoute au fichier *tableau.txt* une ligne de puissances.

On efface *data.txt*.
On sort et on se retrouve donc dans la boucle du programme **TEST**.

3) Dans le programme **creerdat_ARMA** :
On simule les données :

* si $loi = 0$:
  On appelle **G05EGF** qui "sets up a reference vector for an ARMA model with normally distributed errors. It also initialises the series to a stationary position."
  On appelle **G05EWF** qui "returns the next term from an ARMA time series using the reference vector set up by **G05EGF**".

* si $loi \neq 0$:
  On appelle **simARM** (–>5) (resp. **simAR**, resp. **simMA**) Les données seront les $Mind+p+1$ à $Mind+p+nT$ valeurs de $YtpMn$ (resp. $YtpMn$, resp. $YtMn$).

Puis, on estime les paramètres : appel de **G13DCF**.
Enfin, on crée le fichier *param.i* des paramètres de la simulation et on sort pour se retrouver dans **ARMA22.out**, qui va ensuite nous envoyer dans **calcstat**.

4) Dans **calcstat** :
On met les données du fichier *data.txt* dans la matrice $valeurs$.
On crée le vecteur des $sigchap2$.
On crée la matrice des $epschap$.
On commence la grande boucle de 1 à 10000.
On affiche un décompte à l'écran.
On calcule toutes les statistiques que l'on compare aux quantiles appropriés pour incrémenter (ou pas) le compteur qui permettra de calculer les puissances.
Fin de la grande boucle.

On calcule la puissance des tests en % en se servant des compteurs créés.
On calcule les quantiles.
On calcule les rangs des valeurs des statistiques.
On calcule les moyennes et écart-types des valeurs des statistiques.
On calcule le nombre de fois où chaque polynôme a été choisi par les procédures de Ledwina.
On écrit les résultats dans *resultat.ijkl.xc*.
On écrit quelques résultats dans *para.ijkl*.
On écrit les résultats dans *tableau.txt*.
On sort pour se retrouver dans **ARMA22.out**.

5) Dans **simulARMA** :

Ce programme réalise les étapes 2 et 3 de la partie « Simulation Algorithm 1 » de l'article de Burn. Cette sous-routine remplace le vecteur $YtpMn$ en entrée par $p+Mind+n$ données simulées d'un modèle ARMA(p,q) de vecteur d'innovations de loi donnée par l'entier loi , les p premières valeurs sont les p valeurs initiales de l'étape 1 de Burn,

les $Mind$ valeurs suivantes sont les valeurs à écarter de la warm-up period de longueur $Mind$. Par conséquent, seules les n dernières valeurs devront être conservées pour la suite de la simulation.
Elle renvoie aussi le vecteur $shocks$ des $marret + p$ innovations utilisées dans l'algorithme « Initialisation 1 » de Burn, ainsi que le vecteur $Atnq$ des $n + Mind + q$ « random shocks » At de l'étape 2 de Burn.

```
c Calcul des valeurs de psi (sauf pour un MA)
      CALL ARMpsi

c Calcul de shocks et Wtip (sauf pour un MA)
      CALL shock

c Étape 2 de l'algorithme: on génère les random shocks.
c On appelle le bon générateur aléatoire suivant la valeur
c  de loi (loi=1,...,19).
Le vecteur Atnq est donc créé.

c Étape 3 de l'algorithme
c YtpMn:  Y(1-p), Y(2-p), ..., Y(0), Y(1) , ..., Y(Mind+n)
c YtpMn(i),i: 1     2           p    p+1         p+Mind+n
c Wtp:          Y(1-p), Y(2-p), ..., Y(0)
c YtpMn(i),i:    1       2           p
c Atnq: eps(1-q),eps(2-q),...,eps(0),eps(1),...,eps(n+Mind)
c Atnq(k), k: 1     2            q     q+1       n+Mind+q
```

Le vecteur $YtpMn$ est alors créé.

A.1. LE SCRIPT *compile*

```
fort77 -c -f test.f
fort77 -c big_prog_ARMA22.f
fort77 -c big_prog_ARMA21.f
fort77 -c big_prog_ARMA12.f
fort77 -c big_prog_ARMA11.f
fort77 -c big_prog_AR2.f
fort77 -c big_prog_AR1.f
fort77 -c big_prog_MA2.f
fort77 -c big_prog_MA1.f
fort77 -c big_prog_ARMA00.f
f77 test.o -lnag
f77 -o ARMA22.out big_prog_ARMA22.o -lnag
f77 -o ARMA21.out big_prog_ARMA21.o -lnag
f77 -o ARMA12.out big_prog_ARMA12.o -lnag
f77 -o ARMA11.out big_prog_ARMA11.o -lnag
f77 -o AR2.out big_prog_AR2.o -lnag
f77 -o AR1.out big_prog_AR1.o -lnag
f77 -o MA2.out big_prog_MA2.o -lnag
f77 -o MA1.out big_prog_MA1.o -lnag
f77 -o ARMA00.out big_prog_ARMA00.o -lnag
rm -f *.o
```

A.2. LISTE DES DIFFÉRENTS PROGRAMMES

```
Polynômes de Legendre:
-->H1.f
-->H2.f
-->H3.f
-->H4.f
-->H5.f
-->H6.f
-->H7.f
-->H8.f
-->H9.f
-->H10.f
Polynômes de Legendre modifiés (sans les ak):
-->H1etoile.f
-->H2etoile.f
-->H3etoile.f
-->H4etoile.f
-->H5etoile.f
-->H6etoile.f
-->H7etoile.f
-->H8etoile.f
-->H9etoile.f
-->H10etoile.f
Polynômes de Legendre modifiés (avec les ak):
-->H1isa.f
-->H2isa.f
-->H3isa.f
-->H4isa.f
-->H5isa.f
-->H6isa.f
-->H7isa.f
-->H8isa.f
-->H9isa.f
-->H10isa.f
Calcul des psi(i) comme dans l'article de Burn:
-->ARpsi.f
-->MApsi.f
-->ARMpsi.f
Calcule la densité d'un échantillon pour une loi de Laplace:
-->dlap.f
Calcule la densité d'un échantillon pour une loi Normale:
-->dnorm.f
Calcule la densité d'un échantillon pour une loi Skew-Normale:
-->dskew.f
Calcule le maximum d'un vecteur:
-->max.f
Calcule la moyenne d'un vecteur:
-->mean.f
Calcule le minimum d'un vecteur:
-->min.f
Calcule les probabilités d'une loi Normale(mu,sigma) aux points d'un  vecteur:
-->pnorm.f
Calcule les quantiles d'une loi Normale(mu,sigma^2):
-->qnorm.f
Simulation d'un échantillon d'une loi de Laplace:
-->rlap.f
Simulation d'un échantillon d'une loi Skew-Normale:
-->rlap.f
Ce programme réalise la partie Initialisation Algoritm 1, de l'article de Burn
Cette sous-routine remplace le vecteur Wtip en entrée par p données
simulées
d'un modèle ARMA(p,q)
de vecteur d'innovations de loi donnée par l'entier loi
elle renvoie aussi le vecteur shocks des marret+p innovations utilisées:
```

```
-->shock.f
Calcule la variance d'un vecteur:
-->var.f
Ce programme réalise les étapes 2 et 3 de Simulation Algorithm 1, de l'article de Burn
Cette sous-routine remplace le vecteur YtpMn en entrée par p+M+n données simulées
d'un modèle AR(p) de vecteur d'innovations de loi donnée par l'entier loi ,
les p premières valeurs sont les p valeurs initiales de l'étape 1 de Burn,
les M valeurs suivantes sont les valeurs à écarter de la warm-up
period de
longueur Mind
donc seules les n dernières valeurs devront être conservées pour la
suite de la
simulation
elle renvoie aussi le vecteur shocks des marret+p innovations
utilisées dans
l'algorithme
Initialisation 1 de Burn, ainsi que le vecteur Atn des n+Mind random
shocks At de
l'étape 2 de Burn.
-->simulAR.f
Ce programme réalise les étapes 2 et 3 de Simulation Algorithm 1, de l'article de Burn
Cette sous-routine remplace le vecteur YtpMn en entrée par p+Mind+n données simulées
d'un modèle ARMA(p,q) de vecteur d'innovations de loi donnée par l'entier loi ,
les p premières valeurs sont les p valeurs initiales de l'étape 1 de Burn,
les Mind valeurs suivantes sont les valeurs à écarter de la warm-up period de
longueur Mind
donc seules les n dernières valeurs devront être conservées pour la
suite de la simulation
elle renvoie aussi le vecteur shocks des marret+p innovations
utilisées dans l'algorithme
Initialisation 1 de Burn, ainsi que le vecteur Atnq des n+Mind+q
random shocks At de l'étape 2 de Burn.
-->simulARMA.f
Ce programme réalise les étapes 2 et 3 de Simulation Algorithm 1, de l'article de Burn
Cette sous-routine remplace le vecteur YtMn en entrée par M+n données simulées
d'un modèle MA(q) de vecteur d'innovations de loi donnée par l'entier loi ,
les M valeurs suivantes sont les valeurs a écarter de la warm-up
period de longueur Mind
donc seules les n dernières valeurs devront être conservées pour la
suite de la simulation
elle renvoie aussi le vecteur Atnq des n+Mind+q random shocks At de l'étape 2 de Burn.
-->simulMA.f
Idée de ce  programme:
Étape1)
On crée un fichier de données 'data.txt' en utilisant certaines valeurs de paramètres
à lire dans un fichier d'entrée 'ftemp' crée par le programme test.f à l'aide du
fichier 'paramentree', on stocke les résultats autres que
les données dans un fichier 'param.i'
Étape 2)
ensuite on utilise le fichier 'data.txt'
avec le programme calcstat pour créer le fichier des statistiques 'resultat.i'
Étape 3)
Ensuite, on efface le fichier 'data.txt'
-->big_prog_AR1.f
-->big_prog_AR2.f
-->big_prog_MA1.f
-->big_prog_MA2.f
-->big_prog_ARMA11.f
-->big_prog_ARMA12.f
-->big_prog_ARMA21.f
-->big_prog_ARMA22.f
Ce programme crée le fichier de données data.txt qui contient, en ligne:
sigch2 et epschap:
-->creerdat_ARMA.f
-->creerdat_MA.f
```

```
-->creerdat_AR.f
Idée:
On crée un fichier ARMA22.out, ARMA21.out, ARMA12.out, ARMA11.out,
AR2.out, AR1.out, MA2.out et MA1.out qui vont lire des valeurs de p,q,phi,teta et loi
dans le fichier param_entree(c'est big_prog_ARMA.f, big_prog_AR.f,
big_prog_MA.f compilés pour
différentes valeurs de p et q)
Par exemple, si c'est MA2.out il laissera tomber la valeur de p et phi.
Ensuite, ce programme va appeler ces différents fichier.out successivement,
à l'aide de la commande 'shell'
-->test.f
```

A.3. MOYENNES ET VARIANCES EMPIRIQUES DES DIFFÉRENTES LOIS

```
c     loi des erreurs
c     si loi=0 : Normale(0,sigma^2)
c     si loi=1 : Khi2 centrée (df1)
c     si loi=2 : Student (df2)
c     si loi=3 : Skew-Normale(lambda)
c     si loi=4 : Laplace
c     si loi=5 : Weibull(b,k)
c     si loi=6 : Gamma(p,q)
c     si loi=7 : Log-Normale(g,d)
c     si loi=8 : Beta(p,q)
c     si loi=9 : Uniform(a,b)
c     si loi=10 : Shifted exp (l,b)
c     si loi=11 : Pareto(a,k)
c     si loi=12 : Shifted Pareto
c     si loi=13 : SU(g,d)
c     si loi=14 : TU(l)
c     si loi=15 : Logistic
c     si loi=16 : SC(p,d)
c     si loi=17 : LC(p,m)
c     si loi=18 : SB(g,d)
c     si loi=19 : S(a,b)
```

Dans ce fichier, je regroupe les résultats de moyennes et de variances empiriques pour les 19 lois précédentes, pour des échantillons de 100000 observations.
J'ai retranche leur espérance a toutes ces lois lors de la simulation.

```
loi=0: N(0,1)
Moyenne=  0.000985820168 Variance=  1.00143444

loi=1: Khi2(10) centrée
Moyenne=  0.00869464802 Variance=  19.9797421

loi=2: Student(5)
Moyenne= -0.00933666901 Variance=  1.6789549

loi=3: Skew-Normale(2)
Moyenne=  0.00123123782 Variance=  0.49155381

loi=4: Laplace
Moyenne= -0.00719270677 Variance=  1.97827061

loi=5: Weibull(1,2)
Moyenne=  0.00124143833 Variance=  0.249560685

loi6: Gamma(1,2)
Moyenne=  0.00496575333 Variance=  3.99297096
```

```
loi7: Log-Normale(0,1)
Moyenne= -0.00610782268 Variance=  4.59459082

loi8: Beta(1,2)
Moyenne=  0.00039170213 Variance=  0.0558931017

loi9: Uniform(1,2)
Moyenne= -0.00129533504 Variance=  0.083051074


loi10: Shifted exponentielle(1.2)
La y a peut être un probleme... a revoir ...

loi11: Pareto(1,2)
La y a peut être un probleme... a revoir ...

loi12: Shifted Pareto
Moyenne= -0.0163636928 Variance=  9.21524524

loi13: SU(0,1)
Moyenne=  0.00324440832 Variance=  3.11910338

loi14: TU(1.5)
Moyenne= -0.00265401613 Variance=  0.351612348

loi15: Logistic
Moyenne= -0.00911216487 Variance=  3.26081122

loi16: SC(0.05,3)
Moyenne= -0.00296611369 Variance=  1.40954288

loi17: LC(0.2,3)
Moyenne= -0.00223043422 Variance=  2.44646694

loi18: SB(0,0.5)
Moyenne= -0.000192865407 Variance=  0.0987884625

loi19:
a voir car espérance n'est pas connue...
```

A.4. LE PROGRAMME MAIN

Programme test.f

```
c Idee:
c On cree un fichier ARMA22.out, ARMA21.out, ARMA12.out, ARMA11.out,
c AR2.out, AR1.out, MA2.out, MA1.out et ARMA00.out qui vont lire des valeurs de p,q,
    phi,teta et loi
c dans le fichier paramentree(c'est big_prog_ARMA.f, big_prog_AR.f, big_prog_MA.f
    compiles pour
c differentes valeurs de p et q)
c Par exemple, si c'est MA2.out il laissera tomber la valeur de p et phi.
c Ensuite, ce programe va appeller ces differents fichier.out successivement,
c a l'aide de la commande 'shell'
      PROGRAM main
c-----------------------------------------------------------------------
c Declaration des variables
c-----------------------------------------------------------------------
      INTEGER nbline,i,j
      INTEGER p,q
      INTEGER loi
      DOUBLE PRECISION vec2(2),vec3(3),vec4(4),vec5(5),vec6(6)
      DOUBLE PRECISION phi(2), teta(2)
      DOUBLE PRECISION temp(1000,7)
```

```
      CHARACTER*11 parin
      CHARACTER*12 ARMA22
      CHARACTER*12 ARMA21
      CHARACTER*12 ARMA12
      CHARACTER*12 ARMA11
      CHARACTER*12 ARMA00
      CHARACTER*9 AR2
      CHARACTER*9 AR1
      CHARACTER*9 MA2
      CHARACTER*9 MA1
      CHARACTER*7 ftmp
      CHARACTER*10 efface
      CHARACTER filtab*19
c-----------------------------------------------------------------------
c Fin de declaration des variables
c-----------------------------------------------------------------------
      efface='rm ./ftemp'
      ftmp='./ftemp'
c     C'est big_prog_ARMA.f compile avec l'option -o et p=2, q=2
      ARMA22='./ARMA22.out'
c     C'est big_prog_ARMA.f compile avec l'option -o et p=2, q=1
      ARMA21='./ARMA21.out'
c     C'est big_prog_ARMA.f compile avec l'option -o et p=1, q=2
      ARMA12='./ARMA12.out'
c     C'est big_prog_ARMA.f compile avec l'option -o et p=1, q=1
      ARMA11='./ARMA11.out'
c     C'est big_prog_AR.f compile avec l'option -o et p=2
      AR2='./AR2.out'
c     C'est big_prog_AR.f compile avec l'option -o et p=1
      AR1='./AR1.out'
c     C'est big_prog_MA.f compile avec l'option -o et q=2
      MA2='./MA2.out'
c     C'est big_prog_MA.f compile avec l'option -o et q=1
      MA1='./MA1.out'
c     C'est big_prog_ARMA.f compile avec l'option -o et p=0, q=0
c     adpate pour le cas (0,0) ??
      ARMA00='./ARMA00.out'
      filtab='./SIMUL/tableau.txt'
            OPEN(UNIT=13, FILE=filtab , STATUS='NEW' )
      WRITE(13,*) 'Nbcle T p q loi alpha phi1 phi2 teta1 teta2 PuissR1
     +PuissR2  PuissR3  PuissR4 PuissR5 PuissLed1  PuissLed2  PuissBD
     +PuissAD PuissJB QuLed1calc QuLed2calc'
            CLOSE(UNIT=13)
      parin='paramentree'
c     Lit dans le fichier paramentree, les valeurs
c     de p, q, loi, phi(1), phi(2), teta(1), teta(2)
c     et les stocke dans la matrice temp, cela permet aussi
c     de calculer le nombre de lignes de ce fichier
      OPEN(UNIT=16, FILE=parin , STATUS='OLD' )
      i=1
      DO  WHILE (2 .GT. 1)
                READ(16,*,END=10) (temp(i,j),j=1,7)
                i=i+1
      END DO
      CLOSE(16)
            c     Le fichier paramentree contient nbline=i-1 lignes
 10   nbline=i-1
      DO 20, j=1,nbline
         p=temp(j,1)
         q=temp(j,2)
         loi=temp(j,3)
         phi(1)=temp(j,4)
         phi(2)=temp(j,5)
         teta(1)=temp(j,6)
         teta(2)=temp(j,7)
```

```
c     On cree le fichier ftemp qui contient la ligne en cours du fichier paramentree
c     en ne gardant bien sur que ce qui est necessaire
      OPEN(UNIT=15, FILE=ftmp , STATUS='NEW' )
      IF ((p .EQ. 2) .AND. (q .EQ. 2)) THEN
      vec6(1)=j
      vec6(2)=loi
      vec6(3)=phi(1)
      vec6(4)=phi(2)
      vec6(5)=teta(1)
      vec6(6)=teta(2)
      WRITE(15,*) vec6
      ENDIF
      IF ((p .EQ. 2) .AND. (q .EQ. 1)) THEN
      vec5(1)=j
      vec5(2)=loi
      vec5(3)=phi(1)
      vec5(4)=phi(2)
      vec5(5)=teta(1)
      WRITE(15,*) vec5
      ENDIF
      IF ((p .EQ. 1) .AND. (q .EQ. 2)) THEN
      vec5(1)=j
      vec5(2)=loi
      vec5(3)=phi(1)
      vec5(4)=teta(1)
      vec5(5)=teta(2)
      WRITE(15,*) vec5
      ENDIF
      IF ((p .EQ. 1) .AND. (q .EQ. 1)) THEN
      vec4(1)=j
      vec4(2)=loi
      vec4(3)=phi(1)
      vec4(4)=teta(1)
      WRITE(15,*) vec4
      ENDIF
      IF ((p .EQ. 2) .AND. (q .EQ. 0)) THEN
      vec4(1)=j
      vec4(2)=loi
      vec4(3)=phi(1)
      vec4(4)=phi(2)
      WRITE(15,*) vec4
      ENDIF
      IF ((p .EQ. 1) .AND. (q .EQ. 0)) THEN
      vec3(1)=j
      vec3(2)=loi
      vec3(3)=phi(1)
      WRITE(15,*) vec3
      ENDIF
      IF ((p .EQ. 0) .AND. (q .EQ. 2)) THEN
      vec4(1)=j
      vec4(2)=loi
      vec4(3)=teta(1)
      vec4(4)=teta(2)
      WRITE(15,*) vec4
      ENDIF
      IF ((p .EQ. 0) .AND. (q .EQ. 1)) THEN
      vec3(1)=j
      vec3(2)=loi
      vec3(3)=teta(1)
      WRITE(15,*) vec3
      ENDIF
      IF ((p .EQ. 0) .AND. (q .EQ. 0)) THEN
      vec2(1)=j
      vec2(2)=loi
      WRITE(15,*) vec2
```


```
      ENDIF
      CLOSE(UNIT=15)
      c     On appelle suivant les valeurs dans le fichier ftemp, le bon programme .
          out
c     Puis on efface ftemp
         IF ((p .EQ. 2) .AND. (q .EQ. 2)) THEN
            CALL SYSTEM(ARMA22)
            CALL SYSTEM(efface)
            ENDIF
         IF ((p .EQ. 2) .AND. (q .EQ. 1)) THEN
            CALL SYSTEM(ARMA21)
            CALL SYSTEM(efface)
            ENDIF
         IF ((p .EQ. 1) .AND. (q .EQ. 2)) THEN
            CALL SYSTEM(ARMA12)
            CALL SYSTEM(efface)
            ENDIF
         IF ((p .EQ. 1) .AND. (q .EQ. 1)) THEN
            CALL SYSTEM(ARMA11)
            CALL SYSTEM(efface)
            ENDIF
         IF ((p .EQ. 2) .AND. (q .EQ. 0)) THEN
            CALL SYSTEM(AR2)
            CALL SYSTEM(efface)
            ENDIF
         IF ((p .EQ. 1) .AND. (q .EQ. 0)) THEN
            CALL SYSTEM(AR1)
            CALL SYSTEM(efface)
            ENDIF
         IF ((p .EQ. 0) .AND. (q .EQ. 2)) THEN
            CALL SYSTEM(MA2)
            CALL SYSTEM(efface)
            ENDIF
         IF ((p .EQ. 0) .AND. (q .EQ. 1)) THEN
            CALL SYSTEM(MA1)
            CALL SYSTEM(efface)
            ENDIF
         IF ((p .EQ. 0) .AND. (q .EQ. 0)) THEN
            CALL SYSTEM(ARMA00)
            CALL SYSTEM(efface)
            ENDIF
 20   CONTINUE
      END
```

A.5. LES PROGRAMMES BIG_PROG_ARMAPQ

Programme big_prog_ARMA00.f

```
c Idee de ce  programme:
c Etape1)
c On cree un fichier de donnees 'data.txt' en utilisant certaines valeurs de
    parametres
c a lire dans un fichier d'entree 'ftemp' cree par le programme test.f a l'aide du
c fichier 'paramentree', on stocke les resultats autres que
c les donnees dans un fichier 'param.i'
c Etape 2)
c ensuite on utilise le fichier 'data.txt'
c avec le programme calcstat pour creer le fichier des statistiques 'resultat.i'
c Etape 3)
c Ensuite, on efface le fichier 'data.txt'
      PROGRAM main
c     Les ** indiquent les endroits ou des changements peuvent etre necessaires
c     ------------------------------------------------------------------
c     DEBUT DE DECLARATION DES VARIABLES
c     ------------------------------------------------------------------
```

```
c     Si isa=.TRUE. (ie MEAN=.TRUE.) on prend les polynomes modifies avec les ak  **
c     Si isa=.FALSE. (ie MEAN=.FALSE.) on prend les polynomes modifies sans les ak
      LOGICAL isa
      PARAMETER(isa=.FALSE.)
c-------------------------------------------
c     parametres calcstat
c     Niveau 1 du test                                          **
      DOUBLE PRECISION alpha
      PARAMETER(alpha=0.1)
c     Niveau 2 du test                                          **
      DOUBLE PRECISION alpha2
      PARAMETER(alpha2=0.05)
c     Quantiles avec alpha
      DOUBLE PRECISION  khi(10)
c     Quantiles avec alpha2
      DOUBLE PRECISION  khi2(10)
c     Quantiles Ledwi1 et Ledwi2 avec alpha                              **
      DOUBLE PRECISION QuLed1, QuLed2
c     avec T=50
      PARAMETER(QuLed1=3.98,QuLed2=5.696)
c     avec T=100
c      PARAMETER(QuLed1=3.38,QuLed2=5.373)
c     avec T=200
c      PARAMETER(QuLed1=3.10,QuLed2=5.096)
c     Quantiles Ledwi1 et Ledwi2 avec alpha2                              **
      DOUBLE PRECISION QuLe12, QuLe22
c     avec T=50
      PARAMETER(QuLe12=5.88,QuLe22=7.662)
c     avec T=100
c      PARAMETER(QuLe12=5.43,QuLe22=7.213)
c     avec T=200
c      PARAMETER(QuLe12=4.89,QuLe22=6.911)
c     Quantiles Brockwell et Davis et Anderson Darling et JB avec alpha       **
      DOUBLE PRECISION QuBD, QuAD, QuJB
c     avec T=50
      PARAMETER(QuBD=0.920,QuAD=1.743,QuJB=4.61)
c     avec T=100
c      PARAMETER(QuBD=0.958,QuAD=1.743,QuJB=4.61)
c     avec T=200
c      PARAMETER(QuBD=0.978,QuAD=1.743,QuJB=4.61)
c     Quantiles Brockwell et Davis et Anderson Darling et JB avec alpha2      **
      DOUBLE PRECISION QuBD2, QuAD2, QuJB2
c     avec T=50
      PARAMETER(QuBD2=0.899,QuAD2=2.308,QuJB2=5.99)
c      avec T=100
c      PARAMETER(QuBD2=0.947,QuAD2=2.308,QuJB2=5.99)
c     avec T=200
c      PARAMETER(QuBD2=0.973,QuAD2=2.308,QuJB2=5.99)
c     Nombre de polynomes                                       **
      INTEGER Kp
      PARAMETER(Kp=10)
c     ordre du modele AR
      INTEGER p
      PARAMETER(p=0)
c     ordre du modele MA
      INTEGER q
      PARAMETER(q=0)
c     nombre d'observations dans mon echantillon                          **
c     Si on change la valeur de nT, il faut aller modifier
c     la valeur dans FORMAT a la fin du programme creerdat_ARMA00.f
c     et dans calcstat.f
      INTEGER nT
      PARAMETER(nT=50)
      INTEGER n,m
      PARAMETER(m=nT+1)
```

```
c     n=nombre de lignes au maximum dans le fichier de donnees          **
c     cela correspond a nbcle s'il n'y a pas eu d'erreurs
c     dans tous les cas mettre ici n=nbcle=nombre d'echantillons souhaites
      PARAMETER(n=10000)
      CHARACTER filer*8, filew*21                                       **
      DOUBLE PRECISION valeur(n,m)
      DOUBLE PRECISION sch2vc(n)
      DOUBLE PRECISION epscha(n,nT)
      INTEGER compt(Kp)
      INTEGER compt2(Kp)
      DOUBLE PRECISION statn(n,Kp)
      INTEGER KoLed1(n)
      INTEGER KoLed2(n)
      DOUBLE PRECISION Ledwi1(Kp)
      DOUBLE PRECISION Ledwi2(Kp-1)
      DOUBLE PRECISION snLed1(n)
      DOUBLE PRECISION snLed2(n)
      DOUBLE PRECISION stanBD(n)
      DOUBLE PRECISION stanAD(n), stanJB(n)
      DOUBLE PRECISION Z(nT)
      DOUBLE PRECISION D(nT)
      DOUBLE PRECISION Davant(nT), E2(nT)
      DOUBLE PRECISION U(nT)
      DOUBLE PRECISION hUetoi(nT,Kp)
      DOUBLE PRECISION Uavant(nT)
      DOUBLE PRECISION hKetoi(Kp)
      DOUBLE PRECISION hU(nT,Kp)
      DOUBLE PRECISION hK(Kp)
      DOUBLE PRECISION vecteu(Kp)
      DOUBLE PRECISION vectoi(Kp)
      DOUBLE PRECISION stat(Kp)
      DOUBLE PRECISION Zavant(nT)
      DOUBLE PRECISION BDa(nT)
      DOUBLE PRECISION BDb(nT)
      DOUBLE PRECISION BDc(nT)
      DOUBLE PRECISION BDd(nT)
      DOUBLE PRECISION Davan2(nT)
      DOUBLE PRECISION ADa(nT)
      DOUBLE PRECISION ADb(nT)
      DOUBLE PRECISION Puiss(Kp)
      DOUBLE PRECISION Puiss2(Kp)
      DOUBLE PRECISION snsort(n,Kp)
      DOUBLE PRECISION snsorv(n)
      DOUBLE PRECISION Qucalc(Kp)
      DOUBLE PRECISION snLe1s(n)
      DOUBLE PRECISION snLe2s(n)
      DOUBLE PRECISION snBDso(n)
      DOUBLE PRECISION snADso(n), snJBso(n)
c-----------------------------------------------------------------------
c     parametres creerdat
c     Nom du fichier de donnees en sortie                                **
      CHARACTER dataf*8
c     Nom du fichier de parametres en sortie                             **
      CHARACTER paramf*17
c     Nombre d'echantillons souhaites
      INTEGER nbcle,nbclea
c     ecart-type des erreurs                                            **
      DOUBLE PRECISION sigma
      PARAMETER(sigma=1)
c     moyenne dans mon modele ARMA                                      **
      DOUBLE PRECISION mu
      PARAMETER(mu=0)
c     parametre de la khi-deux                                          **
      INTEGER df1
      PARAMETER(df1=10)
```


```
c     parametre de la student                                          **
      INTEGER df2
      PARAMETER(df2=5)
c     parametre de la skew-normale                                     **
      DOUBLE PRECISION lambda
      PARAMETER(lambda=2.0)
      DOUBLE PRECISION loi5b, loi5k, loi6p, loi6q
      DOUBLE PRECISION loi7g, loi7d, loi8p, loi8q, loi9a, loi9b
      DOUBLE PRECISION loi10b, loi10l, loi11a, loi11k, loi13g, loi13d
      DOUBLE PRECISION loi14l, loi16p, loi16d, loi17p, loi17m
      DOUBLE PRECISION loi18g, loi18d, loi19a, loi19b
      PARAMETER(loi5b=1.0, loi5k=2.0, loi6p=4, loi6q=1)                  **
      PARAMETER(loi7g=0.0,loi7d=1.0, loi8p=2, loi8q=2, loi9a=0,loi9b=2)    **
      PARAMETER(loi10b=0.2,loi10l=1, loi11a=2.0,loi11k=0.5)            **
      PARAMETER(loi13g=0.0,loi13d=1.0, loi14l=1.5)                     **
      PARAMETER(loi16p=0.05,loi16d=3.0, loi17p=0.2,loi17m=3.0)          **
      PARAMETER(loi18g=0.0,loi18d=0.5, loi19a=1.1, loi19b=0.5)         **
c     loi des erreurs
c     si loi=0 : Normale(0,sigma^2)
c     si loi=1 : Khi2 centree (df1)
c     si loi=2 : Student (df2)
c     si loi=3 : Skew-Normale(lambda)
c     si loi=4 : Laplace
c     si loi=5 : Weibull(b,k)
c     si loi=6 : Gamma(p,q)
c     si loi=7 : Log-Normale(g,d)
c     si loi=8 : Beta(p,q)
c     si loi=9 : Uniform(a,b)
c     si loi=10 : Shifted exp (l,b)
c     si loi=11 : Pareto(a,k)
c     si loi=12 : Shifted Pareto
c     si loi=13 : SU(g,d)
c     si loi=14 : TU(l)
c     si loi=15 : Logistic
c     si loi=16 : SC(p,d)
c     si loi=17 : LC(p,m)
c     si loi=18 : SB(g,d)
c     si loi=19 : S(a,b)
      INTEGER loi
c     vecteur des donnees cree
      DOUBLE PRECISION donees(nT)
c     Esperances des lois SU, SB et S                                  **
      DOUBLE PRECISION EspSU,EspSB,EspS
      PARAMETER(EspSU=0.0, EspSB=0.5, EspS=0.0)
      CHARACTER*13 efface
      DOUBLE PRECISION temp(2)
      INTEGER j
      CHARACTER*5 ftemp
      CHARACTER*1 INTCH1
      CHARACTER*2 INTCH2
      CHARACTER*3 INTCH3
      CHARACTER*4 INTCH4
      DOUBLE PRECISION matric(n,18)
      INTEGER dnT
      PARAMETER(dnT=Kp)
      DOUBLE PRECISION QLed1u,QLe12u
      PARAMETER(QLed1u=QuLed1,QLe12u=QuLe12)
      DOUBLE PRECISION QLed2u,QLe22u
      PARAMETER(QLed2u=QuLed2,QLe22u=QuLe22)
      DOUBLE PRECISION QBDu,QBD2u
      PARAMETER(QBDu=QuBD,QBD2u=QuBD2)
      DOUBLE PRECISION QADu,QAD2u
      PARAMETER(QADu=QuAD,QAD2u=QuAD2)
      DOUBLE PRECISION QJBu,QJB2u
      PARAMETER(QJBu=QuJB,QJB2u=QuJB2)
```

```
      DOUBLE PRECISION phi(2), teta(2)
      INTEGER NI, NX, IFAIL
      PARAMETER(NX=4,NI=9)
      INTEGER IA(NI)
      DOUBLE PRECISION XA(NX)
c     seed=1 (non-repeatable sequence) ou seed=0 (repeatable sequence)
     **
      INTEGER seed
      PARAMETER(seed=0)
c-----------------------------------------------------------
c Fin de declaration des variables
c-----------------------------------------------------------
c G05CBF Initialise random number generating routines to give repeatable sequence
c G05CCF Initialise random number generating routines to give non-repeatable sequence
c G05CFF Save state of random number generating routines
c G05CGF Restore state of random number generating routines
      IF (seed .EQ. 1) THEN
      CALL G05CCF
      ENDIF
      IF (seed .EQ. 0) THEN
      CALL G05CBF(0)
      ENDIF
      IFAIL=0
c     Rajouter ici, si voulu, les IA et les XA a prendre a la fin du fichier para.ijkl
     :         **
c     A (de)commenter si necessaire (si on rajoute les IA et XA, on decommente)
c      CALL G05CGF(IA,NI,XA,NX,IFAIL)
      CALL G05CFF(IA,NI,XA,NX,IFAIL)
      nbcle=n
      efface='rm ./data.txt'
      ftemp='ftemp'
c     On va lire les parametres necessaires dans ftemp
c     temp va contenir sur chaque ligne: j, loi
      OPEN(UNIT=15, FILE=ftemp , STATUS='OLD' )
      READ(15,*,END=10) (temp(j),j=1,2)
      CLOSE(15)
 10   j=temp(1)
      loi=temp(2)
      phi(1)=0.0
      phi(2)=0.0
      teta(1)=0.0
      teta(2)=0.0
c     On sauvegarde la valeur de nbcle
      nbclea=nbcle
c     Nom du fichier de donnees en sortie                                    **
      dataf='data.txt'
c     Permet de convertir l'entier j en chaine de caracteres
      IF (j/10 .LT. 1) THEN
         WRITE(INTCH1, '(I1)') j
c     Nom du fichier de parametres en sortie                                 **
         paramf='./SIMUL/para.'//'000'//INTCH1
         filew='./SIMUL/resultat.'//'000'//INTCH1
      ENDIF
      IF ((j/10 .EQ. 1) .OR. ((j/10 .GT. 1) .AND. (j/10 .LT. 10))) THEN
         WRITE(INTCH2, '(I2)') j
c     Nom du fichier de parametres en sortie                                 **
         paramf='./SIMUL/para.'//'00'//INTCH2
         filew='./SIMUL/resultat.'//'00'//INTCH2
      ENDIF
      IF ((j/10 .EQ. 10) .OR. ((j/10 .GT. 10) .AND. (j/10 .LT. 100)))
     + THEN
         WRITE(INTCH3, '(I3)') j
c     Nom du fichier de parametres en sortie                                 **
         paramf='./SIMUL/para.'//'0'//INTCH3
         filew='./SIMUL/resultat.'//'0'//INTCH3
```


```
      ENDIF
      IF ((j/10 .EQ. 100) .OR. ((j/10 .GT. 100) .AND. (j/10 .LT. 1000)))
     + THEN
         WRITE(INTCH4, '(I4)') j
c     Nom du fichier de parametres en sortie                              **
         paramf='./SIMUL/para.'//INTCH4
         filew='./SIMUL/resultat.'//INTCH4
      ENDIF
      IF (loi .EQ. 0) THEN
c     Quantiles de la loi du Khi2 (K=10;alpha=0.1)                        **
      khi(1)=2.71
      khi(2)=4.61
      khi(3)=6.25
      khi(4)=7.78
      khi(5)=9.24
      khi(6)=10.64
      khi(7)=12.02
      khi(8)=13.36
      khi(9)=14.68
      khi(10)=15.99
c     Quantiles de la loi du Khi2 (K=10;alpha=0.05)                       **
      khi2(1)=3.84
      khi2(2)=5.99
      khi2(3)=7.81
      khi2(4)=9.49
      khi2(5)=11.07
      khi2(6)=12.59
      khi2(7)=14.07
      khi2(8)=15.51
      khi2(9)=16.92
      khi2(10)=18.31
      ENDIF
      IF (loi .NE. 0) THEN
c     Quantiles utilises pour le calcul de la stat(vrais quantiles calcules sous H0)
     **
c     avec alpha=0.1
      khi(1)=2.71
      khi(2)=4.61
      khi(3)=6.25
      khi(4)=7.78
      khi(5)=9.24
      khi(6)=10.64
      khi(7)=12.02
      khi(8)=13.36
      khi(9)=14.68
      khi(10)=15.99
c     Quantiles utilises pour le calcul de la stat(vrais quantiles calcules sous H0)
     **
c     avec alpha=0.05
      khi2(1)=3.84
      khi2(2)=5.99
      khi2(3)=7.81
      khi2(4)=9.49
      khi2(5)=11.07
      khi2(6)=12.59
      khi2(7)=14.07
      khi2(8)=15.51
      khi2(9)=16.92
      khi2(10)=18.31
      ENDIF
c     On peut aussi changer le nom du fichier en entree (data.txt) et     **
c     le nom du fichier en sortie (resultat.txt)
      filer='data.txt'
c-----------------------------------------------------------------------
c     -----------------------------------------------------------------
```

```
c      DEBUT DU PROGRAMME
c      ------------------------------------------------------------------
       CALL cdzero(dataf,paramf,nbcle,sigma,mu,df1,df2,
      + lambda,loi5b,loi5k,loi6p,loi6q,loi7g,loi7d,loi8p,loi8q,loi9a,
      +loi9b,loi10b,loi10l,loi11a,loi11k,loi13g,loi13d,loi14l,loi16p,
      +loi16d,loi17p,loi17m,loi18g,loi18d,loi19a,loi19b,loi,p,q,nT,
      + donees,EspSU,EspSB,EspS)
c Il faut faire modifier par cdARMA la valeur nbcle en entree pour lui faire sortir
c une nouvelle valeur nbcle qui est le nombre de bons echantillons conserves
c Ensuite, il faut utiliser cette valeur dans calcstat, comme etant la vraie valeur
c du nombre d'echantillons disponibles, la valeur de n que l'on a mise etant
c la valeur maximale possible de bons echantillons, donc les differentes matrices
     initialisees
c a l'aide de la valeur de n sont plus grandes que necessaires (elles auraient du etre
c initialisees avec la valeur de nbcle renvoyee par cd ARMA mais ce n'est pas possible
c avec fortran77) et il faut en tenir compte; donc bien regarder partout dans le
c programme calcstat la ou il y a n: OK CA C'EST FAIT!!! A METTRE EN COMMENTAIRES
     QUELQUEPART ...
          CALL calcstat(isa,khi,khi2,QuLed1,QuLe12,QuLed2,QuLe22,QuBD,
      + QuBD2,QuAD,QuAD2,QuJB,QuJB2,alpha,alpha2,Kp,p,q,nT,phi,teta,n,
      + nbcle,m,filer,filew,valeur,sch2vc,epscha,compt,compt2,statn,
      + KoLed1,KoLed2,Ledwi1,Ledwi2,snLed1,snLed2,stanBD,stanAD,stanJB,Z,
      + D,Davant,E2,U,hUetoi,Uavant,hKetoi,hU,hK,vecteu,vectoi,stat,
      + Zavant,BDa,BDb,BDc,BDd,Davan2,ADa,ADb,Puiss,Puiss2,snsort,snsorv,
      + Qucalc,snLe1s,snLe2s,snBDso,snADso,snJBso,matric,paramf,loi,
      + dnT,QLed1u,QLe12u,QLed2u,QLe22u,QBDu,QBD2u,QADu,QAD2u,QJBu,QJB2u)
           nbcle=nbclea
c      FAIRE ATTENTION:
c      voir si mes programmes ne modifient pas certains de leurs parametres en entree
c      ce qui pourrait ammener des erreurs au cours de la boucle que je vais faire dans
       le programme test.f
c      pour les differentes etapes 1), 2) et 3)
         CALL SYSTEM(efface)
c      Sauvegarde du seed
       OPEN(UNIT=14, FILE=paramf , STATUS='OLD', ACCESS='append' )
       WRITE(14,*) 'Sauvegarde du seed:'
       WRITE(14,*) 'XA:'
       WRITE(14,15) 'XA(1)=',XA(1)
       WRITE(14,15) 'XA(2)=',XA(2)
       WRITE(14,15) 'XA(3)=',XA(3)
       WRITE(14,15) 'XA(4)=',XA(4)
 15    FORMAT(A6,1F20.15)
       WRITE(14,*) 'IA:'
       WRITE(14,30) 'IA(1)=',IA(1)
       WRITE(14,30) 'IA(2)=',IA(2)
       WRITE(14,30) 'IA(3)=',IA(3)
       WRITE(14,30) 'IA(4)=',IA(4)
       WRITE(14,30) 'IA(5)=',IA(5)
       WRITE(14,30) 'IA(6)=',IA(6)
       WRITE(14,30) 'IA(7)=',IA(7)
       WRITE(14,30) 'IA(8)=',IA(8)
       WRITE(14,30) 'IA(9)=',IA(9)
 30      FORMAT(A6,1I10)
       CLOSE(UNIT=14)
       END
       INCLUDE 'mean.f'
       INCLUDE 'simulARMA.f'
       INCLUDE 'ARMpsi.f'
       INCLUDE 'shock.f'
       INCLUDE 'rskew.f'
       INCLUDE 'rlap.f'
       INCLUDE 'rpare.f'
       INCLUDE 'rspare.f'
       INCLUDE 'rSU.f'
       INCLUDE 'rTU.f'
```

```
      INCLUDE 'rSC.f'
      INCLUDE 'rLC.f'
      INCLUDE 'rSB.f'
      INCLUDE 'rS.f'
      INCLUDE 'H1etoile.f'
      INCLUDE 'H2etoile.f'
      INCLUDE 'H3etoile.f'
      INCLUDE 'H4etoile.f'
      INCLUDE 'H5etoile.f'
      INCLUDE 'H6etoile.f'
      INCLUDE 'H7etoile.f'
      INCLUDE 'H8etoile.f'
      INCLUDE 'H9etoile.f'
      INCLUDE 'H10etoile.f'
      INCLUDE 'H1isa.f'
      INCLUDE 'H2isa.f'
      INCLUDE 'H3isa.f'
      INCLUDE 'H4isa.f'
      INCLUDE 'H5isa.f'
      INCLUDE 'H6isa.f'
      INCLUDE 'H7isa.f'
      INCLUDE 'H8isa.f'
      INCLUDE 'H9isa.f'
      INCLUDE 'H10isa.f'
      INCLUDE 'H1.f'
      INCLUDE 'H2.f'
      INCLUDE 'H3.f'
      INCLUDE 'H4.f'
      INCLUDE 'H5.f'
      INCLUDE 'H6.f'
      INCLUDE 'H7.f'
      INCLUDE 'H8.f'
      INCLUDE 'H9.f'
      INCLUDE 'H10.f'
      INCLUDE 'qnorm.f'
      INCLUDE 'pnorm.f'
      INCLUDE 'min.f'
      INCLUDE 'max.f'
      INCLUDE 'var.f'
      INCLUDE 'creerdat_ARMA00.f'
      INCLUDE 'calcstat.f'
```

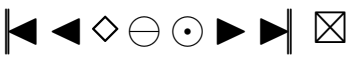
Programme big_prog_AR1.f

```
c Idee de ce  programme:
c Etape1)
c On cree un fichier de donnees 'data.txt' en utilisant certaines valeurs de
     parametres
c a lire dans un fichier d'entree 'ftemp' cree par le programme test.f a l'aide du
c fichier 'paramentree', on stocke les resultats autres que
c les donnees dans un fichier 'param.i'
c Etape 2)
c ensuite on utilise le fichier 'data.txt'
c avec le programme calcstat pour creer le fichier des statistiques 'resultat.i'
c Etape 3)
c Ensuite, on efface le fichier 'data.txt'
      PROGRAM main
c     Les ** indiquent les endroits ou des changements peuvent etre necessaires
c     ------------------------------------------------------------------
c     DEBUT DE DECLARATION DES VARIABLES
c     ------------------------------------------------------------------
c     Si isa=.TRUE. (ie MEAN=.TRUE.) on prend les polynomes modifies avec les ak  **
c     Si isa=.FALSE. (ie MEAN=.FALSE.) on prend les polynomes modifies sans les ak
      LOGICAL isa
      PARAMETER(isa=.FALSE.)
```


```
c-------------------------------------------------
c      parametres calcstat
c      Niveau 1 du test                                        **
       DOUBLE PRECISION alpha
       PARAMETER(alpha=0.1)
c      Niveau 2 du test                                        **
       DOUBLE PRECISION alpha2
       PARAMETER(alpha2=0.05)
c      Quantiles avec alpha
       DOUBLE PRECISION  khi(10)
c      Quantiles avec alpha2
       DOUBLE PRECISION  khi2(10)
c      Quantiles Ledwi1 et Ledwi2 avec alpha                          **
       DOUBLE PRECISION QuLed1, QuLed2
c      avec T=50
       PARAMETER(QuLed1=3.69,QuLed2=5.466)
c      avec T=100
c       PARAMETER(QuLed1=3.275,QuLed2=5.26)
c      avec T=200
c       PARAMETER(QuLed1=3.057,QuLed2=5.043)
c      Quantiles Ledwi1 et Ledwi2 avec alpha2                          **
       DOUBLE PRECISION QuLe12, QuLe22
c      avec T=50
       PARAMETER(QuLe12=5.41,QuLe22=7.14)
c      avec T=100
c       PARAMETER(QuLe12=5.20,QuLe22=6.97)
c      avec T=200
c       PARAMETER(QuLe12=4.751,QuLe22=6.796)
c      Quantiles Brockwell et Davis et Anderson Darling et JB avec alpha       **
       DOUBLE PRECISION QuBD, QuAD, QuJB
c      avec T=50
       PARAMETER(QuBD=0.920,QuAD=1.743,QuJB=4.61)
c      avec T=100
c       PARAMETER(QuBD=0.958,QuAD=1.743,QuJB=4.61)
c      avec T=200
c       PARAMETER(QuBD=0.978,QuAD=1.743,QuJB=4.61)
c      Quantiles Brockwell et Davis et Anderson Darling et JB avec alpha2       **
       DOUBLE PRECISION QuBD2, QuAD2, QuJB2
c      avec T=50
       PARAMETER(QuBD2=0.899,QuAD2=2.308,QuJB2=5.99)
c       avec T=100
c       PARAMETER(QuBD2=0.947,QuAD2=2.308,QuJB2=5.99)
c      avec T=200
c       PARAMETER(QuBD2=0.973,QuAD2=2.308,QuJB2=5.99)
c      Nombre de polynomes                                     **
       INTEGER Kp
       PARAMETER(Kp=10)
c      ordre du modele AR
       INTEGER p
       PARAMETER(p=1)
c      ordre du modele MA
       INTEGER q
       PARAMETER(q=0)
c      nombre d'observations dans mon echantillon                      **
c      Si on change la valeur de nT, il faut aller modifier
c      la valeur dans FORMAT a la fin du programme creerdat_AR.f
       INTEGER nT
       PARAMETER(nT=50)
       INTEGER n,m
       PARAMETER(m=nT+1)
c      n=nombre de lignes au maximum dans le fichier de donnees        **
c      cela correspond a nbcle s'il n'y a pas eu d'erreurs
c      dans tous les cas mettre ici n=nbcle=nombre d'echantillons souhaites
       PARAMETER(n=10000)
       CHARACTER filer*8, filew*21                                     **
```

```
      DOUBLE PRECISION valeur(n,m)
      DOUBLE PRECISION sch2vc(n)
      DOUBLE PRECISION epscha(n,nT)
      INTEGER compt(Kp)
      INTEGER compt2(Kp)
      DOUBLE PRECISION statn(n,Kp)
      INTEGER KoLed1(n)
      INTEGER KoLed2(n)
      DOUBLE PRECISION Ledwi1(Kp)
      DOUBLE PRECISION Ledwi2(Kp-1)
      DOUBLE PRECISION snLed1(n)
      DOUBLE PRECISION snLed2(n)
      DOUBLE PRECISION stanBD(n)
      DOUBLE PRECISION stanAD(n), stanJB(n)
      DOUBLE PRECISION Z(nT)
      DOUBLE PRECISION D(nT)
      DOUBLE PRECISION Davant(nT), E2(nT)
      DOUBLE PRECISION U(nT)
      DOUBLE PRECISION hUetoi(nT,Kp)
      DOUBLE PRECISION Uavant(nT)
      DOUBLE PRECISION hKetoi(Kp)
      DOUBLE PRECISION hU(nT,Kp)
      DOUBLE PRECISION hK(Kp)
      DOUBLE PRECISION vecteu(Kp)
      DOUBLE PRECISION vectoi(Kp)
      DOUBLE PRECISION stat(Kp)
      DOUBLE PRECISION Zavant(nT)
      DOUBLE PRECISION BDa(nT)
      DOUBLE PRECISION BDb(nT)
      DOUBLE PRECISION BDc(nT)
      DOUBLE PRECISION BDd(nT)
      DOUBLE PRECISION Davan2(nT)
      DOUBLE PRECISION ADa(nT)
      DOUBLE PRECISION ADb(nT)
      DOUBLE PRECISION Puiss(Kp)
      DOUBLE PRECISION Puiss2(Kp)
      DOUBLE PRECISION snsort(n,Kp)
      DOUBLE PRECISION snsorv(n)
      DOUBLE PRECISION Qucalc(Kp)
      DOUBLE PRECISION snLe1s(n)
      DOUBLE PRECISION snLe2s(n)
      DOUBLE PRECISION snBDso(n)
      DOUBLE PRECISION snADso(n), snJBso(n)
c-------------------------------------------------------------------
c     parametres creerdat
c     Nom du fichier de donnees en sortie                          **
      CHARACTER dataf*8
c     Nom du fichier de parametres en sortie                       **
      CHARACTER paramf*17
c     Nombre d'echantillons souhaites
      INTEGER nbcle,nbclea
c     rang d'arret dans la random shock method de Burn             **
      INTEGER marret
      PARAMETER(marret=200)
c     Induction period dans la methode de Burn                     **
      INTEGER Mind
      PARAMETER(Mind=200)
c     ecart-type des erreurs                                       **
      DOUBLE PRECISION sigma
      PARAMETER(sigma=1)
c     moyenne dans mon modele ARMA                                 **
      DOUBLE PRECISION mu
      PARAMETER(mu=0)
c     parametre de la khi-deux                                     **
      INTEGER df1
```

```
      PARAMETER(df1=2)
c     parametre de la student                                          **
      INTEGER df2
      PARAMETER(df2=5)
c     parametre de la skew-normale                                     **
      DOUBLE PRECISION lambda
      PARAMETER(lambda=2.0)
      DOUBLE PRECISION loi5b, loi5k, loi6p, loi6q
      DOUBLE PRECISION loi7g, loi7d, loi8p, loi8q, loi9a, loi9b
      DOUBLE PRECISION loi10b, loi10l, loi11a, loi11k, loi13g, loi13d
      DOUBLE PRECISION loi14l, loi16p, loi16d, loi17p, loi17m
      DOUBLE PRECISION loi18g, loi18d, loi19a, loi19b
      PARAMETER(loi18g=1.0,loi18d=1.0, loi19a=1.1, loi19b=0.5)         **
      PARAMETER(loi16p=0.2,loi16d=5.0, loi17p=0.2,loi17m=3.0)          **
      PARAMETER(loi10b=0.2,loi10l=1, loi11a=2.0,loi11k=0.5)            **
      PARAMETER(loi13g=1.0,loi13d=1.0, loi14l=0.7)                     **
      PARAMETER(loi7g=0,loi7d=1, loi8p=2, loi8q=2, loi9a=0,loi9b=2)    **
      PARAMETER(loi5b=1, loi5k=1.8, loi6p=4, loi6q=1)                  **
c     loi des erreurs                                                  **
c     si loi=0 : Normale(0,sigma^2)
c     si loi=1 : Khi2 centree (df1)
c     si loi=2 : Student (df2)
c     si loi=3 : Skew-Normale(lambda)
c     si loi=4 : Laplace
c     si loi=5 : Weibull(b,k)
c     si loi=6 : Gamma(p,q)
c     si loi=7 : Log-Normale(g,d)
c     si loi=8 : Beta(p,q)
c     si loi=9 : Uniform(a,b)
c     si loi=10 : Shifted exp (l,b)
c     si loi=11 : Pareto(a,k)
c     si loi=12 : Shifted Pareto
c     si loi=13 : SU(g,d)
c     si loi=14 : TU(l)
c     si loi=15 : Logistic
c     si loi=16 : SC(p,d)
c     si loi=17 : LC(p,m)
c     si loi=18 : SB(g,d)
c     si loi=19 : S(a,b)
      INTEGER loi
c     rm=max(p,q)
      INTEGER rm
      PARAMETER(rm=1)
c     the autoregressive coefficients of the model
      DOUBLE PRECISION phi(2)
      DOUBLE PRECISION phich(p)
c     the moving-average coefficients of the model
      DOUBLE PRECISION teta(2)
c     vecteur des donnees cree
      DOUBLE PRECISION donees(nT)
c Parametres pour G05EGF: simulation
c     the autoregressive coefficients of the model=phi     Input
      DOUBLE PRECISION A(p)
c     the moving-average coefficients of the model=teta     Input
      DOUBLE PRECISION B(q+1)
c     le vecteur des innovations     Output
      DOUBLE PRECISION R(nT)
c Parametres pour G13DCF: estimation
c     the number of initial parameter estimates   Input                 **
c     mettre p+q+1 si MEAN=.TRUE. , p+q si MEAN=.FALSE.
      INTEGER NPAR
      PARAMETER(NPAR=p+q)
c     first dimension of the array CM       Input                    **
c     mettre p+q+1 si MEAN=.TRUE. , p+q si MEAN=.FALSE.
      INTEGER ICM
```

```
      PARAMETER(ICM=p+q)
c     Workspace
      INTEGER IW(NPAR+rm+3)
c     initial parameter estimates     Input/Output
      DOUBLE PRECISION PAR(NPAR)
c     W(i,t) must be set equal to the observation at time t     Input
c     of the ith series
      DOUBLE PRECISION W(1,nT)
c     the accuracy to which the solution in PAR and QQ is required     Input        **
      DOUBLE PRECISION CGETOL
      PARAMETER(CGETOL=0.0001)
c     residual at time t for series i, for i = 1,2,...,k      Output
      DOUBLE PRECISION V(1,nT)
c     estimated first derivative of the log-likelihood function     Output
      DOUBLE PRECISION G(NPAR)
c     estimate of the correlation coefficient between the ith and     Output
c     jth elements in the PAR array
      DOUBLE PRECISION CM(ICM,NPAR)
c     Workspace
      DOUBLE PRECISION WORK((5+3*nT+2*rm)+(2*p+q+2*rm*(rm+2)+9)+
     +                      (NPAR+1)*(5*(NPAR+1)+29)/2+(rm+1)**2)
c     MEAN must be set to .TRUE. if components of mu are to     Input              **
c     be estimated and .FALSE. if all elements of mu are to be taken as zero
      LOGICAL MEAN
      PARAMETER(MEAN=.FALSE.)
c     PARHLD(i) must be set to .TRUE., if PAR(i) is to be       Input
c     held constant at its input value and .FALSE., if PAR(i) is a
c     free parameter, for i = 1,2,...,NPAR.
      LOGICAL PARHLD(NPAR)
c     EXACT must be set equal to .TRUE. if the user wishes     Input              **
c     the routine to compute exact maximum likelihood estimates.
c     EXACT must be set equal to .FALSE. if only conditional
c     likelihood estimates are required.
c     voir EXACT plus bas
      LOGICAL EXACT
c     Contient les p donnees initiales de la serie
      DOUBLE PRECISION Wtip(p)
c     marret+p innovations genere par shock.f
      DOUBLE PRECISION shocks(marret+p)
c     defini dans Burn, calcule par ARpsi.f
      DOUBLE PRECISION psi(marret+1)
c     utile dans simAR.f de longueur max(p,q+1)
      DOUBLE PRECISION phi2(marret)
c     utile dans simAR.f
      DOUBLE PRECISION YtpMn(p+Mind+nT)
c     utile dans simAR.f
      DOUBLE PRECISION Atn(nT+Mind)
c     Esperances des lois SU, SB et S                          **
      DOUBLE PRECISION EspSU,EspSB,EspS
      PARAMETER(EspSU=0, EspSB=0, EspS=0)
      CHARACTER*13 efface
      DOUBLE PRECISION temp(3)
      INTEGER j
      CHARACTER*5 ftemp
      CHARACTER*1 INTCH1
      CHARACTER*2 INTCH2
      CHARACTER*3 INTCH3
      CHARACTER*4 INTCH4
      DOUBLE PRECISION matric(n,18)
      INTEGER dnT
      PARAMETER(dnT=Kp)
      DOUBLE PRECISION QLed1u,QLe12u
      PARAMETER(QLed1u=QuLed1,QLe12u=QuLe12)
      DOUBLE PRECISION QLed2u,QLe22u
      PARAMETER(QLed2u=QuLed2,QLe22u=QuLe22)
```

```
      DOUBLE PRECISION QBDu,QBD2u
      PARAMETER(QBDu=QuBD,QBD2u=QuBD2)
      DOUBLE PRECISION QADu,QAD2u
      PARAMETER(QADu=QuAD,QAD2u=QuAD2)
      DOUBLE PRECISION QJBu,QJB2u
      PARAMETER(QJBu=QuJB,QJB2u=QuJB2)
c     the mean of the time series      Input
      DOUBLE PRECISION E
      PARAMETER(E=mu)
c     the number of autoregressive coefficients supplied      Input
      INTEGER NA
      PARAMETER(NA=p)
c     the number of moving-average coefficients supplied      Input
      INTEGER NB
      PARAMETER(NB=q+1)
c     the dimension of the array R: vecteur des innovations      Input
      INTEGER NR
      PARAMETER(NR=7)
c     the number of observed time series, k  (chez moi k=1)  Input
      INTEGER K
      PARAMETER(K=1)
c     the number of observations in each time series, n (chez moi=nT)   Input
      INTEGER N2
      PARAMETER(N2=nT)
c     the number of AR parameter matrices, p     Input
      INTEGER IP
      PARAMETER(IP=p)
c     the number of MA parameter matrices, q    Input
      INTEGER IQ
      PARAMETER(IQ=q)
c     the first dimension of the arrays QQ, W and V     Input
      INTEGER IK
      PARAMETER(IK=1)
c     the maximum number of likelihood evaluations to be     Input
c     permitted by the search procedure
      INTEGER MAXCAL
      PARAMETER(MAXCAL=40*NPAR*(NPAR+5))
c     which quantities are to be printed     Input
      INTEGER ISHOW
      PARAMETER(ISHOW=0)
c     dimension of the array WORK     Input
      INTEGER LWORK
      PARAMETER(LWORK=(5+3*(nT)+2*rm)+(2*p+q+2*rm*(rm+2)+9)+
     +                    (NPAR+1)*(5*(NPAR+1)+29)/2+(rm+1)**2)
c     dimension of the array IW     Input
      INTEGER LIW
      PARAMETER(LIW=NPAR+rm+3)
c     QQ(i,j) must be set equal to an initial estimate of     Input/Output
c     the (i,j)th element of the covariance matrix of the residual series
      DOUBLE PRECISION QQ(IK,K)
      INTEGER NI, NX, IFAIL
      PARAMETER(NX=4,NI=9)
      INTEGER IA(NI)
      DOUBLE PRECISION XA(NX)
c     seed=1 (non-repeatable sequence) ou seed=0 (repeatable sequence)
     **
      INTEGER seed
      PARAMETER(seed=0)
c-----------------------------------------------------------------
c Fin de declaration des variables
c-----------------------------------------------------------------
c G05CBF Initialise random number generating routines to give repeatable sequence
c G05CCF Initialise random number generating routines to give non-repeatable sequence
c G05CFF Save state of random number generating routines
c G05CGF Restore state of random number generating routines
```


```
      IF (seed .EQ. 1) THEN
      CALL G05CCF
      ENDIF
      IF (seed .EQ. 0) THEN
      CALL G05CBF(0)
      ENDIF
      IFAIL=0
c     Rajouter ici, si voulu, les IA et les XA a prendre a la fin du fichier para.ijkl
     :          **
c     A (de)commenter si necessaire (si on rajoute les IA et XA, on decommente)
c      CALL G05CGF(IA,NI,XA,NX,IFAIL)
      CALL G05CFF(IA,NI,XA,NX,IFAIL)
      EXACT=.TRUE.                                                    **
      nbcle=n
      efface='rm ./data.txt'
      ftemp='ftemp'
c     On va lire les parametres necessaires dans ftemp
c     temp va contenir sur chaque ligne: j, loi, phi(1)
      OPEN(UNIT=15, FILE=ftemp , STATUS='OLD' )
      READ(15,*,END=10) (temp(j),j=1,3)
      CLOSE(15)
 10   j=temp(1)
      loi=temp(2)
      phi(1)=temp(3)
      phi(2)=0.0
      teta(1)=0.0
      teta(2)=0.0
c     On sauvegarde la valeur de nbcle
      nbclea=nbcle
c     Nom du fichier de donnees en sortie                             **
      dataf='data.txt'
c     Permet de convertir l'entier j en chaine de caracteres
      IF (j/10 .LT. 1) THEN
         WRITE(INTCH1, '(I1)') j
c     Nom du fichier de parametres en sortie                          **
         paramf='./SIMUL/para.'//'000'//INTCH1
         filew='./SIMUL/resultat.'//'000'//INTCH1
      ENDIF
      IF ((j/10 .EQ. 1) .OR. ((j/10 .GT. 1) .AND. (j/10 .LT. 10))) THEN
         WRITE(INTCH2, '(I2)') j
c     Nom du fichier de parametres en sortie                          **
         paramf='./SIMUL/para.'//'00'//INTCH2
         filew='./SIMUL/resultat.'//'00'//INTCH2
      ENDIF
      IF ((j/10 .EQ. 10) .OR. ((j/10 .GT. 10) .AND. (j/10 .LT. 100)))
     + THEN
         WRITE(INTCH3, '(I3)') j
c     Nom du fichier de parametres en sortie                          **
         paramf='./SIMUL/para.'//'0'//INTCH3
         filew='./SIMUL/resultat.'//'0'//INTCH3
      ENDIF
      IF ((j/10 .EQ. 100) .OR. ((j/10 .GT. 100) .AND. (j/10 .LT. 1000)))
     + THEN
         WRITE(INTCH4, '(I4)') j
c     Nom du fichier de parametres en sortie                          **
         paramf='./SIMUL/para.'//INTCH4
         filew='./SIMUL/resultat.'//INTCH4
      ENDIF
      IF (loi .EQ. 0) THEN
c     Quantiles de la loi du Khi2 (K=10;alpha=0.1)                    **
      khi(1)=2.71
      khi(2)=4.61
      khi(3)=6.25
      khi(4)=7.78
      khi(5)=9.24
```

```
      khi(6)=10.64
      khi(7)=12.02
      khi(8)=13.36
      khi(9)=14.68
      khi(10)=15.99
c     Quantiles de la loi du Khi2 (K=10;alpha=0.05)                    **
      khi2(1)=3.84
      khi2(2)=5.99
      khi2(3)=7.81
      khi2(4)=9.49
      khi2(5)=11.07
      khi2(6)=12.59
      khi2(7)=14.07
      khi2(8)=15.51
      khi2(9)=16.92
      khi2(10)=18.31
      ENDIF
      IF (loi .NE. 0) THEN
c     Quantiles utilises pour le calcul de la stat(vrais quantiles calcules sous H0)
     **
c     avec alpha=0.1
      khi(1)=2.71
      khi(2)=4.61
      khi(3)=6.25
      khi(4)=7.78
      khi(5)=9.24
      khi(6)=10.64
      khi(7)=12.02
      khi(8)=13.36
      khi(9)=14.68
      khi(10)=15.99
c     Quantiles utilises pour le calcul de la stat(vrais quantiles calcules sous H0)
     **
c     avec alpha=0.05
      khi2(1)=3.84
      khi2(2)=5.99
      khi2(3)=7.81
      khi2(4)=9.49
      khi2(5)=11.07
      khi2(6)=12.59
      khi2(7)=14.07
      khi2(8)=15.51
      khi2(9)=16.92
      khi2(10)=18.31
      ENDIF
c     On peut aussi changer le nom du fichier en entree (data.txt) et     **
c     le nom du fichier en sortie (resultat.txt)
      filer='data.txt'
c-------------------------------------------------------------
c     ------------------------------------------------------------------
c     DEBUT DU PROGRAMME
c     ------------------------------------------------------------------
      CALL cdatAR(dataf,paramf,nbcle,marret,Mind,sigma,mu,df1,df2,
     +lambda,loi5b,loi5k,loi6p,loi6q,loi7g,loi7d,loi8p,loi8q,loi9a,
     +loi9b,loi10b,loi10l,loi11a,loi11k,loi13g,loi13d,loi14l,loi16p,
     +loi16d,loi17p,loi17m,loi18g,loi18d,loi19a,loi19b,loi,p,q,rm,nT,
     +phi,phich,donees,A,B,R,NPAR,ICM,IW,PAR,W,CGETOL,V,G,CM,WORK,MEAN,
     +PARHLD,EXACT,Wtip,shocks,psi,phi2,YtpMn,Atn,EspSU,EspSB,EspS,
     +E,NA,NB,NR,K,N2,IP,IQ,IK,MAXCAL,ISHOW,LWORK,LIW,QQ)
c Il faut faire modifier par cdARMA la valeur nbcle en entree pour lui faire sortir
c une nouvelle valeur nbcle qui est le nombre de bons echantillons conserves
c Ensuite, il faut utiliser cette valeur dans calcstat, comme etant la vraie valeur
c du nombre d'echantillons disponibles, la valeur de n que l'on a mise etant
c la valeur maximale possible de bons echantillons, donc les differentes matrices
     initialisees
```

```
c a l'aide de la valeur de n sont plus grandes que necessaires (elles auraient du etre
c initialisees avec la valeur de nbcle renvoyee par cd ARMA mais ce n'est pas possible
c avec fortran77) et il faut en tenir compte; donc bien regarder partout dans le
c programme calcstat la ou il y a n: OK CA C'EST FAIT!!! A METTRE EN COMMENTAIRES
     QUELQUEPART ...
          CALL calcstat(isa,khi,khi2,QuLed1,QuLe12,QuLed2,QuLe22,QuBD,
     + QuBD2,QuAD,QuAD2,QuJB,QuJB2,alpha,alpha2,Kp,p,q,nT,phi,teta,n,
     + nbcle,m,filer,filew,valeur,sch2vc,epscha,compt,compt2,statn,
     + KoLed1,KoLed2,Ledwi1,Ledwi2,snLed1,snLed2,stanBD,stanAD,stanJB,Z,
     + D,Davant,E2,U,hUetoi,Uavant,hKetoi,hU,hK,vecteu,vectoi,stat,
     + Zavant,BDa,BDb,BDc,BDd,Davan2,ADa,ADb,Puiss,Puiss2,snsort,snsorv,
     + Qucalc,snLe1s,snLe2s,snBDso,snADso,snJBso,matric,paramf,loi,
     + dnT,QLed1u,QLe12u,QLed2u,QLe22u,QBDu,QBD2u,QADu,QAD2u,QJBu,QJB2u)
          nbcle=nbclea
c      FAIRE ATTENTION:
c      voir si mes programmes ne modifient pas certains de leurs parametres en entree
c      ce qui pourrait ammener des erreurs au cours de la boucle que je vais faire dans
     le programme test.f
c      pour les differentes etapes 1), 2) et 3)
          CALL SYSTEM(efface)
c      Sauvegarde du seed
       OPEN(UNIT=14, FILE=paramf , STATUS='OLD', ACCESS='append' )
       WRITE(14,*) 'Sauvegarde du seed:'
       WRITE(14,*) 'XA:'
       WRITE(14,15) 'XA(1)=',XA(1)
       WRITE(14,15) 'XA(2)=',XA(2)
       WRITE(14,15) 'XA(3)=',XA(3)
       WRITE(14,15) 'XA(4)=',XA(4)
 15    FORMAT(A6,1F20.15)
       WRITE(14,*) 'IA:'
       WRITE(14,30) 'IA(1)=',IA(1)
       WRITE(14,30) 'IA(2)=',IA(2)
       WRITE(14,30) 'IA(3)=',IA(3)
       WRITE(14,30) 'IA(4)=',IA(4)
       WRITE(14,30) 'IA(5)=',IA(5)
       WRITE(14,30) 'IA(6)=',IA(6)
       WRITE(14,30) 'IA(7)=',IA(7)
       WRITE(14,30) 'IA(8)=',IA(8)
       WRITE(14,30) 'IA(9)=',IA(9)
 30       FORMAT(A6,1I10)
       CLOSE(UNIT=14)
       END
       INCLUDE 'mean.f'
       INCLUDE 'simulAR.f'
       INCLUDE 'ARpsi.f'
       INCLUDE 'shock.f'
       INCLUDE 'rskew.f'
       INCLUDE 'rlap.f'
       INCLUDE 'rpare.f'
       INCLUDE 'rspare.f'
       INCLUDE 'rSU.f'
       INCLUDE 'rTU.f'
       INCLUDE 'rSC.f'
       INCLUDE 'rLC.f'
       INCLUDE 'rSB.f'
       INCLUDE 'rS.f'
       INCLUDE 'H1etoile.f'
       INCLUDE 'H2etoile.f'
       INCLUDE 'H3etoile.f'
       INCLUDE 'H4etoile.f'
       INCLUDE 'H5etoile.f'
       INCLUDE 'H6etoile.f'
       INCLUDE 'H7etoile.f'
       INCLUDE 'H8etoile.f'
       INCLUDE 'H9etoile.f'
```

```
      INCLUDE 'H10etoile.f'
      INCLUDE 'H1isa.f'
      INCLUDE 'H2isa.f'
      INCLUDE 'H3isa.f'
      INCLUDE 'H4isa.f'
      INCLUDE 'H5isa.f'
      INCLUDE 'H6isa.f'
      INCLUDE 'H7isa.f'
      INCLUDE 'H8isa.f'
      INCLUDE 'H9isa.f'
      INCLUDE 'H10isa.f'
      INCLUDE 'H1.f'
      INCLUDE 'H2.f'
      INCLUDE 'H3.f'
      INCLUDE 'H4.f'
      INCLUDE 'H5.f'
      INCLUDE 'H6.f'
      INCLUDE 'H7.f'
      INCLUDE 'H8.f'
      INCLUDE 'H9.f'
      INCLUDE 'H10.f'
      INCLUDE 'qnorm.f'
      INCLUDE 'pnorm.f'
      INCLUDE 'min.f'
      INCLUDE 'max.f'
      INCLUDE 'var.f'
      INCLUDE 'creerdat_AR.f'
      INCLUDE 'calcstat.f'
```

Programme big_prog_AR2.f

```
c Idee de ce  programme:
c Etape1)
c On cree un fichier de donnees 'data.txt' en utilisant certaines valeurs de
     parametres
c a lire dans un fichier d'entree 'ftemp' cree par le programme test.f a l'aide du
c fichier 'paramentree', on stocke les resultats autres que
c les donnees dans un fichier 'param.i'
c Etape 2)
c ensuite on utilise le fichier 'data.txt'
c avec le programme calcstat pour creer le fichier des statistiques 'resultat.i'
c Etape 3)
c Ensuite, on efface le fichier 'data.txt'
      PROGRAM main
c     Les ** indiquent les endroits ou des changements peuvent etre necessaires
c     ------------------------------------------------------------------
c     DEBUT DE DECLARATION DES VARIABLES
c     ------------------------------------------------------------------
c     Si isa=.TRUE. (ie MEAN=.TRUE.) on prend les polynomes modifies avec les ak  **
c     Si isa=.FALSE. (ie MEAN=.FALSE.) on prend les polynomes modifies sans les ak
      LOGICAL isa
      PARAMETER(isa=.FALSE.)
c-------------------------------------
c     parametres calcstat
c     Niveau 1 du test                                        **
      DOUBLE PRECISION alpha
      PARAMETER(alpha=0.1)
c     Niveau 2 du test                                        **
      DOUBLE PRECISION alpha2
      PARAMETER(alpha2=0.05)
c     Quantiles avec alpha
      DOUBLE PRECISION  khi(10)
c     Quantiles avec alpha2
      DOUBLE PRECISION  khi2(10)
c     Quantiles Ledwi1 et Ledwi2 avec alpha                          **
```

```
      DOUBLE PRECISION QuLed1, QuLed2
c     avec T=50
c      PARAMETER(QuLed1=3.69,QuLed2=5.466)
c     avec T=100
c      PARAMETER(QuLed1=3.275,QuLed2=5.26)
c     avec T=200
      PARAMETER(QuLed1=3.057,QuLed2=5.043)
c     Quantiles Ledwi1 et Ledwi2 avec alpha2                    **
      DOUBLE PRECISION QuLe12, QuLe22
c     avec T=50
c      PARAMETER(QuLe12=5.41,QuLe22=7.14)
c     avec T=100
c      PARAMETER(QuLe12=5.20,QuLe22=6.97)
c     avec T=200
      PARAMETER(QuLe12=4.751,QuLe22=6.796)
c     Quantiles Brockwell et Davis et Anderson Darling et JB avec alpha      **
      DOUBLE PRECISION QuBD, QuAD, QuJB
c     avec T=50
c      PARAMETER(QuBD=0.920,QuAD=1.743,QuJB=4.61)
c     avec T=100
c       PARAMETER(QuBD=0.958,QuAD=1.743,QuJB=4.61)
c     avec T=200
      PARAMETER(QuBD=0.978,QuAD=1.743,QuJB=4.61)
c     Quantiles Brockwell et Davis et Anderson Darling et JB avec alpha2     **
      DOUBLE PRECISION QuBD2, QuAD2, QuJB2
c     avec T=50
c      PARAMETER(QuBD2=0.899,QuAD2=2.308,QuJB2=5.99)
c      avec T=100
c       PARAMETER(QuBD2=0.947,QuAD2=2.308,QuJB2=5.99)
c     avec T=200
      PARAMETER(QuBD2=0.973,QuAD2=2.308,QuJB2=5.99)
c     Nombre de polynomes                                  **
      INTEGER Kp
      PARAMETER(Kp=10)
c     ordre du modele AR
      INTEGER p
      PARAMETER(p=2)
c     ordre du modele MA
      INTEGER q
      PARAMETER(q=0)
c     nombre d'observations dans mon echantillon                      **
c     Si on change la valeur de nT, il faut aller modifier
c     la valeur dans FORMAT a la fin du programme creerdat_AR.f
      INTEGER nT
      PARAMETER(nT=200)
      INTEGER n,m
      PARAMETER(m=nT+1)
c     n=nombre de lignes au maximum dans le fichier de donnees        **
c     cela correspond a nbcle s'il n'y a pas eu d'erreurs
c     dans tous les cas mettre ici n=nbcle=nombre d'echantillons souhaites
      PARAMETER(n=10000)
      CHARACTER filer*8, filew*21                                     **
      DOUBLE PRECISION valeur(n,m)
      DOUBLE PRECISION sch2vc(n)
      DOUBLE PRECISION epscha(n,nT)
      INTEGER compt(Kp)
      INTEGER compt2(Kp)
      DOUBLE PRECISION statn(n,Kp)
      INTEGER KoLed1(n)
      INTEGER KoLed2(n)
      DOUBLE PRECISION Ledwi1(Kp)
      DOUBLE PRECISION Ledwi2(Kp-1)
      DOUBLE PRECISION snLed1(n)
      DOUBLE PRECISION snLed2(n)
      DOUBLE PRECISION stanBD(n)
```


```
      DOUBLE PRECISION stanAD(n), stanJB(n)
      DOUBLE PRECISION Z(nT)
      DOUBLE PRECISION D(nT)
      DOUBLE PRECISION Davant(nT), E2(nT)
      DOUBLE PRECISION U(nT)
      DOUBLE PRECISION hUetoi(nT,Kp)
      DOUBLE PRECISION Uavant(nT)
      DOUBLE PRECISION hKetoi(Kp)
      DOUBLE PRECISION hU(nT,Kp)
      DOUBLE PRECISION hK(Kp)
      DOUBLE PRECISION vecteu(Kp)
      DOUBLE PRECISION vectoi(Kp)
      DOUBLE PRECISION stat(Kp)
      DOUBLE PRECISION Zavant(nT)
      DOUBLE PRECISION BDa(nT)
      DOUBLE PRECISION BDb(nT)
      DOUBLE PRECISION BDc(nT)
      DOUBLE PRECISION BDd(nT)
      DOUBLE PRECISION Davan2(nT)
      DOUBLE PRECISION ADa(nT)
      DOUBLE PRECISION ADb(nT)
      DOUBLE PRECISION Puiss(Kp)
      DOUBLE PRECISION Puiss2(Kp)
      DOUBLE PRECISION snsort(n,Kp)
      DOUBLE PRECISION snsorv(n)
      DOUBLE PRECISION Qucalc(Kp)
      DOUBLE PRECISION snLe1s(n)
      DOUBLE PRECISION snLe2s(n)
      DOUBLE PRECISION snBDso(n)
      DOUBLE PRECISION snADso(n), snJBso(n)
c-------------------------------------------------------------------
c     parametres creerdat
c     Nom du fichier de donnees en sortie                          **
      CHARACTER dataf*8
c     Nom du fichier de parametres en sortie                       **
      CHARACTER paramf*17
c     Nombre d'echantillons souhaites
      INTEGER nbcle,nbclea
c     rang d'arret dans la random shock method de Burn             **
      INTEGER marret
      PARAMETER(marret=200)
c     Induction period dans la methode de Burn                     **
      INTEGER Mind
      PARAMETER(Mind=200)
c     ecart-type des erreurs                                       **
      DOUBLE PRECISION sigma
      PARAMETER(sigma=1)
c     moyenne dans mon modele ARMA                                 **
      DOUBLE PRECISION mu
      PARAMETER(mu=0)
c     parametre de la khi-deux                                     **
      INTEGER df1
      PARAMETER(df1=2)
c     parametre de la student                                      **
      INTEGER df2
      PARAMETER(df2=5)
c     parametre de la skew-normale                                 **
      DOUBLE PRECISION lambda
      PARAMETER(lambda=2.0)
      DOUBLE PRECISION loi5b, loi5k, loi6p, loi6q
      DOUBLE PRECISION loi7g, loi7d, loi8p, loi8q, loi9a, loi9b
      DOUBLE PRECISION loi10b, loi10l, loi11a, loi11k, loi13g, loi13d
      DOUBLE PRECISION loi14l, loi16p, loi16d, loi17p, loi17m
      DOUBLE PRECISION loi18g, loi18d, loi19a, loi19b
      PARAMETER(loi18g=1.0,loi18d=1.0, loi19a=1.1, loi19b=0.5)      **
```


```
      PARAMETER(loi16p=0.2,loi16d=5.0, loi17p=0.2,loi17m=3.0)                **
      PARAMETER(loi10b=0.2,loi10l=1, loi11a=2.0,loi11k=0.5)                  **
      PARAMETER(loi13g=1.0,loi13d=1.0, loi14l=0.7)                           **
      PARAMETER(loi7g=0,loi7d=1, loi8p=2, loi8q=2, loi9a=0,loi9b=2)          **
      PARAMETER(loi5b=1, loi5k=1.8, loi6p=4, loi6q=1)                        **
c     loi des erreurs
c     si loi=0 : Normale(0,sigma^2)
c     si loi=1 : Khi2 centree (df1)
c     si loi=2 : Student (df2)
c     si loi=3 : Skew-Normale(lambda)
c     si loi=4 : Laplace
c     si loi=5 : Weibull(b,k)
c     si loi=6 : Gamma(p,q)
c     si loi=7 : Log-Normale(g,d)
c     si loi=8 : Beta(p,q)
c     si loi=9 : Uniform(a,b)
c     si loi=10 : Shifted exp (l,b)
c     si loi=11 : Pareto(a,k)
c     si loi=12 : Shifted Pareto
c     si loi=13 : SU(g,d)
c     si loi=14 : TU(l)
c     si loi=15 : Logistic
c     si loi=16 : SC(p,d)
c     si loi=17 : LC(p,m)
c     si loi=18 : SB(g,d)
c     si loi=19 : S(a,b)
      INTEGER loi
c     rm=max(p,q)
      INTEGER rm
      PARAMETER(rm=2)
c     the autoregressive coefficients of the model
      DOUBLE PRECISION phi(2)
      DOUBLE PRECISION phich(p)
c     the moving-average coefficients of the model
      DOUBLE PRECISION teta(2)
c     vecteur des donnees cree
      DOUBLE PRECISION donees(nT)
c Parametres pour G05EGF: simulation
c     the autoregressive coefficients of the model=phi     Input
      DOUBLE PRECISION A(p)
c     the moving-average coefficients of the model=teta     Input
      DOUBLE PRECISION B(q+1)
c     le vecteur des innovations     Output
      DOUBLE PRECISION R(nT)
c Parametres pour G13DCF: estimation
c     the number of initial parameter estimates   Input                    **
c     mettre p+q+1 si MEAN=.TRUE. , p+q si MEAN=.FALSE.
      INTEGER NPAR
      PARAMETER(NPAR=p+q)
c     first dimension of the array CM       Input                       **
c     mettre p+q+1 si MEAN=.TRUE. , p+q si MEAN=.FALSE.
      INTEGER ICM
      PARAMETER(ICM=p+q)
c     Workspace
      INTEGER IW(NPAR+rm+3)
c     initial parameter estimates     Input/Output
      DOUBLE PRECISION PAR(NPAR)
c     W(i,t) must be set equal to the observation at time t     Input
c     of the ith series
      DOUBLE PRECISION W(1,nT)
c     the accuracy to which the solution in PAR and QQ is required     Input      **
      DOUBLE PRECISION CGETOL
      PARAMETER(CGETOL=0.0001)
c     residual at time t for series i, for i = 1,2,...,k     Output
      DOUBLE PRECISION V(1,nT)
```


```
c     estimated first derivative of the log-likelihood function     Output
      DOUBLE PRECISION G(NPAR)
c     estimate of the correlation coefficient between the ith and    Output
c     jth elements in the PAR array
      DOUBLE PRECISION CM(ICM,NPAR)
c     Workspace
      DOUBLE PRECISION WORK((5+3*nT+2*rm)+(2*p+q+2*rm*(rm+2)+9)+
     +                      (NPAR+1)*(5*(NPAR+1)+29)/2+(rm+1)**2)
c     MEAN must be set to .TRUE. if components of mu are to     Input          **
c     be estimated and .FALSE. if all elements of mu are to be taken as zero
      LOGICAL MEAN
      PARAMETER(MEAN=.FALSE.)
c     PARHLD(i) must be set to .TRUE., if PAR(i) is to be       Input
c     held constant at its input value and .FALSE., if PAR(i) is a
c     free parameter, for i = 1,2,...,NPAR.
      LOGICAL PARHLD(NPAR)
c     EXACT must be set equal to .TRUE. if the user wishes    Input          **
c     the routine to compute exact maximum likelihood estimates.
c     EXACT must be set equal to .FALSE. if only conditional
c     likelihood estimates are required.
c     voir EXACT plus bas
      LOGICAL EXACT
c     Contient les p donnees initiales de la serie
      DOUBLE PRECISION Wtip(p)
c     marret+p innovations genere par shock.f
      DOUBLE PRECISION shocks(marret+p)
c     defini dans Burn, calcule par ARpsi.f
      DOUBLE PRECISION psi(marret+1)
c     utile dans simAR.f de longueur max(p,q+1)
      DOUBLE PRECISION phi2(marret)
c     utile dans simAR.f
      DOUBLE PRECISION YtpMn(p+Mind+nT)
c     utile dans simAR.f
      DOUBLE PRECISION Atn(nT+Mind)
c     Esperances des lois SU, SB et S                          **
      DOUBLE PRECISION EspSU,EspSB,EspS
      PARAMETER(EspSU=0, EspSB=0, EspS=0)
      CHARACTER*13 efface
      DOUBLE PRECISION temp(4)
      INTEGER j
      CHARACTER*5 ftemp
      CHARACTER*1 INTCH1
      CHARACTER*2 INTCH2
      CHARACTER*3 INTCH3
      CHARACTER*4 INTCH4
      DOUBLE PRECISION matric(n,18)
      INTEGER dnT
      PARAMETER(dnT=Kp)
      DOUBLE PRECISION QLed1u,QLe12u
      PARAMETER(QLed1u=QuLed1,QLe12u=QuLe12)
      DOUBLE PRECISION QLed2u,QLe22u
      PARAMETER(QLed2u=QuLed2,QLe22u=QuLe22)
      DOUBLE PRECISION QBDu,QBD2u
      PARAMETER(QBDu=QuBD,QBD2u=QuBD2)
      DOUBLE PRECISION QADu,QAD2u
      PARAMETER(QADu=QuAD,QAD2u=QuAD2)
      DOUBLE PRECISION QJBu,QJB2u
      PARAMETER(QJBu=QuJB,QJB2u=QuJB2)
c     the mean of the time series      Input
      DOUBLE PRECISION E
      PARAMETER(E=mu)
c     the number of autoregressive coefficients supplied     Input
      INTEGER NA
      PARAMETER(NA=p)
c     the number of moving-average coefficients supplied     Input
```


```
      INTEGER NB
      PARAMETER(NB=q+1)
c     the dimension of the array R: vecteur des innovations      Input
      INTEGER NR
      PARAMETER(NR=9)
c     the number of observed time series, k  (chez moi k=1)  Input
      INTEGER K
      PARAMETER(K=1)
c     the number of observations in each time series, n (chez moi=nT)   Input
      INTEGER N2
      PARAMETER(N2=nT)
c     the number of AR parameter matrices, p     Input
      INTEGER IP
      PARAMETER(IP=p)
c     the number of MA parameter matrices, q    Input
      INTEGER IQ
      PARAMETER(IQ=q)
c     the first dimension of the arrays QQ, W and V     Input
      INTEGER IK
      PARAMETER(IK=1)
c     the maximum number of likelihood evaluations to be     Input
c     permitted by the search procedure
      INTEGER MAXCAL
      PARAMETER(MAXCAL=40*NPAR*(NPAR+5))
c     which quantities are to be printed     Input
      INTEGER ISHOW
      PARAMETER(ISHOW=0)
c     dimension of the array WORK     Input
      INTEGER LWORK
      PARAMETER(LWORK=(5+3*(nT)+2*rm)+(2*p+q+2*rm*(rm+2)+9)+
     +                    (NPAR+1)*(5*(NPAR+1)+29)/2+(rm+1)**2)
c     dimension of the array IW     Input
      INTEGER LIW
      PARAMETER(LIW=NPAR+rm+3)
c     QQ(i,j) must be set equal to an initial estimate of     Input/Output
c     the (i,j)th element of the covariance matrix of the residual series
      DOUBLE PRECISION QQ(IK,K)
      INTEGER NI, NX, IFAIL
      PARAMETER(NX=4,NI=9)
      INTEGER IA(NI)
      DOUBLE PRECISION XA(NX)
c     seed=1 (non-repeatable sequence) ou seed=0 (repeatable sequence)
     **
      INTEGER seed
      PARAMETER(seed=0)
c-----------------------------------------------------------------
c Fin de declaration des variables
c-----------------------------------------------------------------
c G05CBF Initialise random number generating routines to give repeatable sequence
c G05CCF Initialise random number generating routines to give non-repeatable sequence
c G05CFF Save state of random number generating routines
c G05CGF Restore state of random number generating routines
      IF (seed .EQ. 1) THEN
      CALL G05CCF
      ENDIF
      IF (seed .EQ. 0) THEN
      CALL G05CBF(0)
      ENDIF
      IFAIL=0
c     Rajouter ici, si voulu, les IA et les XA a prendre a la fin du fichier para.ijkl
     :          **
c     A (de)commenter si necessaire (si on rajoute les IA et XA, on decommente)
c      CALL G05CGF(IA,NI,XA,NX,IFAIL)
      CALL G05CFF(IA,NI,XA,NX,IFAIL)
      EXACT=.TRUE.                                                **
```

```
      nbcle=n
      efface='rm ./data.txt'
      ftemp='ftemp'
c     On va lire les parametres necessaires dans ftemp
c     temp va contenir sur chaque ligne: j, loi, phi(1), phi(2)
      OPEN(UNIT=15, FILE=ftemp , STATUS='OLD' )
      READ(15,*,END=10) (temp(j),j=1,4)
      CLOSE(15)
 10   j=temp(1)
      loi=temp(2)
      phi(1)=temp(3)
      phi(2)=temp(4)
      teta(1)=0.0
      teta(2)=0.0
c     On sauvegarde la valeur de nbcle
      nbclea=nbcle
c     Nom du fichier de donnees en sortie                                **
      dataf='data.txt'
c     Permet de convertir l'entier j en chaine de caracteres
      IF (j/10 .LT. 1) THEN
         WRITE(INTCH1, '(I1)') j
c     Nom du fichier de parametres en sortie                             **
         paramf='./SIMUL/para.'//'000'//INTCH1
         filew='./SIMUL/resultat.'//'000'//INTCH1
      ENDIF
      IF ((j/10 .EQ. 1) .OR. ((j/10 .GT. 1) .AND. (j/10 .LT. 10))) THEN
         WRITE(INTCH2, '(I2)') j
c     Nom du fichier de parametres en sortie                             **
         paramf='./SIMUL/para.'//'00'//INTCH2
         filew='./SIMUL/resultat.'//'00'//INTCH2
      ENDIF
      IF ((j/10 .EQ. 10) .OR. ((j/10 .GT. 10) .AND. (j/10 .LT. 100)))
     + THEN
         WRITE(INTCH3, '(I3)') j
c     Nom du fichier de parametres en sortie                             **
         paramf='./SIMUL/para.'//'0'//INTCH3
         filew='./SIMUL/resultat.'//'0'//INTCH3
      ENDIF
      IF ((j/10 .EQ. 100) .OR. ((j/10 .GT. 100) .AND. (j/10 .LT. 1000)))
     + THEN
         WRITE(INTCH4, '(I4)') j
c     Nom du fichier de parametres en sortie                             **
         paramf='./SIMUL/para.'//INTCH4
         filew='./SIMUL/resultat.'//INTCH4
      ENDIF
      IF (loi .EQ. 0) THEN
c     Quantiles de la loi du Khi2 (K=10;alpha=0.1)                     **
      khi(1)=2.71
      khi(2)=4.61
      khi(3)=6.25
      khi(4)=7.78
      khi(5)=9.24
      khi(6)=10.64
      khi(7)=12.02
      khi(8)=13.36
      khi(9)=14.68
      khi(10)=15.99
c     Quantiles de la loi du Khi2 (K=10;alpha=0.05)                     **
      khi2(1)=3.84
      khi2(2)=5.99
      khi2(3)=7.81
      khi2(4)=9.49
      khi2(5)=11.07
      khi2(6)=12.59
      khi2(7)=14.07
```

```
      khi2(8)=15.51
      khi2(9)=16.92
      khi2(10)=18.31
      ENDIF
      IF (loi .NE. 0) THEN
c     Quantiles utilises pour le calcul de la stat(vrais quantiles calcules sous H0)
     **
c     avec alpha=0.1
      khi(1)=2.71
      khi(2)=4.61
      khi(3)=6.25
      khi(4)=7.78
      khi(5)=9.24
      khi(6)=10.64
      khi(7)=12.02
      khi(8)=13.36
      khi(9)=14.68
      khi(10)=15.99
c     Quantiles utilises pour le calcul de la stat(vrais quantiles calcules sous H0)
     **
c     avec alpha=0.05
      khi2(1)=3.84
      khi2(2)=5.99
      khi2(3)=7.81
      khi2(4)=9.49
      khi2(5)=11.07
      khi2(6)=12.59
      khi2(7)=14.07
      khi2(8)=15.51
      khi2(9)=16.92
      khi2(10)=18.31
      ENDIF
c     On peut aussi changer le nom du fichier en entree (data.txt) et     **
c     le nom du fichier en sortie (resultat.txt)
      filer='data.txt'
c-------------------------------------------------------------
c     ------------------------------------------------------------------------
c     DEBUT DU PROGRAMME
c     ------------------------------------------------------------------------
      CALL cdatAR(dataf,paramf,nbcle,marret,Mind,sigma,mu,df1,df2,
     +lambda,loi5b,loi5k,loi6p,loi6q,loi7g,loi7d,loi8p,loi8q,loi9a,
     +loi9b,loi10b,loi10l,loi11a,loi11k,loi13g,loi13d,loi14l,loi16p,
     +loi16d,loi17p,loi17m,loi18g,loi18d,loi19a,loi19b,loi,p,q,rm,nT,
     +phi,phich,donees,A,B,R,NPAR,ICM,IW,PAR,W,CGETOL,V,G,CM,WORK,MEAN,
     +PARHLD,EXACT,Wtip,shocks,psi,phi2,YtpMn,Atn,EspSU,EspSB,EspS,
     +E,NA,NB,NR,K,N2,IP,IQ,IK,MAXCAL,ISHOW,LWORK,LIW,QQ)
c Il faut faire modifier par cdARMA la valeur nbcle en entree pour lui faire sortir
c une nouvelle valeur nbcle qui est le nombre de bons echantillons conserves
c Ensuite, il faut utiliser cette valeur dans calcstat, comme etant la vraie valeur
c du nombre d'echantillons disponibles, la valeur de n que l'on a mise etant
c la valeur maximale possible de bons echantillons, donc les differentes matrices
     initialisees
c a l'aide de la valeur de n sont plus grandes que necessaires (elles auraient du etre
c initialisees avec la valeur de nbcle renvoyee par cd ARMA mais ce n'est pas possible
c avec fortran77) et il faut en tenir compte; donc bien regarder partout dans le
c programme calcstat la ou il y a n: OK CA C'EST FAIT!!! A METTRE EN COMMENTAIRES
     QUELQUEPART ...
      CALL calcstat(isa,khi,khi2,QuLed1,QuLe12,QuLed2,QuLe22,QuBD,
     + QuBD2,QuAD,QuAD2,QuJB,QuJB2,alpha,alpha2,Kp,p,q,nT,phi,teta,n,
     + nbcle,m,filer,filew,valeur,sch2vc,epscha,compt,compt2,statn,
     + KoLed1,KoLed2,Ledwi1,Ledwi2,snLed1,snLed2,stanBD,stanAD,stanJB,Z,
     + D,Davant,E2,U,hUetoi,Uavant,hKetoi,hU,hK,vecteu,vectoi,stat,
     + Zavant,BDa,BDb,BDc,BDd,Davan2,ADa,ADb,Puiss,Puiss2,snsort,snsorv,
     + Qucalc,snLe1s,snLe2s,snBDso,snADso,snJBso,matric,paramf,loi,
     + dnT,QLed1u,QLe12u,QLed2u,QLe22u,QBDu,QBD2u,QADu,QAD2u,QJBu,QJB2u)
```

```
        nbcle=nbclea
c       FAIRE ATTENTION:
c       voir si mes programmes ne modifient pas certains de leurs parametres en entree
c       ce qui pourrait ammener des erreurs au cours de la boucle que je vais faire dans
     le programme test.f
c       pour les differentes etapes 1), 2) et 3)
        CALL SYSTEM(efface)
c       Sauvegarde du seed
        OPEN(UNIT=14, FILE=paramf , STATUS='OLD', ACCESS='append' )
        WRITE(14,*) 'Sauvegarde du seed:'
        WRITE(14,*) 'XA:'
        WRITE(14,15) 'XA(1)=',XA(1)
        WRITE(14,15) 'XA(2)=',XA(2)
        WRITE(14,15) 'XA(3)=',XA(3)
        WRITE(14,15) 'XA(4)=',XA(4)
 15     FORMAT(A6,1F20.15)
        WRITE(14,*) 'IA:'
        WRITE(14,30) 'IA(1)=',IA(1)
        WRITE(14,30) 'IA(2)=',IA(2)
        WRITE(14,30) 'IA(3)=',IA(3)
        WRITE(14,30) 'IA(4)=',IA(4)
        WRITE(14,30) 'IA(5)=',IA(5)
        WRITE(14,30) 'IA(6)=',IA(6)
        WRITE(14,30) 'IA(7)=',IA(7)
        WRITE(14,30) 'IA(8)=',IA(8)
        WRITE(14,30) 'IA(9)=',IA(9)
 30       FORMAT(A6,1I10)
        CLOSE(UNIT=14)
        END
        INCLUDE 'mean.f'
        INCLUDE 'simulAR.f'
        INCLUDE 'ARpsi.f'
        INCLUDE 'shock.f'
        INCLUDE 'rskew.f'
        INCLUDE 'rlap.f'
        INCLUDE 'rpare.f'
        INCLUDE 'rspare.f'
        INCLUDE 'rSU.f'
        INCLUDE 'rTU.f'
        INCLUDE 'rSC.f'
        INCLUDE 'rLC.f'
        INCLUDE 'rSB.f'
        INCLUDE 'rS.f'
        INCLUDE 'H1etoile.f'
        INCLUDE 'H2etoile.f'
        INCLUDE 'H3etoile.f'
        INCLUDE 'H4etoile.f'
        INCLUDE 'H5etoile.f'
        INCLUDE 'H6etoile.f'
        INCLUDE 'H7etoile.f'
        INCLUDE 'H8etoile.f'
        INCLUDE 'H9etoile.f'
        INCLUDE 'H10etoile.f'
        INCLUDE 'H1isa.f'
        INCLUDE 'H2isa.f'
        INCLUDE 'H3isa.f'
        INCLUDE 'H4isa.f'
        INCLUDE 'H5isa.f'
        INCLUDE 'H6isa.f'
        INCLUDE 'H7isa.f'
        INCLUDE 'H8isa.f'
        INCLUDE 'H9isa.f'
        INCLUDE 'H10isa.f'
        INCLUDE 'H1.f'
        INCLUDE 'H2.f'
```

```
      INCLUDE 'H3.f'
      INCLUDE 'H4.f'
      INCLUDE 'H5.f'
      INCLUDE 'H6.f'
      INCLUDE 'H7.f'
      INCLUDE 'H8.f'
      INCLUDE 'H9.f'
      INCLUDE 'H10.f'
      INCLUDE 'qnorm.f'
      INCLUDE 'pnorm.f'
      INCLUDE 'min.f'
      INCLUDE 'max.f'
      INCLUDE 'var.f'
      INCLUDE 'creerdat_AR.f'
      INCLUDE 'calcstat.f'
```

Programme big_prog_MA1.f

```
c Idee de ce  programme:
c Etape1)
c On cree un fichier de donnees 'data.txt' en utilisant certaines valeurs de
     parametres
c a lire dans un fichier d'entree 'ftemp' cree par le programme test.f a l'aide du
c fichier 'paramentree', on stocke les resultats autres que
c les donnees dans un fichier 'param.i'
c Etape 2)
c ensuite on utilise le fichier 'data.txt'
c avec le programme calcstat pour creer le fichier des statistiques 'resultat.i'
c Etape 3)
c Ensuite, on efface le fichier 'data.txt'
      PROGRAM main
c     Les ** indiquent les endroits ou des changements peuvent etre necessaires
c     ------------------------------------------------------------------------
c     DEBUT DE DECLARATION DES VARIABLES
c     ------------------------------------------------------------------------
c     Si isa=.TRUE. (ie MEAN=.TRUE.) on prend les polynomes modifies avec les ak  **
c     Si isa=.FALSE. (ie MEAN=.FALSE.) on prend les polynomes modifies sans les ak
      LOGICAL isa
      PARAMETER(isa=.FALSE.)
c-------------------------------------
c     parametres calcstat
c     Niveau 1 du test                                          **
      DOUBLE PRECISION alpha
      PARAMETER(alpha=0.1)
c     Niveau 2 du test                                          **
      DOUBLE PRECISION alpha2
      PARAMETER(alpha2=0.05)
c     Quantiles avec alpha
      DOUBLE PRECISION  khi(10)
c     Quantiles avec alpha2
      DOUBLE PRECISION  khi2(10)
c     Quantiles Ledwi1 et Ledwi2 avec alpha                          **
      DOUBLE PRECISION QuLed1, QuLed2
c     avec T=50
c      PARAMETER(QuLed1=3.69,QuLed2=5.466)
c     avec T=100
c      PARAMETER(QuLed1=3.275,QuLed2=5.26)
c     avec T=200
      PARAMETER(QuLed1=3.057,QuLed2=5.043)
c     Quantiles Ledwi1 et Ledwi2 avec alpha2                          **
      DOUBLE PRECISION QuLe12, QuLe22
c     avec T=50
c      PARAMETER(QuLe12=5.41,QuLe22=7.14)
c     avec T=100
c      PARAMETER(QuLe12=5.20,QuLe22=6.97)
```

```
c     avec T=200
      PARAMETER(QuLe12=4.751,QuLe22=6.796)
c     Quantiles Brockwell et Davis et Anderson Darling et JB avec alpha      **
      DOUBLE PRECISION QuBD, QuAD, QuJB
c     avec T=50
c      PARAMETER(QuBD=0.920,QuAD=1.743,QuJB=4.61)
c     avec T=100
c      PARAMETER(QuBD=0.958,QuAD=1.743,QuJB=4.61)
c     avec T=200
      PARAMETER(QuBD=0.978,QuAD=1.743,QuJB=4.61)
c     Quantiles Brockwell et Davis et Anderson Darling et JB avec alpha2      **
      DOUBLE PRECISION QuBD2, QuAD2, QuJB2
c     avec T=50
c      PARAMETER(QuBD2=0.899,QuAD2=2.308,QuJB2=5.99)
c      avec T=100
c      PARAMETER(QuBD2=0.947,QuAD2=2.308,QuJB2=5.99)
c     avec T=200
      PARAMETER(QuBD2=0.973,QuAD2=2.308,QuJB2=5.99)
c     Nombre de polynomes                                          **
      INTEGER Kp
      PARAMETER(Kp=10)
c     ordre du modele AR
      INTEGER p
      PARAMETER(p=0)
c     ordre du modele MA
      INTEGER q
      PARAMETER(q=2)
c     nombre d'observations dans mon echantillon                          **
c     Si on change la valeur de nT, il faut aller modifier
c     la valeur dans FORMAT a la fin du programme creerdat_ARMA.f
      INTEGER nT
      PARAMETER(nT=200)
      INTEGER n,m
      PARAMETER(m=nT+1)
c     n=nombre de lignes au maximum dans le fichier de donnees          **
c     cela correspond a nbcle s'il n'y a pas eu d'erreurs
c     dans tous les cas mettre ici n=nbcle=nombre d'echantillons souhaites
      PARAMETER(n=10000)
      CHARACTER filer*8, filew*21                                       **
      DOUBLE PRECISION valeur(n,m)
      DOUBLE PRECISION sch2vc(n)
      DOUBLE PRECISION epscha(n,nT)
      INTEGER compt(Kp)
      INTEGER compt2(Kp)
      DOUBLE PRECISION statn(n,Kp)
      INTEGER KoLed1(n)
      INTEGER KoLed2(n)
      DOUBLE PRECISION Ledwi1(Kp)
      DOUBLE PRECISION Ledwi2(Kp-1)
      DOUBLE PRECISION snLed1(n)
      DOUBLE PRECISION snLed2(n)
      DOUBLE PRECISION stanBD(n)
      DOUBLE PRECISION stanAD(n), stanJB(n)
      DOUBLE PRECISION Z(nT)
      DOUBLE PRECISION D(nT)
      DOUBLE PRECISION Davant(nT), E2(nT)
      DOUBLE PRECISION U(nT)
      DOUBLE PRECISION hUetoi(nT,Kp)
      DOUBLE PRECISION Uavant(nT)
      DOUBLE PRECISION hKetoi(Kp)
      DOUBLE PRECISION hU(nT,Kp)
      DOUBLE PRECISION hK(Kp)
      DOUBLE PRECISION vecteu(Kp)
      DOUBLE PRECISION vectoi(Kp)
      DOUBLE PRECISION stat(Kp)
```


```
      DOUBLE PRECISION Zavant(nT)
      DOUBLE PRECISION BDa(nT)
      DOUBLE PRECISION BDb(nT)
      DOUBLE PRECISION BDc(nT)
      DOUBLE PRECISION BDd(nT)
      DOUBLE PRECISION Davan2(nT)
      DOUBLE PRECISION ADa(nT)
      DOUBLE PRECISION ADb(nT)
      DOUBLE PRECISION Puiss(Kp)
      DOUBLE PRECISION Puiss2(Kp)
      DOUBLE PRECISION snsort(n,Kp)
      DOUBLE PRECISION snsorv(n)
      DOUBLE PRECISION Qucalc(Kp)
      DOUBLE PRECISION snLe1s(n)
      DOUBLE PRECISION snLe2s(n)
      DOUBLE PRECISION snBDso(n)
      DOUBLE PRECISION snADso(n), snJBso(n)
c-----------------------------------------------------------------
c     parametres creerdat
c     Nom du fichier de donnees en sortie                          **
      CHARACTER dataf*8
c     Nom du fichier de parametres en sortie                       **
      CHARACTER paramf*17
c     Nombre d'echantillons souhaites
      INTEGER nbcle,nbclea
c     rang d'arret dans la random shock method de Burn             **
      INTEGER marret
      PARAMETER(marret=200)
c     Induction period dans la methode de Burn                     **
      INTEGER Mind
      PARAMETER(Mind=200)
c     ecart-type des erreurs                                       **
      DOUBLE PRECISION sigma
      PARAMETER(sigma=1)
c     moyenne dans mon modele ARMA                                 **
      DOUBLE PRECISION mu
      PARAMETER(mu=0)
c     parametre de la khi-deux                                     **
      INTEGER df1
      PARAMETER(df1=2)
c     parametre de la student                                      **
      INTEGER df2
      PARAMETER(df2=5)
c     parametre de la skew-normale                                 **
      DOUBLE PRECISION lambda
      PARAMETER(lambda=2.0)
      DOUBLE PRECISION loi5b, loi5k, loi6p, loi6q
      DOUBLE PRECISION loi7g, loi7d, loi8p, loi8q, loi9a, loi9b
      DOUBLE PRECISION loi10b, loi10l, loi11a, loi11k, loi13g, loi13d
      DOUBLE PRECISION loi14l, loi16p, loi16d, loi17p, loi17m
      DOUBLE PRECISION loi18g, loi18d, loi19a, loi19b
      PARAMETER(loi18g=1.0,loi18d=1.0, loi19a=1.1, loi19b=0.5)     **
      PARAMETER(loi16p=0.2,loi16d=5.0, loi17p=0.2,loi17m=3.0)      **
      PARAMETER(loi10b=0.2,loi10l=1, loi11a=2.0,loi11k=0.5)        **
      PARAMETER(loi13g=1.0,loi13d=1.0, loi14l=0.7)                 **
      PARAMETER(loi7g=0,loi7d=1, loi8p=2, loi8q=2, loi9a=0,loi9b=2) **
      PARAMETER(loi5b=1, loi5k=1.8, loi6p=4, loi6q=1)              **
c     loi des erreurs
c     si loi=0 : Normale(0,sigma^2)
c     si loi=1 : Khi2 centree (df1)
c     si loi=2 : Student (df2)
c     si loi=3 : Skew-Normale(lambda)
c     si loi=4 : Laplace
c     si loi=5 : Weibull(b,k)
c     si loi=6 : Gamma(p,q)
```

```
c     si loi=7 : Log-Normale(g,d)
c     si loi=8 : Beta(p,q)
c     si loi=9 : Uniform(a,b)
c     si loi=10 : Shifted exp (l,b)
c     si loi=11 : Pareto(a,k)
c     si loi=12 : Shifted Pareto
c     si loi=13 : SU(g,d)
c     si loi=14 : TU(l)
c     si loi=15 : Logistic
c     si loi=16 : SC(p,d)
c     si loi=17 : LC(p,m)
c     si loi=18 : SB(g,d)
c     si loi=19 : S(a,b)
      INTEGER loi
c     rm=max(p,q)
      INTEGER rm
      PARAMETER(rm=2)
c     the autoregressive coefficients of the model
      DOUBLE PRECISION phi(2)
c     the moving-average coefficients of the model
      DOUBLE PRECISION teta(2)
      DOUBLE PRECISION tetach(q)
c     vecteur des donnees cree
      DOUBLE PRECISION donees(nT)
c Parametres pour G05EGF: simulation
c     the autoregressive coefficients of the model=phi     Input
      DOUBLE PRECISION A(1)
c     the moving-average coefficients of the model=teta     Input
      DOUBLE PRECISION B(q+1)
c     le vecteur des innovations     Output
      DOUBLE PRECISION R(nT)
c Parametres pour G13DCF: estimation
c     the number of initial parameter estimates   Input                    **
c     mettre p+q+1 si MEAN=.TRUE. , p+q si MEAN=.FALSE.
      INTEGER NPAR
      PARAMETER(NPAR=p+q)
c     first dimension of the array CM       Input                        **
c     mettre p+q+1 si MEAN=.TRUE. , p+q si MEAN=.FALSE.
      INTEGER ICM
      PARAMETER(ICM=p+q)
c     Workspace
      INTEGER IW(NPAR+rm+3)
c     initial parameter estimates    Input/Output
      DOUBLE PRECISION PAR(NPAR)
c     W(i,t) must be set equal to the observation at time t    Input
c     of the ith series
      DOUBLE PRECISION W(1,nT)
c     the accuracy to which the solution in PAR and QQ is required    Input      **
      DOUBLE PRECISION CGETOL
      PARAMETER(CGETOL=0.0001)
c     residual at time t for series i, for i = 1,2,...,k     Output
      DOUBLE PRECISION V(1,nT)
c     estimated first derivative of the log-likelihood function    Output
      DOUBLE PRECISION G(NPAR)
c     estimate of the correlation coefficient between the ith and    Output
c     jth elements in the PAR array
      DOUBLE PRECISION CM(ICM,NPAR)
c     Workspace
      DOUBLE PRECISION WORK((5+3*nT+2*rm)+(2*p+q+2*rm*(rm+2)+9)+
     +                    (NPAR+1)*(5*(NPAR+1)+29)/2+(rm+1)**2)
c     MEAN must be set to .TRUE. if components of mu are to    Input           **
c     be estimated and .FALSE. if all elements of mu are to be taken as zero
      LOGICAL MEAN
      PARAMETER(MEAN=.FALSE.)
c     PARHLD(i) must be set to .TRUE., if PAR(i) is to be      Input
```


```
c     held constant at its input value and .FALSE., if PAR(i) is a
c     free parameter, for i = 1,2,...,NPAR.
      LOGICAL PARHLD(NPAR)
c     EXACT must be set equal to .TRUE. if the user wishes    Input              **
c     the routine to compute exact maximum likelihood estimates.
c     EXACT must be set equal to .FALSE. if only conditional
c     likelihood estimates are required.
c     voir EXACT plus bas
      LOGICAL EXACT
c     utile dans simMA.f
      DOUBLE PRECISION YtMn(Mind+nT)
c     utile dans simMA.f
      DOUBLE PRECISION Atnq(nT+Mind+q)
c     Esperances des lois SU, SB et S                                          **
      DOUBLE PRECISION EspSU,EspSB,EspS
      PARAMETER(EspSU=0, EspSB=0, EspS=0)
      CHARACTER*13 efface
      DOUBLE PRECISION temp(4)
      INTEGER j
      CHARACTER*5 ftemp
      CHARACTER*1 INTCH1
      CHARACTER*2 INTCH2
      CHARACTER*3 INTCH3
      CHARACTER*4 INTCH4
      DOUBLE PRECISION matric(n,18)
      INTEGER dnT
      PARAMETER(dnT=Kp)
      DOUBLE PRECISION QLed1u,QLe12u
      PARAMETER(QLed1u=QuLed1,QLe12u=QuLe12)
      DOUBLE PRECISION QLed2u,QLe22u
      PARAMETER(QLed2u=QuLed2,QLe22u=QuLe22)
      DOUBLE PRECISION QBDu,QBD2u
      PARAMETER(QBDu=QuBD,QBD2u=QuBD2)
      DOUBLE PRECISION QADu,QAD2u
      PARAMETER(QADu=QuAD,QAD2u=QuAD2)
      DOUBLE PRECISION QJBu,QJB2u
      PARAMETER(QJBu=QuJB,QJB2u=QuJB2)
c     the mean of the time series      Input
      DOUBLE PRECISION E
      PARAMETER(E=mu)
c     the number of autoregressive coefficients supplied      Input
      INTEGER NA
      PARAMETER(NA=p)
c     the number of moving-average coefficients supplied      Input
      INTEGER NB
      PARAMETER(NB=q+1)
c     the dimension of the array R: vecteur des innovations      Input
      INTEGER NR
      PARAMETER(NR=10)
c     the number of observed time series, k  (chez moi k=1)  Input
      INTEGER K
      PARAMETER(K=1)
c     the number of observations in each time series, n (chez moi=nT)   Input
      INTEGER N2
      PARAMETER(N2=nT)
c     the number of AR parameter matrices, p     Input
      INTEGER IP
      PARAMETER(IP=p)
c     the number of MA parameter matrices, q    Input
      INTEGER IQ
      PARAMETER(IQ=q)
c     the first dimension of the arrays QQ, W and V     Input
      INTEGER IK
      PARAMETER(IK=1)
c     the maximum number of likelihood evaluations to be    Input
```

```
c     permitted by the search procedure
      INTEGER MAXCAL
      PARAMETER(MAXCAL=40*NPAR*(NPAR+5))
c     which quantities are to be printed     Input
      INTEGER ISHOW
      PARAMETER(ISHOW=0)
c     dimension of the array WORK     Input
      INTEGER LWORK
      PARAMETER(LWORK=(5+3*(nT)+2*rm)+(2*p+q+2*rm*(rm+2)+9)+
     +                (NPAR+1)*(5*(NPAR+1)+29)/2+(rm+1)**2)
c     dimension of the array IW     Input
      INTEGER LIW
      PARAMETER(LIW=NPAR+rm+3)
c     QQ(i,j) must be set equal to an initial estimate of     Input/Output
c     the (i,j)th element of the covariance matrix of the residual series
      DOUBLE PRECISION QQ(IK,K)
      INTEGER NI, NX, IFAIL
      PARAMETER(NX=4,NI=9)
      INTEGER IA(NI)
      DOUBLE PRECISION XA(NX)
c     seed=1 (non-repeatable sequence) ou seed=0 (repeatable sequence)
     **
      INTEGER seed
      PARAMETER(seed=0)
c-----------------------------------------------------------------
c Fin de declaration des variables
c-----------------------------------------------------------------
c G05CBF Initialise random number generating routines to give repeatable sequence
c G05CCF Initialise random number generating routines to give non-repeatable sequence
c G05CFF Save state of random number generating routines
c G05CGF Restore state of random number generating routines
      IF (seed .EQ. 1) THEN
      CALL G05CCF
      ENDIF
      IF (seed .EQ. 0) THEN
      CALL G05CBF(0)
      ENDIF
      IFAIL=0
c     Rajouter ici, si voulu, les IA et les XA a prendre a la fin du fichier para.ijkl
     :          **
c     A (de)commenter si necessaire (si on rajoute les IA et XA, on decommente)
c      CALL G05CGF(IA,NI,XA,NX,IFAIL)
      CALL G05CFF(IA,NI,XA,NX,IFAIL)
      EXACT=.TRUE.                                                    **
      nbcle=n
      efface='rm ./data.txt'
      ftemp='ftemp'
c     On va lire les parametres necessaires dans ftemp
c     temp va contenir sur chaque ligne: j, loi, teta(1), teta(2)
      OPEN(UNIT=15, FILE=ftemp , STATUS='OLD' )
      READ(15,*,END=10) (temp(j),j=1,4)
      CLOSE(15)
 10   j=temp(1)
      loi=temp(2)
      teta(1)=temp(3)
      teta(2)=temp(4)
      phi(1)=0.0
      phi(2)=0.0
c     On sauvegarde la valeur de nbcle
      nbclea=nbcle
c     Nom du fichier de donnees en sortie                               **
      dataf='data.txt'
c     Permet de convertir l'entier j en chaine de caracteres
      IF (j/10 .LT. 1) THEN
         WRITE(INTCH1, '(I1)') j
```

```
c     Nom du fichier de parametres en sortie                          **
         paramf='./SIMUL/para.'//'000'//INTCH1
         filew='./SIMUL/resultat.'//'000'//INTCH1
      ENDIF
      IF ((j/10 .EQ. 1) .OR. ((j/10 .GT. 1) .AND. (j/10 .LT. 10))) THEN
         WRITE(INTCH2, '(I2)') j
c     Nom du fichier de parametres en sortie                          **
         paramf='./SIMUL/para.'//'00'//INTCH2
         filew='./SIMUL/resultat.'//'00'//INTCH2
      ENDIF
      IF ((j/10 .EQ. 10) .OR. ((j/10 .GT. 10) .AND. (j/10 .LT. 100)))
     + THEN
         WRITE(INTCH3, '(I3)') j
c     Nom du fichier de parametres en sortie                          **
         paramf='./SIMUL/para.'//'0'//INTCH3
         filew='./SIMUL/resultat.'//'0'//INTCH3
      ENDIF
      IF ((j/10 .EQ. 100) .OR. ((j/10 .GT. 100) .AND. (j/10 .LT. 1000)))
     + THEN
         WRITE(INTCH4, '(I4)') j
c     Nom du fichier de parametres en sortie                          **
         paramf='./SIMUL/para.'//INTCH4
         filew='./SIMUL/resultat.'//INTCH4
      ENDIF
      IF (loi .EQ. 0) THEN
c     Quantiles de la loi du Khi2 (K=10;alpha=0.1)                  **
      khi(1)=2.71
      khi(2)=4.61
      khi(3)=6.25
      khi(4)=7.78
      khi(5)=9.24
      khi(6)=10.64
      khi(7)=12.02
      khi(8)=13.36
      khi(9)=14.68
      khi(10)=15.99
c     Quantiles de la loi du Khi2 (K=10;alpha=0.05)                  **
      khi2(1)=3.84
      khi2(2)=5.99
      khi2(3)=7.81
      khi2(4)=9.49
      khi2(5)=11.07
      khi2(6)=12.59
      khi2(7)=14.07
      khi2(8)=15.51
      khi2(9)=16.92
      khi2(10)=18.31
      ENDIF
      IF (loi .NE. 0) THEN
c     Quantiles utilises pour le calcul de la stat(vrais quantiles calcules sous H0)
     **
c     avec alpha=0.1
      khi(1)=2.71
      khi(2)=4.61
      khi(3)=6.25
      khi(4)=7.78
      khi(5)=9.24
      khi(6)=10.64
      khi(7)=12.02
      khi(8)=13.36
      khi(9)=14.68
      khi(10)=15.99
c     Quantiles utilises pour le calcul de la stat(vrais quantiles calcules sous H0)
     **
c     avec alpha=0.05
```

```
      khi2(1)=3.84
      khi2(2)=5.99
      khi2(3)=7.81
      khi2(4)=9.49
      khi2(5)=11.07
      khi2(6)=12.59
      khi2(7)=14.07
      khi2(8)=15.51
      khi2(9)=16.92
      khi2(10)=18.31
      ENDIF
c     On peut aussi changer le nom du fichier en entree (data.txt) et      **
c     le nom du fichier en sortie (resultat.txt)
      filer='data.txt'
c-------------------------------------------------------------------
c     ------------------------------------------------------------------------
c     DEBUT DU PROGRAMME
c     ------------------------------------------------------------------------
      CALL cdatMA(dataf,paramf,nbcle,marret,Mind,sigma,mu,df1,df2,
     + lambda,loi5b,loi5k,loi6p,loi6q,loi7g,loi7d,loi8p,loi8q,loi9a,
     +loi9b,loi10b,loi10l,loi11a,loi11k,loi13g,loi13d,loi14l,loi16p,
     +loi16d,loi17p,loi17m,loi18g,loi18d,loi19a,loi19b,loi,p,q,rm,nT,
     + teta,tetach,donees,A,
     + B,R,NPAR,ICM,IW,PAR,W,CGETOL,V,G,CM,WORK,MEAN,PARHLD,EXACT,
     + YtMn,Atnq,EspSU,EspSB,EspS,
     +E,NA,NB,NR,K,N2,IP,IQ,IK,MAXCAL,ISHOW,LWORK,LIW,QQ)
c Il faut faire modifier par cdARMA la valeur nbcle en entree pour lui faire sortir
c une nouvelle valeur nbcle qui est le nombre de bons echantillons conserves
c Ensuite, il faut utiliser cette valeur dans calcstat, comme etant la vraie valeur
c du nombre d'echantillons disponibles, la valeur de n que l'on a mise etant
c la valeur maximale possible de bons echantillons, donc les differentes matrices
     initialisees
c a l'aide de la valeur de n sont plus grandes que necessaires (elles auraient du etre
c initialisees avec la valeur de nbcle renvoyee par cd ARMA mais ce n'est pas possible
c avec fortran77) et il faut en tenir compte; donc bien regarder partout dans le
c programme calcstat la ou il y a n: OK CA C'EST FAIT!!! A METTRE EN COMMENTAIRES
     QUELQUEPART ...
         CALL calcstat(isa,khi,khi2,QuLed1,QuLe12,QuLed2,QuLe22,QuBD,
     + QuBD2,QuAD,QuAD2,QuJB,QuJB2,alpha,alpha2,Kp,p,q,nT,phi,teta,n,
     + nbcle,m,filer,filew,valeur,sch2vc,epscha,compt,compt2,statn,
     + KoLed1,KoLed2,Ledwi1,Ledwi2,snLed1,snLed2,stanBD,stanAD,stanJB,Z,
     + D,Davant,E2,U,hUetoi,Uavant,hKetoi,hU,hK,vecteu,vectoi,stat,
     + Zavant,BDa,BDb,BDc,BDd,Davan2,ADa,ADb,Puiss,Puiss2,snsort,snsorv,
     + Qucalc,snLe1s,snLe2s,snBDso,snADso,snJBso,matric,paramf,loi,
     + dnT,QLed1u,QLe12u,QLed2u,QLe22u,QBDu,QBD2u,QADu,QAD2u,QJBu,QJB2u)
         nbcle=nbclea
c     FAIRE ATTENTION:
c     voir si mes programmes ne modifient pas certains de leurs parametres en entree
c     ce qui pourrait ammener des erreurs au cours de la boucle que je vais faire dans
     le programme test.f
c     pour les differentes etapes 1), 2) et 3)
       CALL SYSTEM(efface)
c     Sauvegarde du seed
      OPEN(UNIT=14, FILE=paramf , STATUS='OLD', ACCESS='append' )
      WRITE(14,*) 'Sauvegarde du seed:'
      WRITE(14,*) 'XA:'
      WRITE(14,15) 'XA(1)=',XA(1)
      WRITE(14,15) 'XA(2)=',XA(2)
      WRITE(14,15) 'XA(3)=',XA(3)
      WRITE(14,15) 'XA(4)=',XA(4)
 15   FORMAT(A6,1F20.15)
      WRITE(14,*) 'IA:'
      WRITE(14,30) 'IA(1)=',IA(1)
      WRITE(14,30) 'IA(2)=',IA(2)
      WRITE(14,30) 'IA(3)=',IA(3)
```

```
      WRITE(14,30) 'IA(4)=',IA(4)
      WRITE(14,30) 'IA(5)=',IA(5)
      WRITE(14,30) 'IA(6)=',IA(6)
      WRITE(14,30) 'IA(7)=',IA(7)
      WRITE(14,30) 'IA(8)=',IA(8)
      WRITE(14,30) 'IA(9)=',IA(9)
30       FORMAT(A6,1I10)
      CLOSE(UNIT=14)
      END
      INCLUDE 'mean.f'
      INCLUDE 'simulMA.f'
      INCLUDE 'rskew.f'
      INCLUDE 'rlap.f'
      INCLUDE 'rpare.f'
      INCLUDE 'rspare.f'
      INCLUDE 'rSU.f'
      INCLUDE 'rTU.f'
      INCLUDE 'rSC.f'
      INCLUDE 'rLC.f'
      INCLUDE 'rSB.f'
      INCLUDE 'rS.f'
      INCLUDE 'H1etoile.f'
      INCLUDE 'H2etoile.f'
      INCLUDE 'H3etoile.f'
      INCLUDE 'H4etoile.f'
      INCLUDE 'H5etoile.f'
      INCLUDE 'H6etoile.f'
      INCLUDE 'H7etoile.f'
      INCLUDE 'H8etoile.f'
      INCLUDE 'H9etoile.f'
      INCLUDE 'H10etoile.f'
      INCLUDE 'H1isa.f'
      INCLUDE 'H2isa.f'
      INCLUDE 'H3isa.f'
      INCLUDE 'H4isa.f'
      INCLUDE 'H5isa.f'
      INCLUDE 'H6isa.f'
      INCLUDE 'H7isa.f'
      INCLUDE 'H8isa.f'
      INCLUDE 'H9isa.f'
      INCLUDE 'H10isa.f'
      INCLUDE 'H1.f'
      INCLUDE 'H2.f'
      INCLUDE 'H3.f'
      INCLUDE 'H4.f'
      INCLUDE 'H5.f'
      INCLUDE 'H6.f'
      INCLUDE 'H7.f'
      INCLUDE 'H8.f'
      INCLUDE 'H9.f'
      INCLUDE 'H10.f'
      INCLUDE 'qnorm.f'
      INCLUDE 'pnorm.f'
      INCLUDE 'min.f'
      INCLUDE 'max.f'
      INCLUDE 'var.f'
      INCLUDE 'creerdat_MA.f'
      INCLUDE 'calcstat.f'
```

Programme big_prog_ARMA11.f

```
c Idee de ce  programme:
c Etape1)
c On cree un fichier de donnees 'data.txt' en utilisant certaines valeurs de
     parametres
```


```
c a lire dans un fichier d'entree 'ftemp' cree par le programme test.f a l'aide du
c fichier 'paramentree', on stocke les resultats autres que
c les donnees dans un fichier 'param.i'
c Etape 2)
c ensuite on utilise le fichier 'data.txt'
c avec le programme calcstat pour creer le fichier des statistiques 'resultat.i'
c Etape 3)
c Ensuite, on efface le fichier 'data.txt'
      PROGRAM main
c     Les ** indiquent les endroits ou des changements peuvent etre necessaires
c     ------------------------------------------------------------------------
c     DEBUT DE DECLARATION DES VARIABLES
c     ------------------------------------------------------------------------
c     Si isa=.TRUE. (ie MEAN=.TRUE.) on prend les polynomes modifies avec les ak  **
c     Si isa=.FALSE. (ie MEAN=.FALSE.) on prend les polynomes modifies sans les ak
      LOGICAL isa
      PARAMETER(isa=.FALSE.)
c-------------------------------------
c     parametres calcstat
c     Niveau 1 du test                                          **
      DOUBLE PRECISION alpha
      PARAMETER(alpha=0.1)
c     Niveau 2 du test                                          **
      DOUBLE PRECISION alpha2
      PARAMETER(alpha2=0.05)
c     Quantiles avec alpha
      DOUBLE PRECISION  khi(10)
c     Quantiles avec alpha2
      DOUBLE PRECISION  khi2(10)
c     Quantiles Ledwi1 et Ledwi2 avec alpha                          **
      DOUBLE PRECISION QuLed1, QuLed2
c     avec T=50
      PARAMETER(QuLed1=3.69,QuLed2=5.466)
c     avec T=100
c      PARAMETER(QuLed1=3.275,QuLed2=5.26)
c     avec T=200
c      PARAMETER(QuLed1=3.057,QuLed2=5.043)
c     Quantiles Ledwi1 et Ledwi2 avec alpha2                          **
      DOUBLE PRECISION QuLe12, QuLe22
c     avec T=50
      PARAMETER(QuLe12=5.41,QuLe22=7.14)
c     avec T=100
c      PARAMETER(QuLe12=5.20,QuLe22=6.97)
c     avec T=200
c      PARAMETER(QuLe12=4.751,QuLe22=6.796)
c     Quantiles Brockwell et Davis et Anderson Darling et JB avec alpha       **
      DOUBLE PRECISION QuBD, QuAD, QuJB
c     avec T=50
      PARAMETER(QuBD=0.920,QuAD=1.743,QuJB=4.61)
c     avec T=100
c      PARAMETER(QuBD=0.958,QuAD=1.743,QuJB=4.61)
c     avec T=200
c      PARAMETER(QuBD=0.978,QuAD=1.743,QuJB=4.61)
c     Quantiles Brockwell et Davis et Anderson Darling et JB avec alpha2       **
      DOUBLE PRECISION QuBD2, QuAD2, QuJB2
c     avec T=50
      PARAMETER(QuBD2=0.899,QuAD2=2.308,QuJB2=5.99)
c      avec T=100
c      PARAMETER(QuBD2=0.947,QuAD2=2.308,QuJB2=5.99)
c     avec T=200
c      PARAMETER(QuBD2=0.973,QuAD2=2.308,QuJB2=5.99)
c     Nombre de polynomes                                       **
      INTEGER Kp
      PARAMETER(Kp=10)
c     ordre du modele AR
```

```
      INTEGER p
      PARAMETER(p=1)
c     ordre du modele MA
      INTEGER q
      PARAMETER(q=1)
c     nombre d'observations dans mon echantillon                          **
c     Si on change la valeur de nT, il faut aller modifier
c     la valeur dans FORMAT a la fin du programme creerdat_ARMA.f
      INTEGER nT
      PARAMETER(nT=50)
      INTEGER n,m
      PARAMETER(m=nT+1)
c     n=nombre de lignes au maximum dans le fichier de donnees          **
c     cela correspond a nbcle s'il n'y a pas eu d'erreurs
c     dans tous les cas mettre ici n=nbcle=nombre d'echantillons souhaites
      PARAMETER(n=10000)
      CHARACTER filer*8, filew*21                                       **
      DOUBLE PRECISION valeur(n,m)
      DOUBLE PRECISION sch2vc(n)
      DOUBLE PRECISION epscha(n,nT)
      INTEGER compt(Kp)
      INTEGER compt2(Kp)
      DOUBLE PRECISION statn(n,Kp)
      INTEGER KoLed1(n)
      INTEGER KoLed2(n)
      DOUBLE PRECISION Ledwi1(Kp)
      DOUBLE PRECISION Ledwi2(Kp-1)
      DOUBLE PRECISION snLed1(n)
      DOUBLE PRECISION snLed2(n)
      DOUBLE PRECISION stanBD(n)
      DOUBLE PRECISION stanAD(n), stanJB(n)
      DOUBLE PRECISION Z(nT)
      DOUBLE PRECISION D(nT)
      DOUBLE PRECISION Davant(nT), E2(nT)
      DOUBLE PRECISION U(nT)
      DOUBLE PRECISION hUetoi(nT,Kp)
      DOUBLE PRECISION Uavant(nT)
      DOUBLE PRECISION hKetoi(Kp)
      DOUBLE PRECISION hU(nT,Kp)
      DOUBLE PRECISION hK(Kp)
      DOUBLE PRECISION vecteu(Kp)
      DOUBLE PRECISION vectoi(Kp)
      DOUBLE PRECISION stat(Kp)
      DOUBLE PRECISION Zavant(nT)
      DOUBLE PRECISION BDa(nT)
      DOUBLE PRECISION BDb(nT)
      DOUBLE PRECISION BDc(nT)
      DOUBLE PRECISION BDd(nT)
      DOUBLE PRECISION Davan2(nT)
      DOUBLE PRECISION ADa(nT)
      DOUBLE PRECISION ADb(nT)
      DOUBLE PRECISION Puiss(Kp)
      DOUBLE PRECISION Puiss2(Kp)
      DOUBLE PRECISION snsort(n,Kp)
      DOUBLE PRECISION snsorv(n)
      DOUBLE PRECISION Qucalc(Kp)
      DOUBLE PRECISION snLe1s(n)
      DOUBLE PRECISION snLe2s(n)
      DOUBLE PRECISION snBDso(n)
      DOUBLE PRECISION snADso(n), snJBso(n)
c-------------------------------------------------------------------
c     parametres creerdat
c     Nom du fichier de donnees en sortie                               **
      CHARACTER dataf*8
c     Nom du fichier de parametres en sortie                            **
```

```
      CHARACTER paramf*17
c     Nombre d'echantillons souhaites
      INTEGER nbcle,nbclea
c     rang d'arret dans la random shock method de Burn                **
      INTEGER marret
      PARAMETER(marret=200)
c     Induction period dans la methode de Burn                        **
      INTEGER Mind
      PARAMETER(Mind=200)
c     ecart-type des erreurs                                          **
      DOUBLE PRECISION sigma
      PARAMETER(sigma=1)
c     moyenne dans mon modele ARMA                                    **
      DOUBLE PRECISION mu
      PARAMETER(mu=0)
c     parametre de la khi-deux                                        **
      INTEGER df1
      PARAMETER(df1=2)
c     parametre de la student                                         **
      INTEGER df2
      PARAMETER(df2=5)
c     parametre de la skew-normale                                    **
      DOUBLE PRECISION lambda
      PARAMETER(lambda=2.0)
      DOUBLE PRECISION loi5b, loi5k, loi6p, loi6q
      DOUBLE PRECISION loi7g, loi7d, loi8p, loi8q, loi9a, loi9b
      DOUBLE PRECISION loi10b, loi10l, loi11a, loi11k, loi13g, loi13d
      DOUBLE PRECISION loi14l, loi16p, loi16d, loi17p, loi17m
      DOUBLE PRECISION loi18g, loi18d, loi19a, loi19b
      PARAMETER(loi18g=1.0,loi18d=1.0, loi19a=1.1, loi19b=0.5)        **
      PARAMETER(loi16p=0.2,loi16d=5.0, loi17p=0.2,loi17m=3.0)         **
      PARAMETER(loi10b=0.2,loi10l=1, loi11a=2.0,loi11k=0.5)           **
      PARAMETER(loi13g=1.0,loi13d=1.0, loi14l=0.7)                    **
      PARAMETER(loi7g=0,loi7d=1, loi8p=2, loi8q=2, loi9a=0,loi9b=2)   **
      PARAMETER(loi5b=1, loi5k=1.8, loi6p=4, loi6q=1)                 **
c     loi des erreurs                                                 **
c     si loi=0 : Normale(0,sigma^2)
c     si loi=1 : Khi2 centree (df1)
c     si loi=2 : Student (df2)
c     si loi=3 : Skew-Normale(lambda)
c     si loi=4 : Laplace
c     si loi=5 : Weibull(b,k)
c     si loi=6 : Gamma(p,q)
c     si loi=7 : Log-Normale(g,d)
c     si loi=8 : Beta(p,q)
c     si loi=9 : Uniform(a,b)
c     si loi=10 : Shifted exp (l,b)
c     si loi=11 : Pareto(a,k)
c     si loi=12 : Shifted Pareto
c     si loi=13 : SU(g,d)
c     si loi=14 : TU(l)
c     si loi=15 : Logistic
c     si loi=16 : SC(p,d)
c     si loi=17 : LC(p,m)
c     si loi=18 : SB(g,d)
c     si loi=19 : S(a,b)
      INTEGER loi
c     rm=max(p,q)
      INTEGER rm
      PARAMETER(rm=1)
c     the autoregressive coefficients of the model
      DOUBLE PRECISION phi(2)
      DOUBLE PRECISION phich(p)
c     the moving-average coefficients of the model
      DOUBLE PRECISION teta(2)
```

```
      DOUBLE PRECISION tetach(q)
c     vecteur des donnees cree
      DOUBLE PRECISION donees(nT)
c Parametres pour G05EGF: simulation
c     the autoregressive coefficients of the model=phi     Input
      DOUBLE PRECISION A(p)
c     the moving-average coefficients of the model=teta     Input
      DOUBLE PRECISION B(q+1)
c     le vecteur des innovations     Output
      DOUBLE PRECISION R(nT)
c Parametres pour G13DCF: estimation
c     the number of initial parameter estimates   Input                    **
c     mettre p+q+1 si MEAN=.TRUE. , p+q si MEAN=.FALSE.
      INTEGER NPAR
      PARAMETER(NPAR=p+q)
c     first dimension of the array CM       Input                        **
c     mettre p+q+1 si MEAN=.TRUE. , p+q si MEAN=.FALSE.
      INTEGER ICM
      PARAMETER(ICM=p+q)
c     Workspace
      INTEGER IW(NPAR+rm+3)
c     initial parameter estimates     Input/Output
      DOUBLE PRECISION PAR(NPAR)
c     W(i,t) must be set equal to the observation at time t     Input
c     of the ith series
      DOUBLE PRECISION W(1,nT)
c     the accuracy to which the solution in PAR and QQ is required     Input     **
      DOUBLE PRECISION CGETOL
      PARAMETER(CGETOL=0.0001)
c     residual at time t for series i, for i = 1,2,...,k     Output
      DOUBLE PRECISION V(1,nT)
c     estimated first derivative of the log-likelihood function     Output
      DOUBLE PRECISION G(NPAR)
c     estimate of the correlation coefficient between the ith and     Output
c     jth elements in the PAR array
      DOUBLE PRECISION CM(ICM,NPAR)
c     Workspace
      DOUBLE PRECISION WORK((5+3*nT+2*rm)+(2*p+q+2*rm*(rm+2)+9)+
     +                    (NPAR+1)*(5*(NPAR+1)+29)/2+(rm+1)**2)
c     MEAN must be set to .TRUE. if components of mu are to     Input          **
c     be estimated and .FALSE. if all elements of mu are to be taken as zero
      LOGICAL MEAN
      PARAMETER(MEAN=.FALSE.)
c     PARHLD(i) must be set to .TRUE., if PAR(i) is to be       Input
c     held constant at its input value and .FALSE., if PAR(i) is a
c     free parameter, for i = 1,2,...,NPAR.
      LOGICAL PARHLD(NPAR)
c     EXACT must be set equal to .TRUE. if the user wishes     Input          **
c     the routine to compute exact maximum likelihood estimates.
c     EXACT must be set equal to .FALSE. if only conditional
c     likelihood estimates are required.
c     voir EXACT plus bas
      LOGICAL EXACT
c     Contient les p donnees initiales de la serie
      DOUBLE PRECISION Wtip(p)
c     marret+p innovations genere par shock.f
      DOUBLE PRECISION shocks(marret+p)
c     defini dans Burn, calcule par ARMpsi.f
      DOUBLE PRECISION psi(marret+1)
c     utile dans simARM.f de longueur marret
      DOUBLE PRECISION phi2(marret)
c     utile dans simARM.f de longueur marret
      DOUBLE PRECISION teta2(marret)
c     utile dans simARM.f
      DOUBLE PRECISION YtpMn(p+Mind+nT)
```

```
c      utile dans simARM.f
       DOUBLE PRECISION Atnq(nT+Mind+q)
c      Esperances des lois SU, SB et S                          **
       DOUBLE PRECISION EspSU,EspSB,EspS
       PARAMETER(EspSU=0, EspSB=0, EspS=0)
       CHARACTER*13 efface
       DOUBLE PRECISION temp(4)
       INTEGER j
       CHARACTER*5 ftemp
       CHARACTER*1 INTCH1
       CHARACTER*2 INTCH2
       CHARACTER*3 INTCH3
       CHARACTER*4 INTCH4
       DOUBLE PRECISION matric(n,18)
       INTEGER dnT
       PARAMETER(dnT=Kp)
       DOUBLE PRECISION QLed1u,QLe12u
       PARAMETER(QLed1u=QuLed1,QLe12u=QuLe12)
       DOUBLE PRECISION QLed2u,QLe22u
       PARAMETER(QLed2u=QuLed2,QLe22u=QuLe22)
       DOUBLE PRECISION QBDu,QBD2u
       PARAMETER(QBDu=QuBD,QBD2u=QuBD2)
       DOUBLE PRECISION QADu,QAD2u
       PARAMETER(QADu=QuAD,QAD2u=QuAD2)
       DOUBLE PRECISION QJBu,QJB2u
       PARAMETER(QJBu=QuJB,QJB2u=QuJB2)
c      the mean of the time series      Input
       DOUBLE PRECISION E
       PARAMETER(E=mu)
c      the number of autoregressive coefficients supplied      Input
       INTEGER NA
       PARAMETER(NA=p)
c      the number of moving-average coefficients supplied      Input
       INTEGER NB
       PARAMETER(NB=q+1)
c      the dimension of the array R: vecteur des innovations      Input
       INTEGER NR
       PARAMETER(NR=9)
c      the number of observed time series, k  (chez moi k=1)  Input
       INTEGER K
       PARAMETER(K=1)
c      the number of observations in each time series, n (chez moi=nT)   Input
       INTEGER N2
       PARAMETER(N2=nT)
c      the number of AR parameter matrices, p     Input
       INTEGER IP
       PARAMETER(IP=p)
c      the number of MA parameter matrices, q    Input
       INTEGER IQ
       PARAMETER(IQ=q)
c      the first dimension of the arrays QQ, W and V     Input
       INTEGER IK
       PARAMETER(IK=1)
c      the maximum number of likelihood evaluations to be     Input
c      permitted by the search procedure
       INTEGER MAXCAL
       PARAMETER(MAXCAL=40*NPAR*(NPAR+5))
c      which quantities are to be printed     Input
       INTEGER ISHOW
       PARAMETER(ISHOW=0)
c      dimension of the array WORK     Input
       INTEGER LWORK
       PARAMETER(LWORK=(5+3*(nT)+2*rm)+(2*p+q+2*rm*(rm+2)+9)+
      +                (NPAR+1)*(5*(NPAR+1)+29)/2+(rm+1)**2)
c      dimension of the array IW     Input
```

```
      INTEGER LIW
      PARAMETER(LIW=NPAR+rm+3)
c     QQ(i,j) must be set equal to an initial estimate of    Input/Output
c     the (i,j)th element of the covariance matrix of the residual series
      DOUBLE PRECISION QQ(IK,K)
      INTEGER NI, NX, IFAIL
      PARAMETER(NX=4,NI=9)
      INTEGER IA(NI)
      DOUBLE PRECISION XA(NX)
c     seed=1 (non-repeatable sequence) ou seed=0 (repeatable sequence)
     **
      INTEGER seed
      PARAMETER(seed=0)
c-----------------------------------------------------------------
c Fin de declaration des variables
c-----------------------------------------------------------------
c G05CBF Initialise random number generating routines to give repeatable sequence
c G05CCF Initialise random number generating routines to give non-repeatable sequence
c G05CFF Save state of random number generating routines
c G05CGF Restore state of random number generating routines
      IF (seed .EQ. 1) THEN
      CALL G05CCF
      ENDIF
      IF (seed .EQ. 0) THEN
      CALL G05CBF(0)
      ENDIF
      IFAIL=0
c     Rajouter ici, si voulu, les IA et les XA a prendre a la fin du fichier para.ijkl
     :         **
c     A (de)commenter si necessaire (si on rajoute les IA et XA, on decommente)
c      CALL G05CGF(IA,NI,XA,NX,IFAIL)
      CALL G05CFF(IA,NI,XA,NX,IFAIL)
      EXACT=.TRUE.                                                    **
      nbcle=n
      efface='rm ./data.txt'
      ftemp='ftemp'
c     On va lire les parametres necessaires dans ftemp
c     temp va contenir sur chaque ligne: j, loi, phi(1), teta(1)
      OPEN(UNIT=15, FILE=ftemp , STATUS='OLD' )
      READ(15,*,END=10) (temp(j),j=1,4)
      CLOSE(15)
 10   j=temp(1)
      loi=temp(2)
      phi(1)=temp(3)
      teta(1)=temp(4)
      phi(2)=0.0
      teta(2)=0.0
c     On sauvegarde la valeur de nbcle
      nbclea=nbcle
c     Nom du fichier de donnees en sortie                             **
      dataf='data.txt'
c     Permet de convertir l'entier j en chaine de caracteres
      IF (j/10 .LT. 1) THEN
         WRITE(INTCH1, '(I1)') j
c     Nom du fichier de parametres en sortie                          **
         paramf='./SIMUL/para.'//'000'//INTCH1
         filew='./SIMUL/resultat.'//'000'//INTCH1
      ENDIF
      IF ((j/10 .EQ. 1) .OR. ((j/10 .GT. 1) .AND. (j/10 .LT. 10))) THEN
         WRITE(INTCH2, '(I2)') j
c     Nom du fichier de parametres en sortie                          **
         paramf='./SIMUL/para.'//'00'//INTCH2
         filew='./SIMUL/resultat.'//'00'//INTCH2
      ENDIF
      IF ((j/10 .EQ. 10) .OR. ((j/10 .GT. 10) .AND. (j/10 .LT. 100)))
```

```
     + THEN
          WRITE(INTCH3, '(I3)') j
c     Nom du fichier de parametres en sortie                                **
          paramf='./SIMUL/para.'//'0'//INTCH3
          filew='./SIMUL/resultat.'//'0'//INTCH3
       ENDIF
       IF ((j/10 .EQ. 100) .OR. ((j/10 .GT. 100) .AND. (j/10 .LT. 1000)))
     + THEN
          WRITE(INTCH4, '(I4)') j
c     Nom du fichier de parametres en sortie                                **
          paramf='./SIMUL/para.'//INTCH4
          filew='./SIMUL/resultat.'//INTCH4
       ENDIF
       IF (loi .EQ. 0) THEN
c     Quantiles de la loi du Khi2 (K=10;alpha=0.1)                        **
       khi(1)=2.71
       khi(2)=4.61
       khi(3)=6.25
       khi(4)=7.78
       khi(5)=9.24
       khi(6)=10.64
       khi(7)=12.02
       khi(8)=13.36
       khi(9)=14.68
       khi(10)=15.99
c     Quantiles de la loi du Khi2 (K=10;alpha=0.05)                         **
       khi2(1)=3.84
       khi2(2)=5.99
       khi2(3)=7.81
       khi2(4)=9.49
       khi2(5)=11.07
       khi2(6)=12.59
       khi2(7)=14.07
       khi2(8)=15.51
       khi2(9)=16.92
       khi2(10)=18.31
       ENDIF
       IF (loi .NE. 0) THEN
c     Quantiles utilises pour le calcul de la stat(vrais quantiles calcules sous H0)
     **
c     avec alpha=0.1
       khi(1)=2.71
       khi(2)=4.61
       khi(3)=6.25
       khi(4)=7.78
       khi(5)=9.24
       khi(6)=10.64
       khi(7)=12.02
       khi(8)=13.36
       khi(9)=14.68
       khi(10)=15.99
c     Quantiles utilises pour le calcul de la stat(vrais quantiles calcules sous H0)
     **
c     avec alpha=0.05
       khi2(1)=3.84
       khi2(2)=5.99
       khi2(3)=7.81
       khi2(4)=9.49
       khi2(5)=11.07
       khi2(6)=12.59
       khi2(7)=14.07
       khi2(8)=15.51
       khi2(9)=16.92
       khi2(10)=18.31
       ENDIF
```


```
c      On peut aussi changer le nom du fichier en entree (data.txt) et      **
c      le nom du fichier en sortie (resultat.txt)
       filer='data.txt'
c-------------------------------------------------------------------
c     ------------------------------------------------------------------
c      DEBUT DU PROGRAMME
c     ------------------------------------------------------------------
       CALL cdARMA(dataf,paramf,nbcle,marret,Mind,sigma,mu,df1,df2,
      + lambda,loi5b,loi5k,loi6p,loi6q,loi7g,loi7d,loi8p,loi8q,loi9a,
      +loi9b,loi10b,loi10l,loi11a,loi11k,loi13g,loi13d,loi14l,loi16p,
      +loi16d,loi17p,loi17m,loi18g,loi18d,loi19a,loi19b,loi,p,q,rm,nT,
      + phi,phich,teta,tetach,donees,A,
      + B,R,NPAR,ICM,IW,PAR,W,CGETOL,V,G,CM,WORK,MEAN,PARHLD,EXACT,Wtip,
      + shocks,psi,phi2,teta2,YtpMn,Atnq,EspSU,EspSB,EspS,
      +E,NA,NB,NR,K,N2,IP,IQ,IK,MAXCAL,ISHOW,LWORK,LIW,QQ)
c Il faut faire modifier par cdARMA la valeur nbcle en entree pour lui faire sortir
c une nouvelle valeur nbcle qui est le nombre de bons echantillons conserves
c Ensuite, il faut utiliser cette valeur dans calcstat, comme etant la vraie valeur
c du nombre d'echantillons disponibles, la valeur de n que l'on a mise etant
c la valeur maximale possible de bons echantillons, donc les differentes matrices
     initialisees
c a l'aide de la valeur de n sont plus grandes que necessaires (elles auraient du etre
c initialisees avec la valeur de nbcle renvoyee par cd ARMA mais ce n'est pas possible
c avec fortran77) et il faut en tenir compte; donc bien regarder partout dans le
c programme calcstat la ou il y a n: OK CA C'EST FAIT!!! A METTRE EN COMMENTAIRES
     QUELQUEPART ...
          CALL calcstat(isa,khi,khi2,QuLed1,QuLe12,QuLed2,QuLe22,QuBD,
      + QuBD2,QuAD,QuAD2,QuJB,QuJB2,alpha,alpha2,Kp,p,q,nT,phi,teta,n,
      + nbcle,m,filer,filew,valeur,sch2vc,epscha,compt,compt2,statn,
      + KoLed1,KoLed2,Ledwi1,Ledwi2,snLed1,snLed2,stanBD,stanAD,stanJB,Z,
      + D,Davant,E2,U,hUetoi,Uavant,hKetoi,hU,hK,vecteu,vectoi,stat,
      + Zavant,BDa,BDb,BDc,BDd,Davan2,ADa,ADb,Puiss,Puiss2,snsort,snsorv,
      + Qucalc,snLe1s,snLe2s,snBDso,snADso,snJBso,matric,paramf,loi,
      + dnT,QLed1u,QLe12u,QLed2u,QLe22u,QBDu,QBD2u,QADu,QAD2u,QJBu,QJB2u)
          nbcle=nbclea
c      FAIRE ATTENTION:
c      voir si mes programmes ne modifient pas certains de leurs parametres en entree
c      ce qui pourrait ammener des erreurs au cours de la boucle que je vais faire dans
      le programme test.f
c      pour les differentes etapes 1), 2) et 3)
         CALL SYSTEM(efface)
c      Sauvegarde du seed
       OPEN(UNIT=14, FILE=paramf , STATUS='OLD', ACCESS='append' )
       WRITE(14,*) 'Sauvegarde du seed:'
       WRITE(14,*) 'XA:'
       WRITE(14,15) 'XA(1)=',XA(1)
       WRITE(14,15) 'XA(2)=',XA(2)
       WRITE(14,15) 'XA(3)=',XA(3)
       WRITE(14,15) 'XA(4)=',XA(4)
 15    FORMAT(A6,1F20.15)
       WRITE(14,*) 'IA:'
       WRITE(14,30) 'IA(1)=',IA(1)
       WRITE(14,30) 'IA(2)=',IA(2)
       WRITE(14,30) 'IA(3)=',IA(3)
       WRITE(14,30) 'IA(4)=',IA(4)
       WRITE(14,30) 'IA(5)=',IA(5)
       WRITE(14,30) 'IA(6)=',IA(6)
       WRITE(14,30) 'IA(7)=',IA(7)
       WRITE(14,30) 'IA(8)=',IA(8)
       WRITE(14,30) 'IA(9)=',IA(9)
 30       FORMAT(A6,1I10)
       CLOSE(UNIT=14)
       END
       INCLUDE 'mean.f'
       INCLUDE 'simulARMA.f'
```

```
      INCLUDE 'ARMpsi.f'
      INCLUDE 'shock.f'
      INCLUDE 'rskew.f'
      INCLUDE 'rlap.f'
      INCLUDE 'rpare.f'
      INCLUDE 'rspare.f'
      INCLUDE 'rSU.f'
      INCLUDE 'rTU.f'
      INCLUDE 'rSC.f'
      INCLUDE 'rLC.f'
      INCLUDE 'rSB.f'
      INCLUDE 'rS.f'
      INCLUDE 'H1etoile.f'
      INCLUDE 'H2etoile.f'
      INCLUDE 'H3etoile.f'
      INCLUDE 'H4etoile.f'
      INCLUDE 'H5etoile.f'
      INCLUDE 'H6etoile.f'
      INCLUDE 'H7etoile.f'
      INCLUDE 'H8etoile.f'
      INCLUDE 'H9etoile.f'
      INCLUDE 'H10etoile.f'
      INCLUDE 'H1isa.f'
      INCLUDE 'H2isa.f'
      INCLUDE 'H3isa.f'
      INCLUDE 'H4isa.f'
      INCLUDE 'H5isa.f'
      INCLUDE 'H6isa.f'
      INCLUDE 'H7isa.f'
      INCLUDE 'H8isa.f'
      INCLUDE 'H9isa.f'
      INCLUDE 'H10isa.f'
      INCLUDE 'H1.f'
      INCLUDE 'H2.f'
      INCLUDE 'H3.f'
      INCLUDE 'H4.f'
      INCLUDE 'H5.f'
      INCLUDE 'H6.f'
      INCLUDE 'H7.f'
      INCLUDE 'H8.f'
      INCLUDE 'H9.f'
      INCLUDE 'H10.f'
      INCLUDE 'qnorm.f'
      INCLUDE 'pnorm.f'
      INCLUDE 'min.f'
      INCLUDE 'max.f'
      INCLUDE 'var.f'
      INCLUDE 'creerdat_ARMA.f'
      INCLUDE 'calcstat.f'
```

Programme big_prog_ARMA12.f

```
c Idee de ce  programme:
c Etape1)
c On cree un fichier de donnees 'data.txt' en utilisant certaines valeurs de
     parametres
c a lire dans un fichier d'entree 'ftemp' cree par le programme test.f a l'aide du
c fichier 'paramentree', on stocke les resultats autres que
c les donnees dans un fichier 'param.i'
c Etape 2)
c ensuite on utilise le fichier 'data.txt'
c avec le programme calcstat pour creer le fichier des statistiques 'resultat.i'
c Etape 3)
c Ensuite, on efface le fichier 'data.txt'
      PROGRAM main
```

```
c     Les ** indiquent les endroits ou des changements peuvent etre necessaires
c     ------------------------------------------------------------------------
c     DEBUT DE DECLARATION DES VARIABLES
c     ------------------------------------------------------------------------
c     Si isa=.TRUE. (ie MEAN=.TRUE.) on prend les polynomes modifies avec les ak  **
c     Si isa=.FALSE. (ie MEAN=.FALSE.) on prend les polynomes modifies sans les ak
      LOGICAL isa
      PARAMETER(isa=.FALSE.)
c-------------------------------------
c     parametres calcstat
c     Niveau 1 du test                                          **
      DOUBLE PRECISION alpha
      PARAMETER(alpha=0.1)
c     Niveau 2 du test                                          **
      DOUBLE PRECISION alpha2
      PARAMETER(alpha2=0.05)
c     Quantiles avec alpha
      DOUBLE PRECISION  khi(10)
c     Quantiles avec alpha2
      DOUBLE PRECISION  khi2(10)
c     Quantiles Ledwi1 et Ledwi2 avec alpha                              **
      DOUBLE PRECISION QuLed1, QuLed2
c     avec T=50
c      PARAMETER(QuLed1=3.69,QuLed2=5.466)
c     avec T=100
      PARAMETER(QuLed1=3.275,QuLed2=5.26)
c     avec T=200
c      PARAMETER(QuLed1=3.057,QuLed2=5.043)
c     Quantiles Ledwi1 et Ledwi2 avec alpha2                              **
      DOUBLE PRECISION QuLe12, QuLe22
c     avec T=50
c      PARAMETER(QuLe12=5.41,QuLe22=7.14)
c     avec T=100
      PARAMETER(QuLe12=5.20,QuLe22=6.97)
c     avec T=200
c      PARAMETER(QuLe12=4.751,QuLe22=6.796)
c     Quantiles Brockwell et Davis et Anderson Darling et JB avec alpha       **
      DOUBLE PRECISION QuBD, QuAD, QuJB
c     avec T=50
c      PARAMETER(QuBD=0.920,QuAD=1.743,QuJB=4.61)
c     avec T=100
      PARAMETER(QuBD=0.958,QuAD=1.743,QuJB=4.61)
c     avec T=200
c      PARAMETER(QuBD=0.978,QuAD=1.743,QuJB=4.61)
c     Quantiles Brockwell et Davis et Anderson Darling et JB avec alpha2      **
      DOUBLE PRECISION QuBD2, QuAD2, QuJB2
c     avec T=50
c      PARAMETER(QuBD2=0.899,QuAD2=2.308,QuJB2=5.99)
c      avec T=100
      PARAMETER(QuBD2=0.947,QuAD2=2.308,QuJB2=5.99)
c     avec T=200
c      PARAMETER(QuBD2=0.973,QuAD2=2.308,QuJB2=5.99)
c     Nombre de polynomes                                       **
      INTEGER Kp
      PARAMETER(Kp=10)
c     ordre du modele AR
      INTEGER p
      PARAMETER(p=1)
c     ordre du modele MA
      INTEGER q
      PARAMETER(q=2)
c     nombre d'observations dans mon echantillon                          **
c     Si on change la valeur de nT, il faut aller modifier
c     la valeur dans FORMAT a la fin du programme creerdat_ARMA.f
      INTEGER nT
```

```
      PARAMETER(nT=100)
      INTEGER n,m
      PARAMETER(m=nT+1)
c     n=nombre de lignes au maximum dans le fichier de donnees          **
c     cela correspond a nbcle s'il n'y a pas eu d'erreurs
c     dans tous les cas mettre ici n=nbcle=nombre d'echantillons souhaites
      PARAMETER(n=10000)
      CHARACTER filer*8, filew*21                                       **
      DOUBLE PRECISION valeur(n,m)
      DOUBLE PRECISION sch2vc(n)
      DOUBLE PRECISION epscha(n,nT)
      INTEGER compt(Kp)
      INTEGER compt2(Kp)
      DOUBLE PRECISION statn(n,Kp)
      INTEGER KoLed1(n)
      INTEGER KoLed2(n)
      DOUBLE PRECISION Ledwi1(Kp)
      DOUBLE PRECISION Ledwi2(Kp-1)
      DOUBLE PRECISION snLed1(n)
      DOUBLE PRECISION snLed2(n)
      DOUBLE PRECISION stanBD(n)
      DOUBLE PRECISION stanAD(n), stanJB(n)
      DOUBLE PRECISION Z(nT)
      DOUBLE PRECISION D(nT)
      DOUBLE PRECISION Davant(nT), E2(nT)
      DOUBLE PRECISION U(nT)
      DOUBLE PRECISION hUetoi(nT,Kp)
      DOUBLE PRECISION Uavant(nT)
      DOUBLE PRECISION hKetoi(Kp)
      DOUBLE PRECISION hU(nT,Kp)
      DOUBLE PRECISION hK(Kp)
      DOUBLE PRECISION vecteu(Kp)
      DOUBLE PRECISION vectoi(Kp)
      DOUBLE PRECISION stat(Kp)
      DOUBLE PRECISION Zavant(nT)
      DOUBLE PRECISION BDa(nT)
      DOUBLE PRECISION BDb(nT)
      DOUBLE PRECISION BDc(nT)
      DOUBLE PRECISION BDd(nT)
      DOUBLE PRECISION Davan2(nT)
      DOUBLE PRECISION ADa(nT)
      DOUBLE PRECISION ADb(nT)
      DOUBLE PRECISION Puiss(Kp)
      DOUBLE PRECISION Puiss2(Kp)
      DOUBLE PRECISION snsort(n,Kp)
      DOUBLE PRECISION snsorv(n)
      DOUBLE PRECISION Qucalc(Kp)
      DOUBLE PRECISION snLe1s(n)
      DOUBLE PRECISION snLe2s(n)
      DOUBLE PRECISION snBDso(n)
      DOUBLE PRECISION snADso(n), snJBso(n)
c-----------------------------------------------------------------------
c     parametres creerdat
c     Nom du fichier de donnees en sortie                               **
      CHARACTER dataf*8
c     Nom du fichier de parametres en sortie                            **
      CHARACTER paramf*17
c     Nombre d'echantillons souhaites
      INTEGER nbcle,nbclea
c     rang d'arret dans la random shock method de Burn                  **
      INTEGER marret
      PARAMETER(marret=200)
c     Induction period dans la methode de Burn                          **
      INTEGER Mind
      PARAMETER(Mind=200)
```

```
c     ecart-type des erreurs                                          **
      DOUBLE PRECISION sigma
      PARAMETER(sigma=1)
c     moyenne dans mon modele ARMA                                    **
      DOUBLE PRECISION mu
      PARAMETER(mu=0)
c     parametre de la khi-deux                                        **
      INTEGER df1
      PARAMETER(df1=4)
c     parametre de la student                                         **
      INTEGER df2
      PARAMETER(df2=5)
c     parametre de la skew-normale                                    **
      DOUBLE PRECISION lambda
      PARAMETER(lambda=2.0)
      DOUBLE PRECISION loi5b, loi5k, loi6p, loi6q
      DOUBLE PRECISION loi7g, loi7d, loi8p, loi8q, loi9a, loi9b
      DOUBLE PRECISION loi10b, loi10l, loi11a, loi11k, loi13g, loi13d
      DOUBLE PRECISION loi14l, loi16p, loi16d, loi17p, loi17m
      DOUBLE PRECISION loi18g, loi18d, loi19a, loi19b
      PARAMETER(loi5b=1.0, loi5k=2.0, loi6p=4, loi6q=1)                  **
      PARAMETER(loi7g=0.0,loi7d=1.0, loi8p=2, loi8q=2, loi9a=0,loi9b=2)    **
      PARAMETER(loi10b=0.2,loi10l=1, loi11a=2.0,loi11k=0.5)           **
      PARAMETER(loi13g=1.0,loi13d=1.0, loi14l=10)                    **
      PARAMETER(loi16p=0.05,loi16d=5.0, loi17p=0.05,loi17m=3.0)        **
      PARAMETER(loi18g=1.0,loi18d=1.0, loi19a=1.1, loi19b=0.5)        **
c     loi des erreurs
c     si loi=0 : Normale(0,sigma^2)
c     si loi=1 : Khi2 centree (df1)
c     si loi=2 : Student (df2)
c     si loi=3 : Skew-Normale(lambda)
c     si loi=4 : Laplace
c     si loi=5 : Weibull(b,k)
c     si loi=6 : Gamma(p,q)
c     si loi=7 : Log-Normale(g,d)
c     si loi=8 : Beta(p,q)
c     si loi=9 : Uniform(a,b)
c     si loi=10 : Shifted exp (l,b)
c     si loi=11 : Pareto(a,k)
c     si loi=12 : Shifted Pareto
c     si loi=13 : SU(g,d)
c     si loi=14 : TU(l)
c     si loi=15 : Logistic
c     si loi=16 : SC(p,d)
c     si loi=17 : LC(p,m)
c     si loi=18 : SB(g,d)
c     si loi=19 : S(a,b)
      INTEGER loi
c     rm=max(p,q)
      INTEGER rm
      PARAMETER(rm=2)
c     the autoregressive coefficients of the model
      DOUBLE PRECISION phi(2)
      DOUBLE PRECISION phich(p)
c     the moving-average coefficients of the model
      DOUBLE PRECISION teta(2)
      DOUBLE PRECISION tetach(q)
c     vecteur des donnees cree
      DOUBLE PRECISION donees(nT)
c Parametres pour G05EGF: simulation
c     the autoregressive coefficients of the model=phi     Input
      DOUBLE PRECISION A(p)
c     the moving-average coefficients of the model=teta     Input
      DOUBLE PRECISION B(q+1)
c     le vecteur des innovations     Output
```

```
      DOUBLE PRECISION R(nT)
c Parametres pour G13DCF: estimation
c     the number of initial parameter estimates   Input                    **
c     mettre p+q+1 si MEAN=.TRUE. , p+q si MEAN=.FALSE.
      INTEGER NPAR
      PARAMETER(NPAR=p+q)
c     first dimension of the array CM       Input                          **
c     mettre p+q+1 si MEAN=.TRUE. , p+q si MEAN=.FALSE.
      INTEGER ICM
      PARAMETER(ICM=p+q)
c     Workspace
      INTEGER IW(NPAR+rm+3)
c     initial parameter estimates    Input/Output
      DOUBLE PRECISION PAR(NPAR)
c     W(i,t) must be set equal to the observation at time t    Input
c     of the ith series
      DOUBLE PRECISION W(1,nT)
c     the accuracy to which the solution in PAR and QQ is required    Input    **
      DOUBLE PRECISION CGETOL
      PARAMETER(CGETOL=0.0001)
c     residual at time t for series i, for i = 1,2,...,k     Output
      DOUBLE PRECISION V(1,nT)
c     estimated first derivative of the log-likelihood function    Output
      DOUBLE PRECISION G(NPAR)
c     estimate of the correlation coefficient between the ith and    Output
c     jth elements in the PAR array
      DOUBLE PRECISION CM(ICM,NPAR)
c     Workspace
      DOUBLE PRECISION WORK((5+3*nT+2*rm)+(2*p+q+2*rm*(rm+2)+9)+
     +                      (NPAR+1)*(5*(NPAR+1)+29)/2+(rm+1)**2)
c     MEAN must be set to .TRUE. if components of mu are to    Input           **
c     be estimated and .FALSE. if all elements of mu are to be taken as zero
      LOGICAL MEAN
      PARAMETER(MEAN=.FALSE.)
c     PARHLD(i) must be set to .TRUE., if PAR(i) is to be       Input
c     held constant at its input value and .FALSE., if PAR(i) is a
c     free parameter, for i = 1,2,...,NPAR.
      LOGICAL PARHLD(NPAR)
c     EXACT must be set equal to .TRUE. if the user wishes    Input           **
c     the routine to compute exact maximum likelihood estimates.
c     EXACT must be set equal to .FALSE. if only conditional
c     likelihood estimates are required.
c     voir EXACT plus bas
      LOGICAL EXACT
c     Contient les p donnees initiales de la serie
      DOUBLE PRECISION Wtip(p)
c     marret+p innovations genere par shock.f
      DOUBLE PRECISION shocks(marret+p)
c     defini dans Burn, calcule par ARMpsi.f
      DOUBLE PRECISION psi(marret+1)
c     utile dans simARM.f de longueur marret
      DOUBLE PRECISION phi2(marret)
c     utile dans simARM.f de longueur marret
      DOUBLE PRECISION teta2(marret)
c     utile dans simARM.f
      DOUBLE PRECISION YtpMn(p+Mind+nT)
c     utile dans simARM.f
      DOUBLE PRECISION Atnq(nT+Mind+q)
c     Esperances des lois SU, SB et S                          **
      DOUBLE PRECISION EspSU,EspSB,EspS
      PARAMETER(EspSU=-1.93758, EspSB=0.696735, EspS=0)
      CHARACTER*13 efface
      DOUBLE PRECISION temp(5)
      INTEGER j
      CHARACTER*5 ftemp
```

```
      CHARACTER*1 INTCH1
      CHARACTER*2 INTCH2
      CHARACTER*3 INTCH3
      CHARACTER*4 INTCH4
      DOUBLE PRECISION matric(n,18)
      INTEGER dnT
      PARAMETER(dnT=Kp)
      DOUBLE PRECISION QLed1u,QLe12u
      PARAMETER(QLed1u=QuLed1,QLe12u=QuLe12)
      DOUBLE PRECISION QLed2u,QLe22u
      PARAMETER(QLed2u=QuLed2,QLe22u=QuLe22)
      DOUBLE PRECISION QBDu,QBD2u
      PARAMETER(QBDu=QuBD,QBD2u=QuBD2)
      DOUBLE PRECISION QADu,QAD2u
      PARAMETER(QADu=QuAD,QAD2u=QuAD2)
      DOUBLE PRECISION QJBu,QJB2u
      PARAMETER(QJBu=QuJB,QJB2u=QuJB2)
c     the mean of the time series      Input
      DOUBLE PRECISION E
      PARAMETER(E=mu)
c     the number of autoregressive coefficients supplied      Input
      INTEGER NA
      PARAMETER(NA=p)
c     the number of moving-average coefficients supplied      Input
      INTEGER NB
      PARAMETER(NB=q+1)
c     the dimension of the array R: vecteur des innovations      Input
      INTEGER NR
      PARAMETER(NR=11)
c     the number of observed time series, k  (chez moi k=1)  Input
      INTEGER K
      PARAMETER(K=1)
c     the number of observations in each time series, n (chez moi=nT)   Input
      INTEGER N2
      PARAMETER(N2=nT)
c     the number of AR parameter matrices, p     Input
      INTEGER IP
      PARAMETER(IP=p)
c     the number of MA parameter matrices, q    Input
      INTEGER IQ
      PARAMETER(IQ=q)
c     the first dimension of the arrays QQ, W and V     Input
      INTEGER IK
      PARAMETER(IK=1)
c     the maximum number of likelihood evaluations to be     Input
c     permitted by the search procedure
      INTEGER MAXCAL
      PARAMETER(MAXCAL=40*NPAR*(NPAR+5))
c     which quantities are to be printed     Input
      INTEGER ISHOW
      PARAMETER(ISHOW=0)
c     dimension of the array WORK     Input
      INTEGER LWORK
      PARAMETER(LWORK=(5+3*(nT)+2*rm)+(2*p+q+2*rm*(rm+2)+9)+
     +                (NPAR+1)*(5*(NPAR+1)+29)/2+(rm+1)**2)
c     dimension of the array IW     Input
      INTEGER LIW
      PARAMETER(LIW=NPAR+rm+3)
c     QQ(i,j) must be set equal to an initial estimate of     Input/Output
c     the (i,j)th element of the covariance matrix of the residual series
      DOUBLE PRECISION QQ(IK,K)
      INTEGER NI, NX, IFAIL
      PARAMETER(NX=4,NI=9)
      INTEGER IA(NI)
      DOUBLE PRECISION XA(NX)
```

```
c      seed=1 (non-repeatable sequence) ou seed=0 (repeatable sequence)
     **
       INTEGER seed
       PARAMETER(seed=0)
c-----------------------------------------------------------------
c Fin de declaration des variables
c-----------------------------------------------------------------
c G05CBF Initialise random number generating routines to give repeatable sequence
c G05CCF Initialise random number generating routines to give non-repeatable sequence
c G05CFF Save state of random number generating routines
c G05CGF Restore state of random number generating routines
       IF (seed .EQ. 1) THEN
       CALL G05CCF
       ENDIF
       IF (seed .EQ. 0) THEN
       CALL G05CBF(0)
       ENDIF
       IFAIL=0
c      Rajouter ici, si voulu, les IA et les XA a prendre a la fin du fichier para.ijkl
     :          **
c      A (de)commenter si necessaire (si on rajoute les IA et XA, on decommente)
c       CALL G05CGF(IA,NI,XA,NX,IFAIL)
       CALL G05CFF(IA,NI,XA,NX,IFAIL)
       EXACT=.TRUE.                                                    **
       nbcle=n
       efface='rm ./data.txt'
       ftemp='ftemp'
c      On va lire les parametres necessaires dans ftemp
c      temp va contenir sur chaque ligne: j, loi, phi(1), teta(1), teta(2)
       OPEN(UNIT=15, FILE=ftemp , STATUS='OLD' )
       READ(15,*,END=10) (temp(j),j=1,5)
       CLOSE(15)
 10    j=temp(1)
       loi=temp(2)
       phi(1)=temp(3)
       teta(1)=temp(4)
       teta(2)=temp(5)
       phi(2)=0.0
c      On sauvegarde la valeur de nbcle
       nbclea=nbcle
c      Nom du fichier de donnees en sortie                             **
       dataf='data.txt'
c      Permet de convertir l'entier j en chaine de caracteres
       IF (j/10 .LT. 1) THEN
          WRITE(INTCH1, '(I1)') j
c      Nom du fichier de parametres en sortie                          **
          paramf='./SIMUL/para.'//'000'//INTCH1
          filew='./SIMUL/resultat.'//'000'//INTCH1
       ENDIF
       IF ((j/10 .EQ. 1) .OR. ((j/10 .GT. 1) .AND. (j/10 .LT. 10))) THEN
          WRITE(INTCH2, '(I2)') j
c      Nom du fichier de parametres en sortie                          **
          paramf='./SIMUL/para.'//'00'//INTCH2
          filew='./SIMUL/resultat.'//'00'//INTCH2
       ENDIF
       IF ((j/10 .EQ. 10) .OR. ((j/10 .GT. 10) .AND. (j/10 .LT. 100)))
      + THEN
          WRITE(INTCH3, '(I3)') j
c      Nom du fichier de parametres en sortie                          **
          paramf='./SIMUL/para.'//'0'//INTCH3
          filew='./SIMUL/resultat.'//'0'//INTCH3
       ENDIF
       IF ((j/10 .EQ. 100) .OR. ((j/10 .GT. 100) .AND. (j/10 .LT. 1000)))
      + THEN
          WRITE(INTCH4, '(I4)') j
```

```
c     Nom du fichier de parametres en sortie                          **
         paramf='./SIMUL/para.'//INTCH4
         filew='./SIMUL/resultat.'//INTCH4
      ENDIF
      IF (loi .EQ. 0) THEN
c     Quantiles de la loi du Khi2 (K=10;alpha=0.1)                    **
      khi(1)=2.71
      khi(2)=4.61
      khi(3)=6.25
      khi(4)=7.78
      khi(5)=9.24
      khi(6)=10.64
      khi(7)=12.02
      khi(8)=13.36
      khi(9)=14.68
      khi(10)=15.99
c     Quantiles de la loi du Khi2 (K=10;alpha=0.05)                   **
      khi2(1)=3.84
      khi2(2)=5.99
      khi2(3)=7.81
      khi2(4)=9.49
      khi2(5)=11.07
      khi2(6)=12.59
      khi2(7)=14.07
      khi2(8)=15.51
      khi2(9)=16.92
      khi2(10)=18.31
      ENDIF
      IF (loi .NE. 0) THEN
c     Quantiles utilises pour le calcul de la stat(vrais quantiles calcules sous H0)
     **
c     avec alpha=0.1
      khi(1)=2.71
      khi(2)=4.61
      khi(3)=6.25
      khi(4)=7.78
      khi(5)=9.24
      khi(6)=10.64
      khi(7)=12.02
      khi(8)=13.36
      khi(9)=14.68
      khi(10)=15.99
c     Quantiles utilises pour le calcul de la stat(vrais quantiles calcules sous H0)
     **
c     avec alpha=0.05
      khi2(1)=3.84
      khi2(2)=5.99
      khi2(3)=7.81
      khi2(4)=9.49
      khi2(5)=11.07
      khi2(6)=12.59
      khi2(7)=14.07
      khi2(8)=15.51
      khi2(9)=16.92
      khi2(10)=18.31
      ENDIF
c     On peut aussi changer le nom du fichier en entree (data.txt) et    **
c     le nom du fichier en sortie (resultat.txt)
      filer='data.txt'
c-------------------------------------------------------------------
c     ---------------------------------------------------------------
c     DEBUT DU PROGRAMME
c     ---------------------------------------------------------------
      CALL cdARMA(dataf,paramf,nbcle,marret,Mind,sigma,mu,df1,df2,
     + lambda,loi5b,loi5k,loi6p,loi6q,loi7g,loi7d,loi8p,loi8q,loi9a,
```

```
     +loi9b,loi10b,loi10l,loi11a,loi11k,loi13g,loi13d,loi14l,loi16p,
     +loi16d,loi17p,loi17m,loi18g,loi18d,loi19a,loi19b,loi,p,q,rm,nT,
     + phi,phich,teta,tetach,donees,A,
     + B,R,NPAR,ICM,IW,PAR,W,CGETOL,V,G,CM,WORK,MEAN,PARHLD,EXACT,Wtip,
     + shocks,psi,phi2,teta2,YtpMn,Atnq,EspSU,EspSB,EspS,
     +E,NA,NB,NR,K,N2,IP,IQ,IK,MAXCAL,ISHOW,LWORK,LIW,QQ)
c Il faut faire modifier par cdARMA la valeur nbcle en entree pour lui faire sortir
c une nouvelle valeur nbcle qui est le nombre de bons echantillons conserves
c Ensuite, il faut utiliser cette valeur dans calcstat, comme etant la vraie valeur
c du nombre d'echantillons disponibles, la valeur de n que l'on a mise etant
c la valeur maximale possible de bons echantillons, donc les differentes matrices
    initialisees
c a l'aide de la valeur de n sont plus grandes que necessaires (elles auraient du etre
c initialisees avec la valeur de nbcle renvoyee par cd ARMA mais ce n'est pas possible
c avec fortran77) et il faut en tenir compte; donc bien regarder partout dans le
c programme calcstat la ou il y a n: OK CA C'EST FAIT!!! A METTRE EN COMMENTAIRES
    QUELQUEPART ...
        CALL calcstat(isa,khi,khi2,QuLed1,QuLe12,QuLed2,QuLe22,QuBD,
     + QuBD2,QuAD,QuAD2,QuJB,QuJB2,alpha,alpha2,Kp,p,q,nT,phi,teta,n,
     + nbcle,m,filer,filew,valeur,sch2vc,epscha,compt,compt2,statn,
     + KoLed1,KoLed2,Ledwi1,Ledwi2,snLed1,snLed2,stanBD,stanAD,stanJB,Z,
     + D,Davant,E2,U,hUetoi,Uavant,hKetoi,hU,hK,vecteu,vectoi,stat,
     + Zavant,BDa,BDb,BDc,BDd,Davan2,ADa,ADb,Puiss,Puiss2,snsort,snsorv,
     + Qucalc,snLe1s,snLe2s,snBDso,snADso,snJBso,matric,paramf,loi,
     + dnT,QLed1u,QLe12u,QLed2u,QLe22u,QBDu,QBD2u,QADu,QAD2u,QJBu,QJB2u)
        nbcle=nbclea
c     FAIRE ATTENTION:
c     voir si mes programmes ne modifient pas certains de leurs parametres en entree
c     ce qui pourrait ammener des erreurs au cours de la boucle que je vais faire dans
     le programme test.f
c     pour les differentes etapes 1), 2) et 3)
       CALL SYSTEM(efface)
c     Sauvegarde du seed
      OPEN(UNIT=14, FILE=paramf , STATUS='OLD', ACCESS='append' )
      WRITE(14,*) 'Sauvegarde du seed:'
      WRITE(14,*) 'XA:'
      WRITE(14,15) 'XA(1)=',XA(1)
      WRITE(14,15) 'XA(2)=',XA(2)
      WRITE(14,15) 'XA(3)=',XA(3)
      WRITE(14,15) 'XA(4)=',XA(4)
 15   FORMAT(A6,1F20.15)
      WRITE(14,*) 'IA:'
      WRITE(14,30) 'IA(1)=',IA(1)
      WRITE(14,30) 'IA(2)=',IA(2)
      WRITE(14,30) 'IA(3)=',IA(3)
      WRITE(14,30) 'IA(4)=',IA(4)
      WRITE(14,30) 'IA(5)=',IA(5)
      WRITE(14,30) 'IA(6)=',IA(6)
      WRITE(14,30) 'IA(7)=',IA(7)
      WRITE(14,30) 'IA(8)=',IA(8)
      WRITE(14,30) 'IA(9)=',IA(9)
 30     FORMAT(A6,1I10)
      CLOSE(UNIT=14)
      END
      INCLUDE 'mean.f'
      INCLUDE 'simulARMA.f'
      INCLUDE 'ARMpsi.f'
      INCLUDE 'shock.f'
      INCLUDE 'rskew.f'
      INCLUDE 'rlap.f'
      INCLUDE 'rpare.f'
      INCLUDE 'rspare.f'
      INCLUDE 'rSU.f'
      INCLUDE 'rTU.f'
      INCLUDE 'rSC.f'
```

```
      INCLUDE 'rLC.f'
      INCLUDE 'rSB.f'
      INCLUDE 'rS.f'
      INCLUDE 'H1etoile.f'
      INCLUDE 'H2etoile.f'
      INCLUDE 'H3etoile.f'
      INCLUDE 'H4etoile.f'
      INCLUDE 'H5etoile.f'
      INCLUDE 'H6etoile.f'
      INCLUDE 'H7etoile.f'
      INCLUDE 'H8etoile.f'
      INCLUDE 'H9etoile.f'
      INCLUDE 'H10etoile.f'
      INCLUDE 'H1isa.f'
      INCLUDE 'H2isa.f'
      INCLUDE 'H3isa.f'
      INCLUDE 'H4isa.f'
      INCLUDE 'H5isa.f'
      INCLUDE 'H6isa.f'
      INCLUDE 'H7isa.f'
      INCLUDE 'H8isa.f'
      INCLUDE 'H9isa.f'
      INCLUDE 'H10isa.f'
      INCLUDE 'H1.f'
      INCLUDE 'H2.f'
      INCLUDE 'H3.f'
      INCLUDE 'H4.f'
      INCLUDE 'H5.f'
      INCLUDE 'H6.f'
      INCLUDE 'H7.f'
      INCLUDE 'H8.f'
      INCLUDE 'H9.f'
      INCLUDE 'H10.f'
      INCLUDE 'qnorm.f'
      INCLUDE 'pnorm.f'
      INCLUDE 'min.f'
      INCLUDE 'max.f'
      INCLUDE 'var.f'
      INCLUDE 'creerdat_ARMA.f'
      INCLUDE 'calcstat.f'
```

Programme big_prog_ARMA21.f

```
c Idee de ce  programme:
c Etape1)
c On cree un fichier de donnees 'data.txt' en utilisant certaines valeurs de
     parametres
c a lire dans un fichier d'entree 'ftemp' cree par le programme test.f a l'aide du
c fichier 'paramentree', on stocke les resultats autres que
c les donnees dans un fichier 'param.i'
c Etape 2)
c ensuite on utilise le fichier 'data.txt'
c avec le programme calcstat pour creer le fichier des statistiques 'resultat.i'
c Etape 3)
c Ensuite, on efface le fichier 'data.txt'
      PROGRAM main
c     Les ** indiquent les endroits ou des changements peuvent etre necessaires
c     ------------------------------------------------------------------
c     DEBUT DE DECLARATION DES VARIABLES
c     ------------------------------------------------------------------
c     Si isa=.TRUE. (ie MEAN=.TRUE.) on prend les polynomes modifies avec les ak  **
c     Si isa=.FALSE. (ie MEAN=.FALSE.) on prend les polynomes modifies sans les ak
      LOGICAL isa
      PARAMETER(isa=.FALSE.)
c-----------------------------------
```

```
c     parametres calcstat
c     Niveau 1 du test                                          **
      DOUBLE PRECISION alpha
      PARAMETER(alpha=0.1)
c     Niveau 2 du test                                          **
      DOUBLE PRECISION alpha2
      PARAMETER(alpha2=0.05)
c     Quantiles avec alpha
      DOUBLE PRECISION  khi(10)
c     Quantiles avec alpha2
      DOUBLE PRECISION  khi2(10)
c     Quantiles Ledwi1 et Ledwi2 avec alpha                          **
      DOUBLE PRECISION QuLed1, QuLed2
c     avec T=50
      PARAMETER(QuLed1=3.69,QuLed2=5.466)
c     avec T=100
c      PARAMETER(QuLed1=3.275,QuLed2=5.26)
c     avec T=200
c      PARAMETER(QuLed1=3.057,QuLed2=5.043)
c     Quantiles Ledwi1 et Ledwi2 avec alpha2                          **
      DOUBLE PRECISION QuLe12, QuLe22
c     avec T=50
      PARAMETER(QuLe12=5.41,QuLe22=7.14)
c     avec T=100
c      PARAMETER(QuLe12=5.20,QuLe22=6.97)
c     avec T=200
c      PARAMETER(QuLe12=4.751,QuLe22=6.796)
c     Quantiles Brockwell et Davis et Anderson Darling et JB avec alpha       **
      DOUBLE PRECISION QuBD, QuAD, QuJB
c     avec T=50
      PARAMETER(QuBD=0.920,QuAD=1.743,QuJB=4.61)
c     avec T=100
c      PARAMETER(QuBD=0.958,QuAD=1.743,QuJB=4.61)
c     avec T=200
c      PARAMETER(QuBD=0.978,QuAD=1.743,QuJB=4.61)
c     Quantiles Brockwell et Davis et Anderson Darling et JB avec alpha2       **
      DOUBLE PRECISION QuBD2, QuAD2, QuJB2
c     avec T=50
      PARAMETER(QuBD2=0.899,QuAD2=2.308,QuJB2=5.99)
c      avec T=100
c      PARAMETER(QuBD2=0.947,QuAD2=2.308,QuJB2=5.99)
c     avec T=200
c      PARAMETER(QuBD2=0.973,QuAD2=2.308,QuJB2=5.99)
c     Nombre de polynomes                                       **
      INTEGER Kp
      PARAMETER(Kp=10)
c     ordre du modele AR
      INTEGER p
      PARAMETER(p=2)
c     ordre du modele MA
      INTEGER q
      PARAMETER(q=1)
c     nombre d'observations dans mon echantillon                      **
c     Si on change la valeur de nT, il faut aller modifier
c     la valeur dans FORMAT a la fin du programme creerdat_ARMA.f
      INTEGER nT
      PARAMETER(nT=50)
      INTEGER n,m
      PARAMETER(m=nT+1)
c     n=nombre de lignes au maximum dans le fichier de donnees          **
c     cela correspond a nbcle s'il n'y a pas eu d'erreurs
c     dans tous les cas mettre ici n=nbcle=nombre d'echantillons souhaites
      PARAMETER(n=10000)
      CHARACTER filer*8, filew*21                                     **
      DOUBLE PRECISION valeur(n,m)
```


```
      DOUBLE PRECISION sch2vc(n)
      DOUBLE PRECISION epscha(n,nT)
      INTEGER compt(Kp)
      INTEGER compt2(Kp)
      DOUBLE PRECISION statn(n,Kp)
      INTEGER KoLed1(n)
      INTEGER KoLed2(n)
      DOUBLE PRECISION Ledwi1(Kp)
      DOUBLE PRECISION Ledwi2(Kp-1)
      DOUBLE PRECISION snLed1(n)
      DOUBLE PRECISION snLed2(n)
      DOUBLE PRECISION stanBD(n)
      DOUBLE PRECISION stanAD(n), stanJB(n)
      DOUBLE PRECISION Z(nT)
      DOUBLE PRECISION D(nT)
      DOUBLE PRECISION Davant(nT), E2(nT)
      DOUBLE PRECISION U(nT)
      DOUBLE PRECISION hUetoi(nT,Kp)
      DOUBLE PRECISION Uavant(nT)
      DOUBLE PRECISION hKetoi(Kp)
      DOUBLE PRECISION hU(nT,Kp)
      DOUBLE PRECISION hK(Kp)
      DOUBLE PRECISION vecteu(Kp)
      DOUBLE PRECISION vectoi(Kp)
      DOUBLE PRECISION stat(Kp)
      DOUBLE PRECISION Zavant(nT)
      DOUBLE PRECISION BDa(nT)
      DOUBLE PRECISION BDb(nT)
      DOUBLE PRECISION BDc(nT)
      DOUBLE PRECISION BDd(nT)
      DOUBLE PRECISION Davan2(nT)
      DOUBLE PRECISION ADa(nT)
      DOUBLE PRECISION ADb(nT)
      DOUBLE PRECISION Puiss(Kp)
      DOUBLE PRECISION Puiss2(Kp)
      DOUBLE PRECISION snsort(n,Kp)
      DOUBLE PRECISION snsorv(n)
      DOUBLE PRECISION Qucalc(Kp)
      DOUBLE PRECISION snLe1s(n)
      DOUBLE PRECISION snLe2s(n)
      DOUBLE PRECISION snBDso(n)
      DOUBLE PRECISION snADso(n), snJBso(n)
c-------------------------------------------------------------------
c     parametres creerdat
c     Nom du fichier de donnees en sortie                          **
      CHARACTER dataf*8
c     Nom du fichier de parametres en sortie                       **
      CHARACTER paramf*17
c     Nombre d'echantillons souhaites
      INTEGER nbcle,nbclea
c     rang d'arret dans la random shock method de Burn             **
      INTEGER marret
      PARAMETER(marret=200)
c     Induction period dans la methode de Burn                     **
      INTEGER Mind
      PARAMETER(Mind=200)
c     ecart-type des erreurs                                       **
      DOUBLE PRECISION sigma
      PARAMETER(sigma=1)
c     moyenne dans mon modele ARMA                                 **
      DOUBLE PRECISION mu
      PARAMETER(mu=0)
c     parametre de la khi-deux                                     **
      INTEGER df1
      PARAMETER(df1=4)
```

```
c     parametre de la student                                          **
      INTEGER df2
      PARAMETER(df2=5)
c     parametre de la skew-normale                                     **
      DOUBLE PRECISION lambda
      PARAMETER(lambda=2.0)
      DOUBLE PRECISION loi5b, loi5k, loi6p, loi6q
      DOUBLE PRECISION loi7g, loi7d, loi8p, loi8q, loi9a, loi9b
      DOUBLE PRECISION loi10b, loi10l, loi11a, loi11k, loi13g, loi13d
      DOUBLE PRECISION loi14l, loi16p, loi16d, loi17p, loi17m
      DOUBLE PRECISION loi18g, loi18d, loi19a, loi19b
      PARAMETER(loi5b=1.0, loi5k=2.0, loi6p=4, loi6q=1)                  **
      PARAMETER(loi7g=0.0,loi7d=1.0, loi8p=2, loi8q=2, loi9a=0,loi9b=2)    **
      PARAMETER(loi10b=0.2,loi10l=1, loi11a=2.0,loi11k=0.5)            **
      PARAMETER(loi13g=1.0,loi13d=1.0, loi14l=10)                     **
      PARAMETER(loi16p=0.05,loi16d=5.0, loi17p=0.05,loi17m=3.0)          **
      PARAMETER(loi18g=1.0,loi18d=1.0, loi19a=1.1, loi19b=0.5)         **
c     loi des erreurs
c     si loi=0 : Normale(0,sigma^2)
c     si loi=1 : Khi2 centree (df1)
c     si loi=2 : Student (df2)
c     si loi=3 : Skew-Normale(lambda)
c     si loi=4 : Laplace
c     si loi=5 : Weibull(b,k)
c     si loi=6 : Gamma(p,q)
c     si loi=7 : Log-Normale(g,d)
c     si loi=8 : Beta(p,q)
c     si loi=9 : Uniform(a,b)
c     si loi=10 : Shifted exp (l,b)
c     si loi=11 : Pareto(a,k)
c     si loi=12 : Shifted Pareto
c     si loi=13 : SU(g,d)
c     si loi=14 : TU(l)
c     si loi=15 : Logistic
c     si loi=16 : SC(p,d)
c     si loi=17 : LC(p,m)
c     si loi=18 : SB(g,d)
c     si loi=19 : S(a,b)
      INTEGER loi
c     rm=max(p,q)
      INTEGER rm
      PARAMETER(rm=2)
c     the autoregressive coefficients of the model
      DOUBLE PRECISION phi(2)
      DOUBLE PRECISION phich(p)
c     the moving-average coefficients of the model
      DOUBLE PRECISION teta(2)
      DOUBLE PRECISION tetach(q)
c     vecteur des donnees cree
      DOUBLE PRECISION donees(nT)
c Parametres pour G05EGF: simulation
c     the autoregressive coefficients of the model=phi     Input
      DOUBLE PRECISION A(p)
c     the moving-average coefficients of the model=teta     Input
      DOUBLE PRECISION B(q+1)
c     le vecteur des innovations     Output
      DOUBLE PRECISION R(nT)
c Parametres pour G13DCF: estimation
c     the number of initial parameter estimates   Input                 **
c     mettre p+q+1 si MEAN=.TRUE. , p+q si MEAN=.FALSE.
      INTEGER NPAR
      PARAMETER(NPAR=p+q)
c     first dimension of the array CM       Input                    **
c     mettre p+q+1 si MEAN=.TRUE. , p+q si MEAN=.FALSE.
      INTEGER ICM
```

```
      PARAMETER(ICM=p+q)
c     Workspace
      INTEGER IW(NPAR+rm+3)
c     initial parameter estimates     Input/Output
      DOUBLE PRECISION PAR(NPAR)
c     W(i,t) must be set equal to the observation at time t     Input
c     of the ith series
      DOUBLE PRECISION W(1,nT)
c     the accuracy to which the solution in PAR and QQ is required     Input        **
      DOUBLE PRECISION CGETOL
      PARAMETER(CGETOL=0.0001)
c     residual at time t for series i, for i = 1,2,...,k      Output
      DOUBLE PRECISION V(1,nT)
c     estimated first derivative of the log-likelihood function     Output
      DOUBLE PRECISION G(NPAR)
c     estimate of the correlation coefficient between the ith and     Output
c     jth elements in the PAR array
      DOUBLE PRECISION CM(ICM,NPAR)
c     Workspace
      DOUBLE PRECISION WORK((5+3*nT+2*rm)+(2*p+q+2*rm*(rm+2)+9)+
     +                      (NPAR+1)*(5*(NPAR+1)+29)/2+(rm+1)**2)
c     MEAN must be set to .TRUE. if components of mu are to     Input              **
c     be estimated and .FALSE. if all elements of mu are to be taken as zero
      LOGICAL MEAN
      PARAMETER(MEAN=.FALSE.)
c     PARHLD(i) must be set to .TRUE., if PAR(i) is to be       Input
c     held constant at its input value and .FALSE., if PAR(i) is a
c     free parameter, for i = 1,2,...,NPAR.
      LOGICAL PARHLD(NPAR)
c     EXACT must be set equal to .TRUE. if the user wishes     Input              **
c     the routine to compute exact maximum likelihood estimates.
c     EXACT must be set equal to .FALSE. if only conditional
c     likelihood estimates are required.
c     voir EXACT plus bas
      LOGICAL EXACT
c     Contient les p donnees initiales de la serie
      DOUBLE PRECISION Wtip(p)
c     marret+p innovations genere par shock.f
      DOUBLE PRECISION shocks(marret+p)
c     defini dans Burn, calcule par ARMpsi.f
      DOUBLE PRECISION psi(marret+1)
c     utile dans simARM.f de longueur marret
      DOUBLE PRECISION phi2(marret)
c     utile dans simARM.f de longueur marret
      DOUBLE PRECISION teta2(marret)
c     utile dans simARM.f
      DOUBLE PRECISION YtpMn(p+Mind+nT)
c     utile dans simARM.f
      DOUBLE PRECISION Atnq(nT+Mind+q)
c     Esperances des lois SU, SB et S                                    **
      DOUBLE PRECISION EspSU,EspSB,EspS
      PARAMETER(EspSU=-1.93758, EspSB=0.696735, EspS=0)
      CHARACTER*13 efface
      DOUBLE PRECISION temp(5)
      INTEGER j
      CHARACTER*5 ftemp
      CHARACTER*1 INTCH1
      CHARACTER*2 INTCH2
      CHARACTER*3 INTCH3
      CHARACTER*4 INTCH4
      DOUBLE PRECISION matric(n,18)
      INTEGER dnT
      PARAMETER(dnT=Kp)
      DOUBLE PRECISION QLed1u,QLe12u
      PARAMETER(QLed1u=QuLed1,QLe12u=QuLe12)
```

```
      DOUBLE PRECISION QLed2u,QLe22u
      PARAMETER(QLed2u=QuLed2,QLe22u=QuLe22)
      DOUBLE PRECISION QBDu,QBD2u
      PARAMETER(QBDu=QuBD,QBD2u=QuBD2)
      DOUBLE PRECISION QADu,QAD2u
      PARAMETER(QADu=QuAD,QAD2u=QuAD2)
      DOUBLE PRECISION QJBu,QJB2u
      PARAMETER(QJBu=QuJB,QJB2u=QuJB2)
c     the mean of the time series      Input
      DOUBLE PRECISION E
      PARAMETER(E=mu)
c     the number of autoregressive coefficients supplied      Input
      INTEGER NA
      PARAMETER(NA=p)
c     the number of moving-average coefficients supplied      Input
      INTEGER NB
      PARAMETER(NB=q+1)
c     the dimension of the array R: vecteur des innovations      Input
      INTEGER NR
      PARAMETER(NR=10)
c     the number of observed time series, k  (chez moi k=1)  Input
      INTEGER K
      PARAMETER(K=1)
c     the number of observations in each time series, n (chez moi=nT)   Input
      INTEGER N2
      PARAMETER(N2=nT)
c     the number of AR parameter matrices, p     Input
      INTEGER IP
      PARAMETER(IP=p)
c     the number of MA parameter matrices, q    Input
      INTEGER IQ
      PARAMETER(IQ=q)
c     the first dimension of the arrays QQ, W and V     Input
      INTEGER IK
      PARAMETER(IK=1)
c     the maximum number of likelihood evaluations to be     Input
c     permitted by the search procedure
      INTEGER MAXCAL
      PARAMETER(MAXCAL=40*NPAR*(NPAR+5))
c     which quantities are to be printed     Input
      INTEGER ISHOW
      PARAMETER(ISHOW=0)
c     dimension of the array WORK     Input
      INTEGER LWORK
      PARAMETER(LWORK=(5+3*(nT)+2*rm)+(2*p+q+2*rm*(rm+2)+9)+
     +                    (NPAR+1)*(5*(NPAR+1)+29)/2+(rm+1)**2)
c     dimension of the array IW     Input
      INTEGER LIW
      PARAMETER(LIW=NPAR+rm+3)
c     QQ(i,j) must be set equal to an initial estimate of     Input/Output
c     the (i,j)th element of the covariance matrix of the residual series
      DOUBLE PRECISION QQ(IK,K)
      INTEGER NI, NX, IFAIL
      PARAMETER(NX=4,NI=9)
      INTEGER IA(NI)
      DOUBLE PRECISION XA(NX)
c     seed=1 (non-repeatable sequence) ou seed=0 (repeatable sequence)
     **
      INTEGER seed
      PARAMETER(seed=0)
c-----------------------------------------------------------------
c Fin de declaration des variables
c-----------------------------------------------------------------
c G05CBF Initialise random number generating routines to give repeatable sequence
c G05CCF Initialise random number generating routines to give non-repeatable sequence
```


```
c G05CFF Save state of random number generating routines
c G05CGF Restore state of random number generating routines
      IF (seed .EQ. 1) THEN
      CALL G05CCF
      ENDIF
      IF (seed .EQ. 0) THEN
      CALL G05CBF(0)
      ENDIF
      IFAIL=0
c     Rajouter ici, si voulu, les IA et les XA a prendre a la fin du fichier para.ijkl
     :          **
c     A (de)commenter si necessaire (si on rajoute les IA et XA, on decommente)
c      CALL G05CGF(IA,NI,XA,NX,IFAIL)
      CALL G05CFF(IA,NI,XA,NX,IFAIL)
      EXACT=.TRUE.                                                    **
      nbcle=n
      efface='rm ./data.txt'
      ftemp='ftemp'
c     On va lire les parametres necessaires dans ftemp
c     temp va contenir sur chaque ligne: j, loi, phi(1), phi(2), teta(1), teta(2)
      OPEN(UNIT=15, FILE=ftemp , STATUS='OLD' )
      READ(15,*,END=10) (temp(j),j=1,5)
      CLOSE(15)
 10   j=temp(1)
      loi=temp(2)
      phi(1)=temp(3)
      phi(2)=temp(4)
      teta(1)=temp(5)
      teta(2)=0.0
c     On sauvegarde la valeur de nbcle
      nbclea=nbcle
c     Nom du fichier de donnees en sortie                             **
      dataf='data.txt'
c     Permet de convertir l'entier j en chaine de caracteres
      IF (j/10 .LT. 1) THEN
         WRITE(INTCH1, '(I1)') j
c     Nom du fichier de parametres en sortie                          **
         paramf='./SIMUL/para.'//'000'//INTCH1
         filew='./SIMUL/resultat.'//'000'//INTCH1
      ENDIF
      IF ((j/10 .EQ. 1) .OR. ((j/10 .GT. 1) .AND. (j/10 .LT. 10))) THEN
         WRITE(INTCH2, '(I2)') j
c     Nom du fichier de parametres en sortie                          **
         paramf='./SIMUL/para.'//'00'//INTCH2
         filew='./SIMUL/resultat.'//'00'//INTCH2
      ENDIF
      IF ((j/10 .EQ. 10) .OR. ((j/10 .GT. 10) .AND. (j/10 .LT. 100)))
     + THEN
         WRITE(INTCH3, '(I3)') j
c     Nom du fichier de parametres en sortie                          **
         paramf='./SIMUL/para.'//'0'//INTCH3
         filew='./SIMUL/resultat.'//'0'//INTCH3
      ENDIF
      IF ((j/10 .EQ. 100) .OR. ((j/10 .GT. 100) .AND. (j/10 .LT. 1000)))
     + THEN
         WRITE(INTCH4, '(I4)') j
c     Nom du fichier de parametres en sortie                          **
         paramf='./SIMUL/para.'//INTCH4
         filew='./SIMUL/resultat.'//INTCH4
      ENDIF
      IF (loi .EQ. 0) THEN
c     Quantiles de la loi du Khi2 (K=10;alpha=0.1)                    **
      khi(1)=2.71
      khi(2)=4.61
      khi(3)=6.25
```

```
      khi(4)=7.78
      khi(5)=9.24
      khi(6)=10.64
      khi(7)=12.02
      khi(8)=13.36
      khi(9)=14.68
      khi(10)=15.99
c     Quantiles de la loi du Khi2 (K=10;alpha=0.05)                    **
      khi2(1)=3.84
      khi2(2)=5.99
      khi2(3)=7.81
      khi2(4)=9.49
      khi2(5)=11.07
      khi2(6)=12.59
      khi2(7)=14.07
      khi2(8)=15.51
      khi2(9)=16.92
      khi2(10)=18.31
      ENDIF
      IF (loi .NE. 0) THEN
c     Quantiles utilises pour le calcul de la stat(vrais quantiles calcules sous H0)
     **
c     avec alpha=0.1
      khi(1)=2.71
      khi(2)=4.61
      khi(3)=6.25
      khi(4)=7.78
      khi(5)=9.24
      khi(6)=10.64
      khi(7)=12.02
      khi(8)=13.36
      khi(9)=14.68
      khi(10)=15.99
c     Quantiles utilises pour le calcul de la stat(vrais quantiles calcules sous H0)
     **
c     avec alpha=0.05
      khi2(1)=3.84
      khi2(2)=5.99
      khi2(3)=7.81
      khi2(4)=9.49
      khi2(5)=11.07
      khi2(6)=12.59
      khi2(7)=14.07
      khi2(8)=15.51
      khi2(9)=16.92
      khi2(10)=18.31
      ENDIF
c     On peut aussi changer le nom du fichier en entree (data.txt) et  **
c     le nom du fichier en sortie (resultat.txt)
      filer='data.txt'
c-------------------------------------------------------------
c     ------------------------------------------------------------------
c     DEBUT DU PROGRAMME
c     ------------------------------------------------------------------
      CALL cdARMA(dataf,paramf,nbcle,marret,Mind,sigma,mu,df1,df2,
     + lambda,loi5b,loi5k,loi6p,loi6q,loi7g,loi7d,loi8p,loi8q,loi9a,
     +loi9b,loi10b,loi10l,loi11a,loi11k,loi13g,loi13d,loi14l,loi16p,
     +loi16d,loi17p,loi17m,loi18g,loi18d,loi19a,loi19b,loi,p,q,rm,nT,
     + phi,phich,teta,tetach,donees,A,
     + B,R,NPAR,ICM,IW,PAR,W,CGETOL,V,G,CM,WORK,MEAN,PARHLD,EXACT,Wtip,
     + shocks,psi,phi2,teta2,YtpMn,Atnq,EspSU,EspSB,EspS,
     +E,NA,NB,NR,K,N2,IP,IQ,IK,MAXCAL,ISHOW,LWORK,LIW,QQ)
c Il faut faire modifier par cdARMA la valeur nbcle en entree pour lui faire sortir
c une nouvelle valeur nbcle qui est le nombre de bons echantillons conserves
c Ensuite, il faut utiliser cette valeur dans calcstat, comme etant la vraie valeur
```

```
c du nombre d'echantillons disponibles, la valeur de n que l'on a mise etant
c la valeur maximale possible de bons echantillons, donc les differentes matrices
     initialisees
c a l'aide de la valeur de n sont plus grandes que necessaires (elles auraient du etre
c initialisees avec la valeur de nbcle renvoyee par cd ARMA mais ce n'est pas possible
c avec fortran77) et il faut en tenir compte; donc bien regarder partout dans le
c programme calcstat la ou il y a n: OK CA C'EST FAIT!!! A METTRE EN COMMENTAIRES
     QUELQUEPART ...
          CALL calcstat(isa,khi,khi2,QuLed1,QuLe12,QuLed2,QuLe22,QuBD,
     + QuBD2,QuAD,QuAD2,QuJB,QuJB2,alpha,alpha2,Kp,p,q,nT,phi,teta,n,
     + nbcle,m,filer,filew,valeur,sch2vc,epscha,compt,compt2,statn,
     + KoLed1,KoLed2,Ledwi1,Ledwi2,snLed1,snLed2,stanBD,stanAD,stanJB,Z,
     + D,Davant,E2,U,hUetoi,Uavant,hKetoi,hU,hK,vecteu,vectoi,stat,
     + Zavant,BDa,BDb,BDc,BDd,Davan2,ADa,ADb,Puiss,Puiss2,snsort,snsorv,
     + Qucalc,snLe1s,snLe2s,snBDso,snADso,snJBso,matric,paramf,loi,
     + dnT,QLed1u,QLe12u,QLed2u,QLe22u,QBDu,QBD2u,QADu,QAD2u,QJBu,QJB2u)
          nbcle=nbclea
c      FAIRE ATTENTION:
c      voir si mes programmes ne modifient pas certains de leurs parametres en entree
c      ce qui pourrait ammener des erreurs au cours de la boucle que je vais faire dans
     le programme test.f
c      pour les differentes etapes 1), 2) et 3)
          CALL SYSTEM(efface)
c      Sauvegarde du seed
       OPEN(UNIT=14, FILE=paramf , STATUS='OLD', ACCESS='append' )
       WRITE(14,*) 'Sauvegarde du seed:'
       WRITE(14,*) 'XA:'
       WRITE(14,15) 'XA(1)=',XA(1)
       WRITE(14,15) 'XA(2)=',XA(2)
       WRITE(14,15) 'XA(3)=',XA(3)
       WRITE(14,15) 'XA(4)=',XA(4)
 15    FORMAT(A6,1F20.15)
       WRITE(14,*) 'IA:'
       WRITE(14,30) 'IA(1)=',IA(1)
       WRITE(14,30) 'IA(2)=',IA(2)
       WRITE(14,30) 'IA(3)=',IA(3)
       WRITE(14,30) 'IA(4)=',IA(4)
       WRITE(14,30) 'IA(5)=',IA(5)
       WRITE(14,30) 'IA(6)=',IA(6)
       WRITE(14,30) 'IA(7)=',IA(7)
       WRITE(14,30) 'IA(8)=',IA(8)
       WRITE(14,30) 'IA(9)=',IA(9)
 30       FORMAT(A6,1I10)
       CLOSE(UNIT=14)
       END
       INCLUDE 'mean.f'
       INCLUDE 'simulARMA.f'
       INCLUDE 'ARMpsi.f'
       INCLUDE 'shock.f'
       INCLUDE 'rskew.f'
       INCLUDE 'rlap.f'
       INCLUDE 'rpare.f'
       INCLUDE 'rspare.f'
       INCLUDE 'rSU.f'
       INCLUDE 'rTU.f'
       INCLUDE 'rSC.f'
       INCLUDE 'rLC.f'
       INCLUDE 'rSB.f'
       INCLUDE 'rS.f'
       INCLUDE 'H1etoile.f'
       INCLUDE 'H2etoile.f'
       INCLUDE 'H3etoile.f'
       INCLUDE 'H4etoile.f'
       INCLUDE 'H5etoile.f'
       INCLUDE 'H6etoile.f'
```

```
      INCLUDE 'H7etoile.f'
      INCLUDE 'H8etoile.f'
      INCLUDE 'H9etoile.f'
      INCLUDE 'H10etoile.f'
      INCLUDE 'H1isa.f'
      INCLUDE 'H2isa.f'
      INCLUDE 'H3isa.f'
      INCLUDE 'H4isa.f'
      INCLUDE 'H5isa.f'
      INCLUDE 'H6isa.f'
      INCLUDE 'H7isa.f'
      INCLUDE 'H8isa.f'
      INCLUDE 'H9isa.f'
      INCLUDE 'H10isa.f'
      INCLUDE 'H1.f'
      INCLUDE 'H2.f'
      INCLUDE 'H3.f'
      INCLUDE 'H4.f'
      INCLUDE 'H5.f'
      INCLUDE 'H6.f'
      INCLUDE 'H7.f'
      INCLUDE 'H8.f'
      INCLUDE 'H9.f'
      INCLUDE 'H10.f'
      INCLUDE 'qnorm.f'
      INCLUDE 'pnorm.f'
      INCLUDE 'min.f'
      INCLUDE 'max.f'
      INCLUDE 'var.f'
      INCLUDE 'creerdat_ARMA.f'
      INCLUDE 'calcstat.f'
```

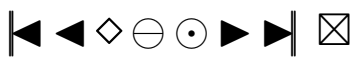
Programme big_prog_ARMA22.f

```
c Idee de ce  programme:
c Etape1)
c On cree un fichier de donnees 'data.txt' en utilisant certaines valeurs de
     parametres
c a lire dans un fichier d'entree 'ftemp' cree par le programme test.f a l'aide du
c fichier 'paramentree', on stocke les resultats autres que
c les donnees dans un fichier 'param.i'
c Etape 2)
c ensuite on utilise le fichier 'data.txt'
c avec le programme calcstat pour creer le fichier des statistiques 'resultat.i'
c Etape 3)
c Ensuite, on efface le fichier 'data.txt'
      PROGRAM main
c      Les ** indiquent les endroits ou des changements peuvent etre necessaires
c      ------------------------------------------------------------------------
c      DEBUT DE DECLARATION DES VARIABLES
c      ------------------------------------------------------------------------
c      Si isa=.TRUE. (ie MEAN=.TRUE.) on prend les polynomes modifies avec les ak  **
c      Si isa=.FALSE. (ie MEAN=.FALSE.) on prend les polynomes modifies sans les ak
      LOGICAL isa
      PARAMETER(isa=.FALSE.)
c-------------------------------------
c      parametres calcstat
c      Niveau 1 du test                                          **
      DOUBLE PRECISION alpha
      PARAMETER(alpha=0.1)
c      Niveau 2 du test                                          **
      DOUBLE PRECISION alpha2
      PARAMETER(alpha2=0.05)
c      Quantiles avec alpha
      DOUBLE PRECISION  khi(10)
```

```
c     Quantiles avec alpha2
      DOUBLE PRECISION  khi2(10)
c     Quantiles Ledwi1 et Ledwi2 avec alpha                    **
      DOUBLE PRECISION QuLed1, QuLed2
c     avec T=50
c      PARAMETER(QuLed1=3.69,QuLed2=5.466)
c     avec T=100
      PARAMETER(QuLed1=3.275,QuLed2=5.26)
c     avec T=200
c      PARAMETER(QuLed1=3.057,QuLed2=5.043)
c     Quantiles Ledwi1 et Ledwi2 avec alpha2                   **
      DOUBLE PRECISION QuLe12, QuLe22
c     avec T=50
c      PARAMETER(QuLe12=5.41,QuLe22=7.14)
c     avec T=100
      PARAMETER(QuLe12=5.20,QuLe22=6.97)
c     avec T=200
c      PARAMETER(QuLe12=4.751,QuLe22=6.796)
c     Quantiles Brockwell et Davis et Anderson Darling et JB avec alpha      **
      DOUBLE PRECISION QuBD, QuAD, QuJB
c     avec T=50
c      PARAMETER(QuBD=0.920,QuAD=1.743,QuJB=4.61)
c     avec T=100
      PARAMETER(QuBD=0.958,QuAD=1.743,QuJB=4.61)
c     avec T=200
c      PARAMETER(QuBD=0.978,QuAD=1.743,QuJB=4.61)
c     Quantiles Brockwell et Davis et Anderson Darling et JB avec alpha2      **
      DOUBLE PRECISION QuBD2, QuAD2, QuJB2
c     avec T=50
c      PARAMETER(QuBD2=0.899,QuAD2=2.308,QuJB2=5.99)
c      avec T=100
      PARAMETER(QuBD2=0.947,QuAD2=2.308,QuJB2=5.99)
c     avec T=200
c      PARAMETER(QuBD2=0.973,QuAD2=2.308,QuJB2=5.99)
c     Nombre de polynomes                                      **
      INTEGER Kp
      PARAMETER(Kp=10)
c     ordre du modele AR
      INTEGER p
      PARAMETER(p=2)
c     ordre du modele MA
      INTEGER q
      PARAMETER(q=2)
c     nombre d'observations dans mon echantillon                     **
c     Si on change la valeur de nT, il faut aller modifier
c     la valeur dans FORMAT a la fin du programme creerdat_ARMA.f
c     et dans le programme calcstat.f
      INTEGER nT
      PARAMETER(nT=100)
      INTEGER n,m
      PARAMETER(m=nT+1)
c     n=nombre de lignes au maximum dans le fichier de donnees       **
c     cela correspond a nbcle s'il n'y a pas eu d'erreurs
c     dans tous les cas mettre ici n=nbcle=nombre d'echantillons souhaites
      PARAMETER(n=10000)
      CHARACTER filer*8, filew*21                              **
      DOUBLE PRECISION valeur(n,m)
      DOUBLE PRECISION sch2vc(n)
      DOUBLE PRECISION epscha(n,nT)
      INTEGER compt(Kp)
      INTEGER compt2(Kp)
      DOUBLE PRECISION statn(n,Kp)
      INTEGER KoLed1(n)
      INTEGER KoLed2(n)
      DOUBLE PRECISION Ledwi1(Kp)
```

```
      DOUBLE PRECISION Ledwi2(Kp-1)
      DOUBLE PRECISION snLed1(n)
      DOUBLE PRECISION snLed2(n)
      DOUBLE PRECISION stanBD(n)
      DOUBLE PRECISION stanAD(n), stanJB(n)
      DOUBLE PRECISION Z(nT)
      DOUBLE PRECISION D(nT)
      DOUBLE PRECISION Davant(nT), E2(nT)
      DOUBLE PRECISION U(nT)
      DOUBLE PRECISION hUetoi(nT,Kp)
      DOUBLE PRECISION Uavant(nT)
      DOUBLE PRECISION hKetoi(Kp)
      DOUBLE PRECISION hU(nT,Kp)
      DOUBLE PRECISION hK(Kp)
      DOUBLE PRECISION vecteu(Kp)
      DOUBLE PRECISION vectoi(Kp)
      DOUBLE PRECISION stat(Kp)
      DOUBLE PRECISION Zavant(nT)
      DOUBLE PRECISION BDa(nT)
      DOUBLE PRECISION BDb(nT)
      DOUBLE PRECISION BDc(nT)
      DOUBLE PRECISION BDd(nT)
      DOUBLE PRECISION Davan2(nT)
      DOUBLE PRECISION ADa(nT)
      DOUBLE PRECISION ADb(nT)
      DOUBLE PRECISION Puiss(Kp)
      DOUBLE PRECISION Puiss2(Kp)
      DOUBLE PRECISION snsort(n,Kp)
      DOUBLE PRECISION snsorv(n)
      DOUBLE PRECISION Qucalc(Kp)
      DOUBLE PRECISION snLe1s(n)
      DOUBLE PRECISION snLe2s(n)
      DOUBLE PRECISION snBDso(n)
      DOUBLE PRECISION snADso(n), snJBso(n)
c-------------------------------------------------------------------
c     parametres creerdat
c     Nom du fichier de donnees en sortie                            **
      CHARACTER dataf*8
c     Nom du fichier de parametres en sortie                         **
      CHARACTER paramf*17
c     Nombre d'echantillons souhaites
      INTEGER nbcle,nbclea
c     rang d'arret dans la random shock method de Burn               **
      INTEGER marret
      PARAMETER(marret=200)
c     Induction period dans la methode de Burn                       **
      INTEGER Mind
      PARAMETER(Mind=200)
c     ecart-type des erreurs                                         **
      DOUBLE PRECISION sigma
      PARAMETER(sigma=1)
c     moyenne dans mon modele ARMA                                   **
      DOUBLE PRECISION mu
      PARAMETER(mu=0)
c     parametre de la khi-deux                                       **
      INTEGER df1
      PARAMETER(df1=4)
c     parametre de la student                                        **
      INTEGER df2
      PARAMETER(df2=5)
c     parametre de la skew-normale                                   **
      DOUBLE PRECISION lambda
      PARAMETER(lambda=2.0)
      DOUBLE PRECISION loi5b, loi5k, loi6p, loi6q
      DOUBLE PRECISION loi7g, loi7d, loi8p, loi8q, loi9a, loi9b
```


```
      DOUBLE PRECISION loi10b, loi10l, loi11a, loi11k, loi13g, loi13d
      DOUBLE PRECISION loi14l, loi16p, loi16d, loi17p, loi17m
      DOUBLE PRECISION loi18g, loi18d, loi19a, loi19b
      PARAMETER(loi5b=1.0, loi5k=2.0, loi6p=4, loi6q=1)                    **
      PARAMETER(loi7g=0.0,loi7d=1.0, loi8p=2, loi8q=2, loi9a=0,loi9b=2)      **
      PARAMETER(loi10b=0.2,loi10l=1, loi11a=2.0,loi11k=0.5)               **
      PARAMETER(loi13g=1.0,loi13d=1.0, loi14l=10)                        **
      PARAMETER(loi16p=0.05,loi16d=5.0, loi17p=0.05,loi17m=3.0)             **
      PARAMETER(loi18g=1.0,loi18d=1.0, loi19a=1.1, loi19b=0.5)            **
c     loi des erreurs
c     si loi=0 : Normale(0,sigma^2)
c     si loi=1 : Khi2 centree (df1)
c     si loi=2 : Student (df2)
c     si loi=3 : Skew-Normale(lambda)
c     si loi=4 : Laplace
c     si loi=5 : Weibull(b,k)
c     si loi=6 : Gamma(p,q)
c     si loi=7 : Log-Normale(g,d)
c     si loi=8 : Beta(p,q)
c     si loi=9 : Uniform(a,b)
c     si loi=10 : Shifted exp (l,b)
c     si loi=11 : Pareto(a,k)
c     si loi=12 : Shifted Pareto
c     si loi=13 : SU(g,d)
c     si loi=14 : TU(l)
c     si loi=15 : Logistic
c     si loi=16 : SC(p,d)
c     si loi=17 : LC(p,m)
c     si loi=18 : SB(g,d)
c     si loi=19 : S(a,b)
      INTEGER loi
c     rm=max(p,q)
      INTEGER rm
      PARAMETER(rm=2)
c     the autoregressive coefficients of the model
      DOUBLE PRECISION phi(2)
      DOUBLE PRECISION phich(p)
c     the moving-average coefficients of the model
      DOUBLE PRECISION teta(2)
      DOUBLE PRECISION tetach(q)
c     vecteur des donnees cree
      DOUBLE PRECISION donees(nT)
c Parametres pour G05EGF: simulation
c     the autoregressive coefficients of the model=phi     Input
      DOUBLE PRECISION A(p)
c     the moving-average coefficients of the model=teta     Input
      DOUBLE PRECISION B(q+1)
c     le vecteur des innovations     Output
      DOUBLE PRECISION R(nT)
c Parametres pour G13DCF: estimation
c     the number of initial parameter estimates   Input                    **
c     mettre p+q+1 si MEAN=.TRUE. , p+q si MEAN=.FALSE.
      INTEGER NPAR
      PARAMETER(NPAR=p+q)
c     first dimension of the array CM       Input                        **
c     mettre p+q+1 si MEAN=.TRUE. , p+q si MEAN=.FALSE.
      INTEGER ICM
      PARAMETER(ICM=p+q)
c     Workspace
      INTEGER IW(NPAR+rm+3)
c     initial parameter estimates     Input/Output
      DOUBLE PRECISION PAR(NPAR)
c     W(i,t) must be set equal to the observation at time t     Input
c     of the ith series
      DOUBLE PRECISION W(1,nT)
```

```
c     the accuracy to which the solution in PAR and QQ is required     Input        **
      DOUBLE PRECISION CGETOL
      PARAMETER(CGETOL=0.0001)
c     residual at time t for series i, for i = 1,2,...,k      Output
      DOUBLE PRECISION V(1,nT)
c     estimated first derivative of the log-likelihood function     Output
      DOUBLE PRECISION G(NPAR)
c     estimate of the correlation coefficient between the ith and     Output
c     jth elements in the PAR array
      DOUBLE PRECISION CM(ICM,NPAR)
c     Workspace
      DOUBLE PRECISION WORK((5+3*nT+2*rm)+(2*p+q+2*rm*(rm+2)+9)+
     +                    (NPAR+1)*(5*(NPAR+1)+29)/2+(rm+1)**2)
c     MEAN must be set to .TRUE. if components of mu are to     Input              **
c     be estimated and .FALSE. if all elements of mu are to be taken as zero
      LOGICAL MEAN
      PARAMETER(MEAN=.FALSE.)
c     PARHLD(i) must be set to .TRUE., if PAR(i) is to be        Input
c     held constant at its input value and .FALSE., if PAR(i) is a
c     free parameter, for i = 1,2,...,NPAR.
      LOGICAL PARHLD(NPAR)
c     EXACT must be set equal to .TRUE. if the user wishes     Input              **
c     the routine to compute exact maximum likelihood estimates.
c     EXACT must be set equal to .FALSE. if only conditional
c     likelihood estimates are required.
c     voir EXACT plus bas
      LOGICAL EXACT
c     Contient les p donnees initiales de la serie
      DOUBLE PRECISION Wtip(p)
c     marret+p innovations genere par shock.f
      DOUBLE PRECISION shocks(marret+p)
c     defini dans Burn, calcule par ARMpsi.f
      DOUBLE PRECISION psi(marret+1)
c     utile dans simARM.f de longueur marret
      DOUBLE PRECISION phi2(marret)
c     utile dans simARM.f de longueur marret
      DOUBLE PRECISION teta2(marret)
c     utile dans simARM.f
      DOUBLE PRECISION YtpMn(p+Mind+nT)
c     utile dans simARM.f
      DOUBLE PRECISION Atnq(nT+Mind+q)
c     Esperances des lois SU, SB et S                              **
      DOUBLE PRECISION EspSU,EspSB,EspS
      PARAMETER(EspSU=-1.93758, EspSB=0.696735, EspS=0)
      CHARACTER*13 efface
      DOUBLE PRECISION temp(6)
      INTEGER j
      CHARACTER*5 ftemp
      CHARACTER*1 INTCH1
      CHARACTER*2 INTCH2
      CHARACTER*3 INTCH3
      CHARACTER*4 INTCH4
      DOUBLE PRECISION matric(n,18)
      INTEGER dnT
      PARAMETER(dnT=Kp)
      DOUBLE PRECISION QLed1u,QLe12u
      PARAMETER(QLed1u=QuLed1,QLe12u=QuLe12)
      DOUBLE PRECISION QLed2u,QLe22u
      PARAMETER(QLed2u=QuLed2,QLe22u=QuLe22)
      DOUBLE PRECISION QBDu,QBD2u
      PARAMETER(QBDu=QuBD,QBD2u=QuBD2)
      DOUBLE PRECISION QADu,QAD2u
      PARAMETER(QADu=QuAD,QAD2u=QuAD2)
      DOUBLE PRECISION QJBu,QJB2u
      PARAMETER(QJBu=QuJB,QJB2u=QuJB2)
```


```
c     the mean of the time series      Input
      DOUBLE PRECISION E
      PARAMETER(E=mu)
c     the number of autoregressive coefficients supplied      Input
      INTEGER NA
      PARAMETER(NA=p)
c     the number of moving-average coefficients supplied      Input
      INTEGER NB
      PARAMETER(NB=q+1)
c     the dimension of the array R: vecteur des innovations      Input
      INTEGER NR
      PARAMETER(NR=12)
c     the number of observed time series, k  (chez moi k=1)  Input
      INTEGER K
      PARAMETER(K=1)
c     the number of observations in each time series, n (chez moi=nT)   Input
      INTEGER N2
      PARAMETER(N2=nT)
c     the number of AR parameter matrices, p     Input
      INTEGER IP
      PARAMETER(IP=p)
c     the number of MA parameter matrices, q    Input
      INTEGER IQ
      PARAMETER(IQ=q)
c     the first dimension of the arrays QQ, W and V     Input
      INTEGER IK
      PARAMETER(IK=1)
c     the maximum number of likelihood evaluations to be     Input
c     permitted by the search procedure
      INTEGER MAXCAL
      PARAMETER(MAXCAL=40*NPAR*(NPAR+5))
c     which quantities are to be printed     Input
      INTEGER ISHOW
      PARAMETER(ISHOW=0)
c     dimension of the array WORK     Input
      INTEGER LWORK
      PARAMETER(LWORK=(5+3*(nT)+2*rm)+(2*p+q+2*rm*(rm+2)+9)+
     +                    (NPAR+1)*(5*(NPAR+1)+29)/2+(rm+1)**2)
c     dimension of the array IW     Input
      INTEGER LIW
      PARAMETER(LIW=NPAR+rm+3)
c     QQ(i,j) must be set equal to an initial estimate of     Input/Output
c     the (i,j)th element of the covariance matrix of the residual series
      DOUBLE PRECISION QQ(IK,K)
      INTEGER NI, NX, IFAIL
      PARAMETER(NX=4,NI=9)
      INTEGER IA(NI)
      DOUBLE PRECISION XA(NX)
c     seed=1 (non-repeatable sequence) ou seed=0 (repeatable sequence)
     **
      INTEGER seed
      PARAMETER(seed=0)
c-----------------------------------------------------------------
c Fin de declaration des variables
c-----------------------------------------------------------------
c G05CBF Initialise random number generating routines to give repeatable sequence
c G05CCF Initialise random number generating routines to give non-repeatable sequence
c G05CFF Save state of random number generating routines
c G05CGF Restore state of random number generating routines
      IF (seed .EQ. 1) THEN
      CALL G05CCF
      ENDIF
      IF (seed .EQ. 0) THEN
      CALL G05CBF(0)
      ENDIF
```

```
      IFAIL=0
c     Rajouter ici, si voulu, les IA et les XA a prendre a la fin du fichier para.ijkl
     :          **
c     A (de)commenter si necessaire (si on rajoute les IA et XA, on decommente)
c      CALL G05CGF(IA,NI,XA,NX,IFAIL)
      CALL G05CFF(IA,NI,XA,NX,IFAIL)
      EXACT=.TRUE.                                                    **
      nbcle=n
      efface='rm ./data.txt'
      ftemp='ftemp'
c     On va lire les parametres necessaires dans ftemp
c     temp va contenir sur chaque ligne: j, loi, phi(1), phi(2), teta(1), teta(2)
      OPEN(UNIT=15, FILE=ftemp , STATUS='OLD' )
      READ(15,*,END=10) (temp(j),j=1,6)
      CLOSE(15)
 10   j=temp(1)
      loi=temp(2)
      phi(1)=temp(3)
      phi(2)=temp(4)
      teta(1)=temp(5)
      teta(2)=temp(6)
c     On sauvegarde la valeur de nbcle
      nbclea=nbcle
c     Nom du fichier de donnees en sortie                                **
      dataf='data.txt'
c     Permet de convertir l'entier j en chaine de caracteres
      IF (j/10 .LT. 1) THEN
         WRITE(INTCH1, '(I1)') j
c     Nom du fichier de parametres en sortie                             **
         paramf='./SIMUL/para.'//'000'//INTCH1
         filew='./SIMUL/resultat.'//'000'//INTCH1
      ENDIF
      IF ((j/10 .EQ. 1) .OR. ((j/10 .GT. 1) .AND. (j/10 .LT. 10))) THEN
         WRITE(INTCH2, '(I2)') j
c     Nom du fichier de parametres en sortie                             **
         paramf='./SIMUL/para.'//'00'//INTCH2
         filew='./SIMUL/resultat.'//'00'//INTCH2
      ENDIF
      IF ((j/10 .EQ. 10) .OR. ((j/10 .GT. 10) .AND. (j/10 .LT. 100)))
     + THEN
         WRITE(INTCH3, '(I3)') j
c     Nom du fichier de parametres en sortie                             **
         paramf='./SIMUL/para.'//'0'//INTCH3
         filew='./SIMUL/resultat.'//'0'//INTCH3
      ENDIF
      IF ((j/10 .EQ. 100) .OR. ((j/10 .GT. 100) .AND. (j/10 .LT. 1000)))
     + THEN
         WRITE(INTCH4, '(I4)') j
c     Nom du fichier de parametres en sortie                             **
         paramf='./SIMUL/para.'//INTCH4
         filew='./SIMUL/resultat.'//INTCH4
      ENDIF
      IF (loi .EQ. 0) THEN
c     Quantiles de la loi du Khi2 (K=10;alpha=0.1)                    **
      khi(1)=2.71
      khi(2)=4.61
      khi(3)=6.25
      khi(4)=7.78
      khi(5)=9.24
      khi(6)=10.64
      khi(7)=12.02
      khi(8)=13.36
      khi(9)=14.68
      khi(10)=15.99
c     Quantiles de la loi du Khi2 (K=10;alpha=0.05)                    **
```

```
      khi2(1)=3.84
      khi2(2)=5.99
      khi2(3)=7.81
      khi2(4)=9.49
      khi2(5)=11.07
      khi2(6)=12.59
      khi2(7)=14.07
      khi2(8)=15.51
      khi2(9)=16.92
      khi2(10)=18.31
      ENDIF
      IF (loi .NE. 0) THEN
c     Quantiles utilises pour le calcul de la stat(vrais quantiles calcules sous H0)
     **
c     avec alpha=0.1
      khi(1)=2.71
      khi(2)=4.61
      khi(3)=6.25
      khi(4)=7.78
      khi(5)=9.24
      khi(6)=10.64
      khi(7)=12.02
      khi(8)=13.36
      khi(9)=14.68
      khi(10)=15.99
c     Quantiles utilises pour le calcul de la stat(vrais quantiles calcules sous H0)
     **
c     avec alpha=0.05
      khi2(1)=3.84
      khi2(2)=5.99
      khi2(3)=7.81
      khi2(4)=9.49
      khi2(5)=11.07
      khi2(6)=12.59
      khi2(7)=14.07
      khi2(8)=15.51
      khi2(9)=16.92
      khi2(10)=18.31
      ENDIF
c     On peut aussi changer le nom du fichier en entree (data.txt) et      **
c     le nom du fichier en sortie (resultat.txt)
      filer='data.txt'
c-------------------------------------------------------------
c     ------------------------------------------------------------------------
c     DEBUT DU PROGRAMME
c     ------------------------------------------------------------------------
      CALL cdARMA(dataf,paramf,nbcle,marret,Mind,sigma,mu,df1,df2,
     + lambda,loi5b,loi5k,loi6p,loi6q,loi7g,loi7d,loi8p,loi8q,loi9a,
     +loi9b,loi10b,loi10l,loi11a,loi11k,loi13g,loi13d,loi14l,loi16p,
     +loi16d,loi17p,loi17m,loi18g,loi18d,loi19a,loi19b,loi,p,q,rm,nT,
     + phi,phich,teta,tetach,donees,A,
     + B,R,NPAR,ICM,IW,PAR,W,CGETOL,V,G,CM,WORK,MEAN,PARHLD,EXACT,Wtip,
     + shocks,psi,phi2,teta2,YtpMn,Atnq,EspSU,EspSB,EspS,
     +E,NA,NB,NR,K,N2,IP,IQ,IK,MAXCAL,ISHOW,LWORK,LIW,QQ)
c Il faut faire modifier par cdARMA la valeur nbcle en entree pour lui faire sortir
c une nouvelle valeur nbcle qui est le nombre de bons echantillons conserves
c Ensuite, il faut utiliser cette valeur dans calcstat, comme etant la vraie valeur
c du nombre d'echantillons disponibles, la valeur de n que l'on a mise etant
c la valeur maximale possible de bons echantillons, donc les differentes matrices
     initialisees
c a l'aide de la valeur de n sont plus grandes que necessaires (elles auraient du etre
c initialisees avec la valeur de nbcle renvoyee par cd ARMA mais ce n'est pas possible
c avec fortran77) et il faut en tenir compte; donc bien regarder partout dans le
c programme calcstat la ou il y a n: OK CA C'EST FAIT!!! A METTRE EN COMMENTAIRES
     QUELQUEPART ...
```


```
      CALL calcstat(isa,khi,khi2,QuLed1,QuLe12,QuLed2,QuLe22,QuBD,
     + QuBD2,QuAD,QuAD2,QuJB,QuJB2,alpha,alpha2,Kp,p,q,nT,phi,teta,n,
     + nbcle,m,filer,filew,valeur,sch2vc,epscha,compt,compt2,statn,
     + KoLed1,KoLed2,Ledwi1,Ledwi2,snLed1,snLed2,stanBD,stanAD,stanJB,Z,
     + D,Davant,E2,U,hUetoi,Uavant,hKetoi,hU,hK,vecteu,vectoi,stat,
     + Zavant,BDa,BDb,BDc,BDd,Davan2,ADa,ADb,Puiss,Puiss2,snsort,snsorv,
     + Qucalc,snLe1s,snLe2s,snBDso,snADso,snJBso,matric,paramf,loi,
     + dnT,QLed1u,QLe12u,QLed2u,QLe22u,QBDu,QBD2u,QADu,QAD2u,QJBu,QJB2u)
        nbcle=nbclea
c     FAIRE ATTENTION:
c     voir si mes programmes ne modifient pas certains de leurs parametres en entree
c     ce qui pourrait ammener des erreurs au cours de la boucle que je vais faire dans
     le programme test.f
c     pour les differentes etapes 1), 2) et 3)
        CALL SYSTEM(efface)
c     Sauvegarde du seed
      OPEN(UNIT=14, FILE=paramf , STATUS='OLD', ACCESS='append' )
      WRITE(14,*) 'Sauvegarde du seed:'
      WRITE(14,*) 'XA:'
      WRITE(14,15) 'XA(1)=',XA(1)
      WRITE(14,15) 'XA(2)=',XA(2)
      WRITE(14,15) 'XA(3)=',XA(3)
      WRITE(14,15) 'XA(4)=',XA(4)
 15   FORMAT(A6,1F20.15)
      WRITE(14,*) 'IA:'
      WRITE(14,30) 'IA(1)=',IA(1)
      WRITE(14,30) 'IA(2)=',IA(2)
      WRITE(14,30) 'IA(3)=',IA(3)
      WRITE(14,30) 'IA(4)=',IA(4)
      WRITE(14,30) 'IA(5)=',IA(5)
      WRITE(14,30) 'IA(6)=',IA(6)
      WRITE(14,30) 'IA(7)=',IA(7)
      WRITE(14,30) 'IA(8)=',IA(8)
      WRITE(14,30) 'IA(9)=',IA(9)
 30     FORMAT(A6,1I10)
      CLOSE(UNIT=14)
      STOP
      END
      INCLUDE 'mean.f'
      INCLUDE 'simulARMA.f'
      INCLUDE 'ARMpsi.f'
      INCLUDE 'shock.f'
      INCLUDE 'rskew.f'
      INCLUDE 'rlap.f'
      INCLUDE 'rpare.f'
      INCLUDE 'rspare.f'
      INCLUDE 'rSU.f'
      INCLUDE 'rTU.f'
      INCLUDE 'rSC.f'
      INCLUDE 'rLC.f'
      INCLUDE 'rSB.f'
      INCLUDE 'rS.f'
      INCLUDE 'H1etoile.f'
      INCLUDE 'H2etoile.f'
      INCLUDE 'H3etoile.f'
      INCLUDE 'H4etoile.f'
      INCLUDE 'H5etoile.f'
      INCLUDE 'H6etoile.f'
      INCLUDE 'H7etoile.f'
      INCLUDE 'H8etoile.f'
      INCLUDE 'H9etoile.f'
      INCLUDE 'H10etoile.f'
      INCLUDE 'H1isa.f'
      INCLUDE 'H2isa.f'
      INCLUDE 'H3isa.f'
```

```
      INCLUDE 'H4isa.f'
      INCLUDE 'H5isa.f'
      INCLUDE 'H6isa.f'
      INCLUDE 'H7isa.f'
      INCLUDE 'H8isa.f'
      INCLUDE 'H9isa.f'
      INCLUDE 'H10isa.f'
      INCLUDE 'H1.f'
      INCLUDE 'H2.f'
      INCLUDE 'H3.f'
      INCLUDE 'H4.f'
      INCLUDE 'H5.f'
      INCLUDE 'H6.f'
      INCLUDE 'H7.f'
      INCLUDE 'H8.f'
      INCLUDE 'H9.f'
      INCLUDE 'H10.f'
      INCLUDE 'qnorm.f'
      INCLUDE 'pnorm.f'
      INCLUDE 'min.f'
      INCLUDE 'max.f'
      INCLUDE 'var.f'
      INCLUDE 'creerdat_ARMA.f'
      INCLUDE 'calcstat.f'
```

A.6. LES PROGRAMMES CREERDAT_ARMAPQ

Programmes creerdat_ARMA00.f

```
c      Debut-Commentaires
c      Nom de la sous-routine: cdzero
c Entrees:
c--------
c voir plus bas
c Sorties:
c--------
c Le fichier data.txt
c Description:
c-----------
c Modele ARMA(0,0)
c Ce programme cree le fichier de donnees data.txt qui contient, en ligne:
c sigch2 et epschap
c Utilisation dans une fonction main:
c-------------------------------------
c       PROGRAM main
cc       Les ** indiquent les endroits ou des changements peuvent etre necessaires
cc       Nom du fichier de donnees en sortie                                    **
c       CHARACTER dataf*8
cc       Nom du fichier de parametres en sortie                                 **
c       CHARACTER paramf*17
cc       Nombre d'echantillons souhaites                                       **
c       INTEGER nbcle
c       PARAMETER(nbcle=10)
cc       ecart-type des erreurs                                                **
c       DOUBLE PRECISION sigma
c       PARAMETER(sigma=1)
cc       moyenne dans mon modele ARMA                                          **
c       DOUBLE PRECISION mu
c       PARAMETER(mu=0)
cc       parametre de la khi-deux                                              **
c       INTEGER df1
c       PARAMETER(df1=2)
cc       parametre de la student                                               **
c       INTEGER df2
c       PARAMETER(df2=5)
```

```
cc       parametre de la skew-normale                                    **
c       DOUBLE PRECISION lambda
c       PARAMETER(lambda=2.0)
c       DOUBLE PRECISION loi5b, loi5k, loi6p, loi6q
c       DOUBLE PRECISION loi7g, loi7d, loi8p, loi8q, loi9a, loi9b
c       DOUBLE PRECISION loi10b, loi10l, loi11a, loi11k, loi13g, loi13d
c       DOUBLE PRECISION loi14l, loi16p, loi16d, loi17p, loi17m
c       DOUBLE PRECISION loi18g, loi18d, loi19a, loi19b
c       PARAMETER(loi18g=1.0,loi18d=1.0, loi19a=1.1, loi19b=0.5)
c       PARAMETER(loi16p=0.2,loi16d=5.0, loi17p=0.2,loi17m=3.0)
c       PARAMETER(loi10b=0.2,loi10l=1, loi11a=2.0,loi11k=0.5)
c       PARAMETER(loi13g=1.0,loi13d=1.0, loi14l=0.7)
c       PARAMETER(loi7g=0,loi7d=1, loi8p=2, loi8q=2, loi9a=0,loi9b=2)
c       PARAMETER(loi5b=1, loi5k=1.8, loi6p=4, loi6q=1)
cc       loi des erreurs                                                 **
cc       si loi=0 : Normale(0,sigma^2)
cc       si loi=1 : Khi2 centree (df1)
cc       si loi=2 : Student (df2)
cc       si loi=3 : Skew-Normale(lambda)
cc       si loi=4 : Laplace
cc       si loi=5 : Weibull(b,k)
cc       si loi=6 : Gamma(p,q)
cc       si loi=7 : Log-Normale(g,d)
cc       si loi=8 : Beta(p,q)
cc       si loi=9 : Uniform(a,b)
cc       si loi=10 : Shifted exp (l,b)
cc       si loi=11 : Pareto(a,k)
cc       si loi=12 : Shifted Pareto
cc       si loi=13 : SU(g,d)
cc       si loi=14 : TU(l)
cc       si loi=15 : Logistic
cc       si loi=16 : SC(p,d)
cc       si loi=17 : LC(p,m)
cc       si loi=18 : SB(g,d)
cc       si loi=19 : S(a,b)
c       INTEGER loi
c       PARAMETER(loi=2)
cc       the number of autoregressive coefficients supplied              **
c       INTEGER p
c       PARAMETER(p=0)
cc       the number of moving-average coefficients supplied              **
c       INTEGER q
c       PARAMETER(q=0)
cc       nombre d'observations dans mon echantillon                      **
cc       Si on change la valeur de nT, il faut aller modifier
cc       la valeur dans FORMAT a la fin du programme creerdat_ARMA.f
c       INTEGER nT
c       PARAMETER(nT=100)
cc       vecteur des donnees cree
c       DOUBLE PRECISION donees(nT)
c       DOUBLE PRECISION EspSU,EspSB,EspS
cc-------------------------------------------------------------------
cc Fin des declarations des variables
cc-------------------------------------------------------------------
cc       Nom du fichier de donnees en sortie
     **
c        dataf='data.txt'
cc       Nom du fichier de parametres en sortie
     **
c        paramf='para.txt'
c        CALL cdzero(dataf,paramf,nbcle,sigma,mu,df1,df2,
c       + lambda,loi5b,loi5k,loi6p,loi6q,loi7g,loi7d,loi8p,loi8q,loi9a,
c       +loi9b,loi10b,loi10l,loi11a,loi11k,loi13g,loi13d,loi14l,loi16p,
c       +loi16d,loi17p,loi17m,loi18g,loi18d,loi19a,loi19b,loi,p,q,nT,
c       + donees,EspSU,EspSB,EspS)
```


```
c       END
c       INCLUDE 'mean.f'
c       INCLUDE 'simulARMA.f'
c       INCLUDE 'ARMpsi.f'
c       INCLUDE 'shock.f'
c       INCLUDE 'rskew.f'
c       INCLUDE 'rlap.f'
c       INCLUDE 'rpare.f'
c       INCLUDE 'rspare.f'
c       INCLUDE 'rSU.f'
c       INCLUDE 'rTU.f'
c       INCLUDE 'rSC.f'
c       INCLUDE 'rLC.f'
c       INCLUDE 'rSB.f'
c       INCLUDE 'rS.f'
c       INCLUDE 'creerdat_ARMA00.f'
c-------------------------------------------
c Instructions de compilation: alias f77='fort77'
c f77 -c nom_du_fichier_contenant_la_fonction_main.f
c f77 -c creerdat_ARMA.f -lnag
c f77 nom_du_fichier_contenant_la_fonction_main.o creerdat_ARMA.o -lnag
c Fonctions exterieures appelees:
c G05EGF, G05EWF et  G13DCF de la librairie NAG Mark16 et simARMA.f
c Auteur: Pierre Lafaye de Micheaux
c Date: 15/02/2001
c Fin-Commentaires
      SUBROUTINE cdzero(dataf,paramf,nbcle,sigma,mu,df1,df2,
     + lambda,loi5b,loi5k,loi6p,loi6q,loi7g,loi7d,loi8p,loi8q,loi9a,
     +loi9b,loi10b,loi10l,loi11a,loi11k,loi13g,loi13d,loi14l,loi16p,
     +loi16d,loi17p,loi17m,loi18g,loi18d,loi19a,loi19b,loi,p,q,nT,
     + donees,EspSU,EspSB,EspS)
c-------------------------------------------------------
c Debut des Declarations des variables
c-------------------------------------------------------
      DOUBLE PRECISION ybarre,meanp
c     Nom du fichier de donnees en sortie
      CHARACTER dataf*8
c     Nom du fichier de parametres en sortie
      CHARACTER paramf*17
      INTEGER i,j
c     Precision machine
      DOUBLE PRECISION preci, X02AJF
      EXTERNAL X02AJF
c     Temps de calcul
      DOUBLE PRECISION CPTIME, S1, S2, X05BAF
      EXTERNAL X05BAF
c     Nombre d'echantillons souhaites
      INTEGER nbcle
c     ecart-type des erreurs
      DOUBLE PRECISION sigma
      DOUBLE PRECISION sigch2
c     moyenne dans mon modele ARMA
      DOUBLE PRECISION mu
      DOUBLE PRECISION much
c     parametre de la khi-deux
      INTEGER df1
c     parametre de la student
      INTEGER df2
c     parametre de la skew-normale
      DOUBLE PRECISION lambda
      DOUBLE PRECISION loi5b, loi5k, loi6p, loi6q
      DOUBLE PRECISION loi7g, loi7d, loi8p, loi8q, loi9a, loi9b
      DOUBLE PRECISION loi10b, loi10l, loi11a, loi11k, loi13g, loi13d
      DOUBLE PRECISION loi14l, loi16p, loi16d, loi17p, loi17m
      DOUBLE PRECISION loi18g, loi18d, loi19a, loi19b
```

```
c     loi des erreurs
c     si loi=0 : Normale(0,sigma^2)
c     si loi=1 : Khi2 centree (df1)
c     si loi=2 : Student (df2)
c     si loi=3 : Skew-Normale(lambda)
c     si loi=4 : Laplace
c     si loi=5 : Weibull(b,k)
c     si loi=6 : Gamma(p,q)
c     si loi=7 : Log-Normale(g,d)
c     si loi=8 : Beta(p,q)
c     si loi=9 : Uniform(a,b)
c     si loi=10 : Shifted exp (l,b)
c     si loi=11 : Pareto(a,k)
c     si loi=12 : Shifted Pareto
c     si loi=13 : SU(g,d)
c     si loi=14 : TU(l)
c     si loi=15 : Logistic
c     si loi=16 : SC(p,d)
c     si loi=17 : LC(p,m)
c     si loi=18 : SB(g,d)
c     si loi=19 : S(a,b)
      INTEGER loi
c     the number of autoregressive coefficients supplied
      INTEGER p
c     the number of moving-average coefficients supplied
      INTEGER q
c     nombre d'observations dans mon echantillon
c     Si on change la valeur de nT, il faut aller modifier
c     la valeur dans FORMAT a la fin du programme
      INTEGER nT
c     vecteur des donnees cree
      DOUBLE PRECISION donees(nT)
      DOUBLE PRECISION pi, a, b, Eweibu ,EspSU, EspSB, EspS
      DOUBLE PRECISION G05DEF, G05DHF
      DOUBLE PRECISION G05DJF, G05DPF, G05DCF, S14AAF
      DOUBLE PRECISION G05DAF, G05DBF, rS
c-----------------------------------------------------------
c Fin des Declarations des variables
c-----------------------------------------------------------
c-----------------------------------------------------------------------
c                              Debut du programme
c-----------------------------------------------------------------------
      S1=X05BAF()
      OPEN(UNIT=12, FILE=dataf , STATUS='NEW' )
c-----------------------------------------------------------------------

c                              q>0 et p>0: modele ARMA(p,q)
c-----------------------------------------------------------------------

      preci=X02AJF()
      DO 1000, j=1,nbcle
      WRITE(6,*) nbcle-j
c-----------------------------------------------------------------------

c                              On simule les donnees:
c-----------------------------------------------------------------------

c donees=[Y1,...,YT]
      pi=DBLE(3.14159265358979)
      a=DBLE(0.0)
      b=DBLE(1.0)
      IF (loi .EQ. 0) THEN
            CALL G05FDF(mu,sigma,nT,donees)
      DO 10, i=1,nT
            donees(i)=donees(i)-mu
```

```
 10      CONTINUE
       ENDIF
       IF (loi .EQ. 1) THEN
          IFAIL=0
          DO 20, i=1,nT
             donees(i)=G05DHF(df1,IFAIL)-DBLE(df1)
 20       CONTINUE
       ENDIF
       IF (loi .EQ. 2) THEN
          IFAIL=0
c                                        l'esperance d'une student est nulle
          DO 30, i=1,nT
             donees(i)=G05DJF(df2,IFAIL)
 30       CONTINUE
       ENDIF
       IF (loi .EQ. 3) THEN
          CALL rskew(nT,donees,lambda)
          DO 40, i=1,nT
             donees(i)=donees(i) -
     +           dsqrt(DBLE(2.0)/pi)*(lambda/
     +       (dsqrt(DBLE(1.0)+lambda*lambda)))
 40       CONTINUE
       ENDIF
       IF (loi .EQ. 4) THEN
c                          l'esperance d'une loi de Laplace est nulle
          CALL rlap(nT,donees)
       ENDIF
       IF (loi .EQ. 5) THEN
          IFAIL=0
          Eweibu=S14AAF(DBLE(1+1/loi5k),IFAIL)/loi5b
          IFAIL=0
          DO 50, i=1,nT
             donees(i)=G05DPF(loi5k,(loi5b)**(-loi5k),IFAIL)-Eweibu
 50       CONTINUE
       ENDIF
       IF (loi .EQ. 6) THEN
          IFAIL=0
          CALL G05FFF(loi6p,loi6q,nT,donees,IFAIL)
          DO 60, i=1,nT
             donees(i)=donees(i) -loi6p*loi6q
 60       CONTINUE
       ENDIF
       IF (loi .EQ. 7) THEN
          DO 70, i=1,nT
             donees(i)=G05DEF(-loi7g/loi7d,1/loi7d)
     +                        -dexp(-loi7g/loi7d+DBLE(0.5)/(loi7d**2))
 70       CONTINUE
       ENDIF
       IF (loi .EQ. 8) THEN
          IFAIL=0
          CALL G05FEF(loi8p,loi8q,nT,donees,IFAIL)
          DO 80, i=1,nT
             donees(i)=donees(i)-loi8p/(loi8p+loi8q)
 80       CONTINUE
       ENDIF
       IF (loi .EQ. 9) THEN
          DO 90, i=1,nT
             donees(i)=G05DAF(loi9a,loi9b)-(loi9a+loi9b)/DBLE(2.0)
 90       CONTINUE
       ENDIF
       IF (loi .EQ. 10) THEN
          DO 100, i=1,nT
             donees(i)=G05DBF(1/loi10b)-DBLE(1.0)/loi10b
 100       CONTINUE
       ENDIF
```


```
      IF (loi .EQ. 11) THEN
         CALL rpare(loi11a,loi11k,nT,donees)
         DO 110, i=1,nT
            donees(i)=donees(i)-loi11k*loi11a/(loi11a-DBLE(1.0))
 110        CONTINUE
      ENDIF
      IF (loi .EQ. 12) THEN
         CALL rspare(nT,donees)
         DO 120, i=1,nT
            donees(i)=donees(i)-DBLE(1)
 120        CONTINUE
      ENDIF
      IF (loi .EQ. 13) THEN
         CALL rSU(loi13g,loi13d,nT,donees)
         DO 130, i=1,nT
            donees(i)=donees(i)-EspSU
 130        CONTINUE
      ENDIF
      IF (loi .EQ. 14) THEN
c                        l'esperance d'une loi TU(l) = 0
         CALL rTU(loi14l,nT,donees)
      ENDIF
      IF (loi .EQ. 15) THEN
         DO 140, i=1,nT
c                 l'esperance d'une loi Logistic = 0
            donees(i)=G05DCF(a,b)
 140         CONTINUE
      ENDIF
      IF (loi .EQ. 16) THEN
c                 l'esperance d'une loi SC(p,d) = 0
         CALL rSC(loi16p,loi16d,nT,donees)
      ENDIF
      IF (loi .EQ. 17) THEN
         CALL rLC(loi17p,loi17m,nT,donees)
         DO 150, i=1,nT
            donees(i)=donees(i) -loi17p*loi17m
 150      CONTINUE
      ENDIF
      IF (loi .EQ. 18) THEN
         CALL rSB(loi18g,loi18d,nT,donees)
         DO 160, i=1,nT
            donees(i)=donees(i)-EspSB
 160        CONTINUE
      ENDIF
      IF (loi .EQ. 19) THEN
         DO 170, i=1,nT
            donees(i)=rS(loi19a,loi19b)-EspS
 170         CONTINUE
      ENDIF
      ybarre=meanp(nT,donees)
      sigch2=DBLE(0.0)
      DO 180, i=1,nT
         sigch2=sigch2+(donees(i))**2
 180     CONTINUE
         sigch2=sigch2/(DBLE(nT))
      much=ybarre
c-----------------------------------------------------------------------------------------------
c                                    On estime les parametres:
c-----------------------------------------------------------------------------------------------

c     On ecrit toutes les donnees dans le fichier
      WRITE(12,200) sigch2, donees
 200  FORMAT(SP,51D20.12)                                                    **
c     FIN DE LA BOUCLE
```

```
1000  CONTINUE
      CLOSE(UNIT=12)
      S2=X05BAF()
      CPTIME=S2-S1
      OPEN(UNIT=14, FILE=paramf , STATUS='NEW' )
      WRITE(14,*) 'Fichier '// paramf
      WRITE(14,*) 'p=',p
      WRITE(14,*) 'q=',q
      WRITE(14,*) 'nT=',nT
      WRITE(14,*) 'loi=',loi
      WRITE(14,*) 'nbcle=',nbcle
      WRITE(14,*) 'sigma=',sigma
      WRITE(14,*) 'mu=', mu
      WRITE(14,*) 'df1=',df1
      WRITE(14,*) 'df2=',df2
      WRITE(14,*) 'lambda=',lambda
      WRITE(14,*) 'loi5b=',loi5b
      WRITE(14,*) 'loi5k=',loi5k
      WRITE(14,*) 'loi6p=',loi6p
      WRITE(14,*) 'loi6q=',loi6q
      WRITE(14,*) 'loi7g=',loi7g
      WRITE(14,*) 'loi7d=',loi7d
      WRITE(14,*) 'loi8p=',loi8p
      WRITE(14,*) 'loi8q=',loi8q
      WRITE(14,*) 'loi9a=',loi9a
      WRITE(14,*) 'loi9b=',loi9b
      WRITE(14,*) 'loi10l=',loi10l
      WRITE(14,*) 'loi10b=',loi10b
      WRITE(14,*) 'loi11a=',loi11a
      WRITE(14,*) 'loi11k=',loi11k
      WRITE(14,*) 'loi13g=',loi13g
      WRITE(14,*) 'loi13d=',loi13d
      WRITE(14,*) 'loi14l=',loi14l
      WRITE(14,*) 'loi16p=',loi16p
      WRITE(14,*) 'loi16d=',loi16d
      WRITE(14,*) 'loi17p=',loi17p
      WRITE(14,*) 'loi17m=',loi17m
      WRITE(14,*) 'loi18g=',loi18g
      WRITE(14,*) 'loi18d=',loi18d
      WRITE(14,*) 'loi19a=',loi19a
      WRITE(14,*) 'loi19b=',loi19b
      WRITE(14,*) 'Precision machine=', preci
      WRITE(14,*) 'QUELQUES RESULTATS:'
      WRITE(14,*) 'Temps de calcul de creation des donnees:', CPTIME
      CLOSE(UNIT=14)
      END
c-----------------------------------------------------------------------
c                                Fin du programme
c-----------------------------------------------------------------------
```

Programmes creerdat_AR.f

```
c     Debut-Commentaires
c     Nom de la sous-routine: cdatAR
c Entrees:
c--------
c voir plus bas
c Sorties:
c--------
c Le fichier data.txt
c Description:
c-----------
c Modele AR(p)
c Ce programme cree le fichier de donnees data.txt qui contient, en ligne:
c sigch2 et epschap
```

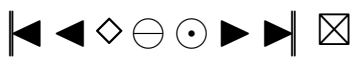
```
c Utilisation dans une fonction main:
c-----------------------------------------
c       PROGRAM main
cc       Les ** indiquent les endroits ou des changements peuvent etre necessaires
cc       Nom du fichier en sortie                                          **
c       CHARACTER dataf*8
cc       Nom du fichier de parametres en sortie                             **
c       CHARACTER paramf*17
cc       Nombre d'echantillons souhaites                                   **
c       INTEGER nbcle
c       PARAMETER(nbcle=10)
cc       rang d'arret dans la random shock method de Burn                  **
c       INTEGER marret
c       PARAMETER(marret=200)
cc       Induction period dans la methode de Burn                          **
c       INTEGER Mind
c       PARAMETER(Mind=200)
cc       ecart-type des erreurs                                            **
c       DOUBLE PRECISION sigma
c       PARAMETER(sigma=1)
cc       moyenne dans mon modele AR                                        **
c       DOUBLE PRECISION mu
c       PARAMETER(mu=0)
cc       parametre de la khi-deux                                          **
c       INTEGER df1
c       PARAMETER(df1=2)
cc       parametre de la student                                           **
c       INTEGER df2
c       PARAMETER(df2=5)
cc       parametre de la skew-normale                                      **
c       DOUBLE PRECISION lambda
c       PARAMETER(lambda=2.0)
c       DOUBLE PRECISION loi5b, loi5k, loi6p, loi6q
c       DOUBLE PRECISION loi7g, loi7d, loi8p, loi8q, loi9a, loi9b
c       DOUBLE PRECISION loi10b, loi10l, loi11a, loi11k, loi13g, loi13d
c       DOUBLE PRECISION loi14l, loi16p, loi16d, loi17p, loi17m
c       DOUBLE PRECISION loi18g, loi18d, loi19a, loi19b
c       PARAMETER(loi18g=1.0,loi18d=1.0, loi19a=1.1, loi19b=0.5)
c       PARAMETER(loi16p=0.2,loi16d=5.0, loi17p=0.2,loi17m=3.0)
c       PARAMETER(loi10b=0.2,loi10l=1, loi11a=2.0,loi11k=0.5)
c       PARAMETER(loi13g=1.0,loi13d=1.0, loi14l=0.7)
c       PARAMETER(loi7g=0,loi7d=1, loi8p=2, loi8q=2, loi9a=0,loi9b=2)
c       PARAMETER(loi5b=1, loi5k=1.8, loi6p=4, loi6q=1)
cc       loi des erreurs                                                   **
cc       si loi=0 : Normale(0,sigma^2)
cc       si loi=1 : Khi2 centree (df1)
cc       si loi=2 : Student (df2)
cc       si loi=3 : Skew-Normale(lambda)
cc       si loi=4 : Laplace
cc       si loi=5 : Weibull(b,k)
cc       si loi=6 : Gamma(p,q)
cc       si loi=7 : Log-Normale(g,d)
cc       si loi=8 : Beta(p,q)
cc       si loi=9 : Uniform(a,b)
cc       si loi=10 : Shifted exp (l,b)
cc       si loi=11 : Pareto(a,k)
cc       si loi=12 : Shifted Pareto
cc       si loi=13 : SU(g,d)
cc       si loi=14 : TU(l)
cc       si loi=15 : Logistic
cc       si loi=16 : SC(p,d)
cc       si loi=17 : LC(p,m)
cc       si loi=18 : SB(g,d)
cc       si loi=19 : S(a,b)
     **
```

```
c       INTEGER loi
c       PARAMETER(loi=0)
cc       the number of autoregressive coefficients supplied                    **
c       INTEGER p
c       PARAMETER(p=2)
cc       the number of moving-average coefficients supplied: q=0 puisque modele AR(p)
c       INTEGER q
c       PARAMETER(q=0)
cc       rm=max(p,q)
c       INTEGER rm
c       PARAMETER(rm=p)
cc       nombre d'observations dans mon echantillon                            **
cc       Si on change la valeur de nT, il faut aller modifier
cc       la valeur dans FORMAT a la fin du programme
c       INTEGER nT
c       PARAMETER(nT=100)
cc       the autoregressive coefficients of the model
c       DOUBLE PRECISION phi(2)
c       DOUBLE PRECISION phich(p)
cc       vecteur des donnees cree
c       DOUBLE PRECISION donees(nT)
cc       Parametres pour G05EGF: simulation
cc       the autoregressive coefficients of the model=phi     Input
c       DOUBLE PRECISION A(p)
cc       the moving-average coefficients of the model=teta     Input
c       DOUBLE PRECISION B(q+1)
cc       le vecteur des innovations     Output
c       DOUBLE PRECISION R(nT)
cc       Parametres pour G13DCF: estimation
cc       the number of initial parameter estimates   Input                     **
cc       mettre p+q+1 si MEAN=.TRUE.
c       INTEGER NPAR
c       PARAMETER(NPAR=p+q)
cc       first dimension of the array CM       Input                        **
cc       mettre p+q+1 si MEAN=.TRUE.
c       INTEGER ICM
c       PARAMETER(ICM=p+q)
cc       Workspace
c       INTEGER IW(NPAR+rm+3)
cc       initial parameter estimates     Input/Output
c       DOUBLE PRECISION PAR(NPAR)
cc       W(i,t) must be set equal to the observation at time t     Input
cc       of the ith series
c       DOUBLE PRECISION W(1,nT)
cc       the accuracy to which the solution in PAR and QQ is required     Input
     **
c       DOUBLE PRECISION CGETOL
c       PARAMETER(CGETOL=0.0001)
cc       residual at time t for series i, for i = 1,2,...,k     Output
c       DOUBLE PRECISION V(1,nT)
cc       estimated first derivative of the log-likelihood function     Output
c       DOUBLE PRECISION G(NPAR)
cc       estimate of the correlation coefficient between the ith and     Output
cc       jth elements in the PAR array
c       DOUBLE PRECISION CM(ICM,NPAR)
cc       Workspace
c       DOUBLE PRECISION WORK((5+3*nT+2*rm)+(2*p+q+2*rm*(rm+2)+9)+
c      +                    (NPAR+1)*(5*(NPAR+1)+29)/2+(rm+1)**2)
cc       MEAN must be set to .TRUE. if components of mu are to     Input          **
cc       be estimated and .FALSE. if all elements of mu are to be taken as zero
c       LOGICAL MEAN
c       PARAMETER(MEAN=.FALSE.)
cc       PARHLD(i) must be set to .TRUE., if PAR(i) is to be       Input
cc       held constant at its input value and .FALSE., if PAR(i) is a
cc       free parameter, for i = 1,2,...,NPAR.
```

```
c       LOGICAL PARHLD(NPAR)
cc      EXACT must be set equal to .TRUE. if the user wishes    Input          **
cc       the routine to compute exact maximum likelihood estimates.
cc      EXACT must be set equal to .FALSE. if only conditional
cc       likelihood estimates are required.
c       LOGICAL EXACT
cc       Contient les p donnees initiales de la serie
c       DOUBLE PRECISION Wtip(p)
cc       marret+p innovations genere par shock.f
c       DOUBLE PRECISION shocks(marret+p)
cc       defini dans Burn, calcule par ARpsi.f
c       DOUBLE PRECISION psi(marret+1)
cc       utile dans simAR.f
c       DOUBLE PRECISION phi2(marret)
cc       utile dans simAR.f
c       DOUBLE PRECISION YtpMn(p+Mind+nT)
cc       utile dans simAR.f
c       DOUBLE PRECISION Atn(nT+Mind),EspSU,EspSB,EspS
cc       the mean of the time series     Input
c       DOUBLE PRECISION E
c       PARAMETER(E=mu)
cc       the number of autoregressive coefficients supplied     Input
c       INTEGER NA
c       PARAMETER(NA=p)
cc       the number of moving-average coefficients supplied     Input
c       INTEGER NB
c       PARAMETER(NB=q+1)
cc       the dimension of the array R: vecteur des innovations     Input
c       INTEGER NR
c       PARAMETER(NR=nT)
cc       the number of observed time series, k  (chez moi k=1)  Input
c       INTEGER K
c       PARAMETER(K=1)
cc       the number of observations in each time series, n (chez moi=nT)   Input
c       INTEGER N2
c       PARAMETER(N2=nT)
cc       the number of AR parameter matrices, p    Input
c       INTEGER IP
c       PARAMETER(IP=p)
cc       the number of MA parameter matrices, q   Input
c       INTEGER IQ
c       PARAMETER(IQ=q)
cc       the first dimension of the arrays QQ, W and V    Input
c       INTEGER IK
c       PARAMETER(IK=1)
cc       the maximum number of likelihood evaluations to be    Input
cc       permitted by the search procedure
c       INTEGER MAXCAL
c       INTEGER MAXCA2
c       PARAMETER(MAXCAL=40*NPAR*(NPAR+5))
cc       which quantities are to be printed    Input
c       INTEGER ISHOW
c       PARAMETER(ISHOW=0)
cc       dimension of the array WORK    Input
c       INTEGER LWORK
c       PARAMETER(LWORK=(5+3*(nT)+2*rm)+(2*p+q+2*rm*(rm+2)+9)+
c      +                    (NPAR+1)*(5*(NPAR+1)+29)/2+(rm+1)**2)
cc       dimension of the array IW    Input
c       INTEGER LIW
c       PARAMETER(LIW=NPAR+rm+3)
cc       QQ(i,j) must be set equal to an initial estimate of    Input/Output
cc       the (i,j)th element of the covariance matrix of the residual series
c       DOUBLE PRECISION QQ(IK,K)
c-----------------------------------------------------------------------
c Fin des declarations des variables
```

```
c-----------------------------------------------------------------------
cc       Nom du fichier en sortie
     **
c        dataf='data.txt'
cc       Nom du fichier de parametres en sortie
     **
c       paramf='para.txt'
cc       Il faut rentrer les valeurs des coefficients phi(i)
     **
c       phi(1)=0.1
c       phi(2)=0.5
c       CALL cdatAR(dataf,paramf,nbcle,marret,Mind,sigma,mu,df1,df2,
c      +   lambda, loi5b, loi5k, loi6p, loi6q,
c      +  loi7g, loi7d, loi8p, loi8q, loi9a, loi9b,
c      +  loi10b, loi10l, loi11a, loi11k, loi13g, loi13d,
c      +  loi14l, loi16p, loi16d, loi17p, loi17m, loi18g, loi18d,
c      +  loi19a, loi19b, loi,p,q,rm,nT,phi,phich,donees,A,B,R,NPAR,
c      + ICM,IW,PAR,W,CGETOL,V,G,CM,WORK,MEAN,PARHLD,EXACT,Wtip,
c      + shocks,psi,phi2,YtpMn,Atn,EspSU,EspSB,EspS,
c      +E,NA,NB,NR,K,N2,IP,IQ,IK,MAXCAL,ISHOW,LWORK,LIW)
c       END
c       INCLUDE 'mean.f'
c       INCLUDE 'simulAR.f'
c       INCLUDE 'ARpsi.f'
c       INCLUDE 'shock.f'
c       INCLUDE 'rskew.f'
c       INCLUDE 'rlap.f'
c       INCLUDE 'rpare.f'
c       INCLUDE 'rspare.f'
c       INCLUDE 'rSU.f'
c       INCLUDE 'rTU.f'
c       INCLUDE 'rSC.f'
c       INCLUDE 'rLC.f'
c       INCLUDE 'rSB.f'
c       INCLUDE 'rS.f'
c       INCLUDE 'creerdat_AR.f'
c-----------------------------------------------------------------------
c Instructions de compilation: alias f77='fort77'
c f77 -c nom_du_fichier_contenant_la_fonction_main.f
c f77 -c creerdat_AR.f -lnag
c f77 nom_du_fichier_contenant_la_fonction_main.o creerdat_AR.o -lnag
c Fonctions exterieures appellees:
c G05EGF, G05EWF et  G13DCF de la librairie NAG MARK16 et simAR.f
c     Auteur: Pierre Lafaye de Micheaux
c     Date: 15/02/2001
c     Fin-Commentaires
      SUBROUTINE cdatAR(dataf,paramf,nbcle,marret,Mind,sigma,mu,df1,df2,
     +lambda,loi5b,loi5k,loi6p,loi6q,loi7g,loi7d,loi8p,loi8q,loi9a,
     +loi9b,loi10b,loi10l,loi11a,loi11k,loi13g,loi13d,loi14l,loi16p,
     +loi16d,loi17p,loi17m,loi18g,loi18d,loi19a,loi19b,loi,p,q,rm,nT,
     +phi,phich,donees,A,B,R,NPAR,ICM,IW,PAR,W,CGETOL,V,G,CM,WORK,MEAN,
     +PARHLD,EXACT,Wtip,shocks,psi,phi2,YtpMn,Atn,EspSU,EspSB,EspS,
     +E,NA,NB,NR,K,N2,IP,IQ,IK,MAXCAL,ISHOW,LWORK,LIW,QQ)
c-----------------------------------------------------------------------
c Debut des Declarations des variables
c-----------------------------------------------------------------------
      DOUBLE PRECISION ybarre,meanp
c     Nom du fichier en sortie
      CHARACTER dataf*8
c     Nom du fichier de parametres en sortie
      CHARACTER paramf*17
      INTEGER i,j,l
c     Compteur des nombres d'erreurs
      INTEGER err1
      INTEGER err2
```

```
      INTEGER err3
      INTEGER err4
      INTEGER err5
      INTEGER err6
      INTEGER err7
      INTEGER err8
c     Precision machine
      DOUBLE PRECISION preci, X02AJF
      EXTERNAL X02AJF
c     Temps de calcul
      DOUBLE PRECISION CPTIME, S1, S2, X05BAF
      EXTERNAL X05BAF
c     Nombre d'echantillons souhaites
      INTEGER nbcle
c     rang d'arret dans la random shock method de Burn
      INTEGER marret
c     Induction period dans la methode de Burn
      INTEGER Mind
c     ecart-type des erreurs
      DOUBLE PRECISION sigma
      DOUBLE PRECISION sigch2
c     moyenne dans mon modele AR
      DOUBLE PRECISION mu
      DOUBLE PRECISION much
c     parametre de la khi-deux
      INTEGER df1
c     parametre de la student
      INTEGER df2
c     parametre de la skew-normale
      DOUBLE PRECISION lambda
      DOUBLE PRECISION loi5b, loi5k, loi6p, loi6q
      DOUBLE PRECISION loi7g, loi7d, loi8p, loi8q, loi9a, loi9b
      DOUBLE PRECISION loi10b, loi10l, loi11a, loi11k, loi13g, loi13d
      DOUBLE PRECISION loi14l, loi16p, loi16d, loi17p, loi17m
      DOUBLE PRECISION loi18g, loi18d, loi19a, loi19b
c     loi des erreurs
c     si loi=0 : Normale(0,sigma^2)
c     si loi=1 : Khi2 centree (df1)
c     si loi=2 : Student (df2)
c     si loi=3 : Skew-Normale(lambda)
c     si loi=4 : Laplace
c     si loi=5 : Weibull(b,k)
c     si loi=6 : Gamma(p,q)
c     si loi=7 : Log-Normale(g,d)
c     si loi=8 : Beta(p,q)
c     si loi=9 : Uniform(a,b)
c     si loi=10 : Shifted exp (l,b)
c     si loi=11 : Pareto(a,k)
c     si loi=12 : Shifted Pareto
c     si loi=13 : SU(g,d)
c     si loi=14 : TU(l)
c     si loi=15 : Logistic
c     si loi=16 : SC(p,d)
c     si loi=17 : LC(p,m)
c     si loi=18 : SB(g,d)
c     si loi=19 : S(a,b)
      INTEGER loi
c     the number of autoregressive coefficients supplied
      INTEGER p
c     the number of moving-average coefficients supplied: q=0 puisque modele AR(p)
      INTEGER q
c     rm=max(p,q)
      INTEGER rm
c     nombre d'observations dans mon echantillon
c     Si on change la valeur de nT, il faut aller modifier
```

```
c     la valeur dans FORMAT a la fin du programme
      INTEGER nT
c     the autoregressive coefficients of the model
      DOUBLE PRECISION phi(2)
      DOUBLE PRECISION phich(p)
c     vecteur des donnees cree
      DOUBLE PRECISION donees(nT)
c Parametres pour G05EGF: simulation
      EXTERNAL G05EGF, G05EWF
      DOUBLE PRECISION G05EWF
c     the mean of the time series      Input
      DOUBLE PRECISION E
c     the number of autoregressive coefficients supplied      Input
      INTEGER NA
c     the autoregressive coefficients of the model=phi      Input
      DOUBLE PRECISION A(NA)
c     the number of moving-average coefficients supplied      Input
      INTEGER NB
c     the moving-average coefficients of the model=teta      Input
      DOUBLE PRECISION B(NB)
c     the dimension of the array R: vecteur des innovations      Input
      INTEGER NR
c     le vecteur des innovations      Output
      DOUBLE PRECISION R(NR)
c     the proportion of the variance of a term in the series      Output
c     that is due to the moving-average (error) terms in the model.
c     The smaller this is, the nearer is the model to
c     non-stationarity
      DOUBLE PRECISION VAR
c     Traitement des erreurs
      INTEGER IFAIL1
c Parametres pour G13DCF: estimation
      EXTERNAL G13DCF
c     the number of observed time series, k  (chez moi k=1)  Input
      INTEGER K
c     the number of observations in each time series, n (chez moi=nT)   Input
      INTEGER N2
c     the number of AR parameter matrices, p     Input
      INTEGER IP
c     the number of MA parameter matrices, q    Input
      INTEGER IQ
c     the number of initial parameter estimates    Input
c     mettre p+q+1 si MEAN=.TRUE.
      INTEGER NPAR
c     the first dimension of the arrays QQ, W and V     Input
      INTEGER IK
c     the frequency with which the automatic monitoring    Input
c     routine is to be called
      INTEGER IPRINT
c     the maximum number of likelihood evaluations to be     Input
c     permitted by the search procedure
      INTEGER MAXCAL
      INTEGER MAXCA2
c     which quantities are to be printed     Input
      INTEGER ISHOW
c     number of iterations performed by the search routine     Output
      INTEGER NITER
c     first dimension of the array CM       Input
c     mettre p+q+1 si MEAN=.TRUE.
      INTEGER ICM
c     dimension of the array WORK     Input
      INTEGER LWORK
c     dimension of the array IW     Input
      INTEGER LIW
c     Workspace
```


```
      INTEGER IW(LIW)
c     Traitement des erreurs   Input/Output
      INTEGER IFAIL2
c     initial parameter estimates    Input/Output
      DOUBLE PRECISION PAR(NPAR)
c     QQ(i,j) must be set equal to an initial estimate of    Input/Output
c     the (i,j)th element of the covariance matrix of the residual series
      DOUBLE PRECISION QQ(IK,K)
c     W(i,t) must be set equal to the observation at time t    Input
c     of the ith series
      DOUBLE PRECISION W(IK,N2)
c     the accuracy to which the solution in PAR and QQ is required    Input
      DOUBLE PRECISION CGETOL
c     value of the log-likelihood function corresponding to the final    Output
c     point held in PAR and QQ
      DOUBLE PRECISION RLOGL
c     residual at time t for series i, for i = 1,2,...,k    Output
      DOUBLE PRECISION V(IK,N2)
c     estimated first derivative of the log-likelihood function    Output
      DOUBLE PRECISION G(NPAR)
c     estimate of the correlation coefficient between the ith and    Output
c     jth elements in the PAR array
      DOUBLE PRECISION CM(ICM,NPAR)
c     Workspace
      DOUBLE PRECISION WORK(LWORK)
c     MEAN must be set to .TRUE. if components of mu are to    Input
c     be estimated and .FALSE. if all elements of mu are to be taken as zero
      LOGICAL MEAN
c     PARHLD(i) must be set to .TRUE., if PAR(i) is to be    Input
c     held constant at its input value and .FALSE., if PAR(i) is a
c     free parameter, for i = 1,2,...,NPAR.
      LOGICAL PARHLD(NPAR)
c     EXACT must be set equal to .TRUE. if the user wishes    Input
c     the routine to compute exact maximum likelihood estimates.
c     EXACT must be set equal to .FALSE. if only conditional
c     likelihood estimates are required.
      LOGICAL EXACT
c     Contient les p donnees initiales de la serie
      DOUBLE PRECISION Wtip(p)
c     marret+p innovations genere par shock.f
      DOUBLE PRECISION shocks(marret+p)
c     defini dans Burn, calcule par ARpsi.f
      DOUBLE PRECISION psi(marret+1)
c     utile dans simAR.f
      DOUBLE PRECISION phi2(marret)
c     utile dans simAR.f
      DOUBLE PRECISION YtpMn(p+Mind+nT)
c     utile dans simAR.f
      DOUBLE PRECISION Atn(nT+Mind),EspSU,EspSB,EspS
c-------------------------------------------------------------------
c Fin des Declarations des variables
c-------------------------------------------------------------------
      err1=0
      err2=0
      err3=0
      err4=0
      err5=0
      err6=0
      err7=0
      err8=0
c-------------------------------------------------------------------
c                              Debut du programme
c-------------------------------------------------------------------
      DO 10, j=1,p
         A(j)=phi(j)
```

```
 10       CONTINUE
      B(1)=sigma
      S1=X05BAF()
      OPEN(UNIT=12, FILE=dataf , STATUS='NEW' )
c-----------------------------------------------------------------------
c                              q=0 et p>0: modele AR(p)
c-----------------------------------------------------------------------

      preci=X02AJF()
      DO 70, i=1,nbcle
      WRITE(6,*) nbcle-i
      IPRINT=-1
      EXACT=.TRUE.
c-----------------------------------------------------------------------
c                                On simule les donnees:
c-----------------------------------------------------------------------

c donees=[Y1,...,YT]
      IF (loi .EQ. 0) THEN
         IFAIL1=0
c     ATTENTION!!! ICI JE N'AI PAS PU ENLEVER la moyenne aux epsilon comme dans
     simulARMA
c     C'est un probleme qu'il faudra resoudre quand mu sera different de 0
         CALL G05EGF(E, A, NA, B, NB, R, NR, VAR, IFAIL1)
c     A la sortie de G05EGF, R contient:
c     NA+0.5, NB+0.5, NA+NB+4+MAX(NA,NB)+0.5, mu, phi, sigma, teta, les NB valeurs
     passees de epsilon
         DO 20, j=1, nT
c     G05EWF va aussi modifier(mettre a jour) R en sortie
          donees(j)=G05EWF(R, NR, IFAIL1)
 20       CONTINUE
      ENDIF
      IF (loi .NE. 0) THEN
         CALL simAR(p,n,marret,Mind,loi,df1,df2,lambda,loi5b,
     +  loi5k, loi6p, loi6q, loi7g, loi7d, loi8p, loi8q, loi9a, loi9b,
     +  loi10b, loi10l, loi11a, loi11k, loi13g, loi13d,
     +  loi14l, loi16p, loi16d, loi17p, loi17m, loi18g, loi18d,
     +loi19a,loi19b,Wtip,shocks,psi,phi,phi2,YtpMn,Atn,EspSU,EspSB,EspS)
         DO 25, j=1,nT
            donees(j)=YtpMn(j+Mind+p)
 25         CONTINUE
      ENDIF
      ybarre=meanp(nT,donees)
c-----------------------------------------------------------------------
c                                   On estime les parametres:
c-----------------------------------------------------------------------

c     Je mets les nT donnees dans W
         DO 30, j=1, nT
            W(1,j)=donees(j)
 30         CONTINUE
      DO 40, j=1,(p+q)
         PAR(j)=phi(j)
         PARHLD(j)=.FALSE.
 40       CONTINUE
      IF (MEAN .EQV. .TRUE.) PAR(p+q+1)=mu
      IF (MEAN .EQV. .TRUE.) PARHLD(p+q+1)=.FALSE.
c     estimation de la variance des residus
      QQ(1,1)=sigma*sigma
      MAXCA2=MAXCAL
 45   IFAIL2=-1
c      WRITE(6,*) donees
```

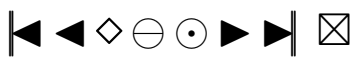
```
c       READ(5,*)
c      Calcul des estimateurs du maximum de vraisemblance exact et des residus
       CALL G13DCF(K, N2, IP, IQ, MEAN, PAR, NPAR, QQ, IK,
      +                    W, PARHLD, EXACT, IPRINT, CGETOL,
      +                    MAXCA2, ISHOW, NITER, RLOGL, V, G, CM,
      +                    ICM, WORK, LWORK, IW, LIW, IFAIL2)
       IF (IFAIL2 .EQ. 1) err1=err1+1
       IF (IFAIL2 .EQ. 2) err2=err2+1
       IF (IFAIL2 .EQ. 3) err3=err3+1
       IF (IFAIL2 .EQ. 4) err4=err4+1
       IF (IFAIL2 .EQ. 5) err5=err5+1
       IF (IFAIL2 .EQ. 6) err6=err6+1
       IF (IFAIL2 .EQ. 7) err7=err7+1
       IF (IFAIL2 .EQ. 8) err8=err8+1
c      Permet d'eliminer le cas d'erreur IFAIL=4
       IF (IFAIL2 .EQ. 4) MAXCA2=MAXCA2+100
       IF (IFAIL2 .EQ. 4) GOTO 45
       DO 50, j=1,p
          phich(j)=PAR(j)
 50       CONTINUE
       IF (MEAN .EQV. .TRUE.) much=PAR(p+1)
       sigch2=QQ(1,1)
c      Mon vecteur des phichapeau
c       WRITE(6,*) 'phichap=',phich
c      Ma variance estimee
c       WRITE(6,*) 'sigchap2=',sigch2
c      Mon vecteur de residus
c       WRITE(6,*) 'Residus:',V
c      Ma moyenne estimee
c       WRITE(6,*) 'Moyennechap=',much
c      On ecrit les donnees dans le fichier s'il n'y a pas eu d'erreur
c      c'est a dire qu'on ne garde que les bons echantillons
c       IF (IFAIL2 .EQ. 0) WRITE(12,60) sigch2, ((V(l,j),j=1,nT),l=1,1)
c      On ecrit toutes donnees dans le fichier sauf s'il y a eu des IFAIL1,2,3
       IF ((IFAIL2 .EQ. 0) .OR. (IFAIL2 .GT. 4)) WRITE(12,60) sigch2,
      + ((V(l,j),j=1,nT),l=1,1)
 60    FORMAT(SP,201D20.12)
c      FIN DE LA BOUCLE
 70    CONTINUE
       CLOSE(UNIT=12)
c      On compte le nombre d'echantillons reellement fabriques dans
c      le cas ou on elimine les mauvais echantillons
c       nbcle=nbcle-(err1+err2+err3+err5+err6+err7+err8)
c      On compte le nombre d'echantillons reellement fabriques dans
c      le cas ou on garde tous les echantillons (sauf IFAIL1,2,3)
       nbcle=nbcle-(err1+err2+err3)
       S2=X05BAF()
       CPTIME=S2-S1
       OPEN(UNIT=14, FILE=paramf , STATUS='NEW' )
       WRITE(14,*) 'Fichier '// paramf
       WRITE(14,*) 'p=',p
       WRITE(14,*) 'q=',q
       WRITE(14,*) 'Phi=',phi
       WRITE(14,*) 'nT=',nT
       WRITE(14,*) 'loi=',loi
       WRITE(14,*) 'nbcle=',nbcle
       WRITE(14,*) 'marret=', marret
       WRITE(14,*) 'Mind=', Mind
       WRITE(14,*) 'sigma=',sigma
       WRITE(14,*) 'mu=', mu
       WRITE(14,*) 'df1=',df1
       WRITE(14,*) 'df2=',df2
       WRITE(14,*) 'lambda=',lambda
       WRITE(14,*) 'loi5b=',loi5b
       WRITE(14,*) 'loi5k=',loi5k
```

```
      WRITE(14,*) 'loi6p=',loi6p
      WRITE(14,*) 'loi6q=',loi6q
      WRITE(14,*) 'loi7g=',loi7g
      WRITE(14,*) 'loi7d=',loi7d
      WRITE(14,*) 'loi8p=',loi8p
      WRITE(14,*) 'loi8q=',loi8q
      WRITE(14,*) 'loi9a=',loi9a
      WRITE(14,*) 'loi9b=',loi9b
      WRITE(14,*) 'loi10l=',loi10l
      WRITE(14,*) 'loi10b=',loi10b
      WRITE(14,*) 'loi11a=',loi11a
      WRITE(14,*) 'loi11k=',loi11k
      WRITE(14,*) 'loi13g=',loi13g
      WRITE(14,*) 'loi13d=',loi13d
      WRITE(14,*) 'loi14l=',loi14l
      WRITE(14,*) 'loi16p=',loi16p
      WRITE(14,*) 'loi16d=',loi16d
      WRITE(14,*) 'loi17p=',loi17p
      WRITE(14,*) 'loi17m=',loi17m
      WRITE(14,*) 'loi18g=',loi18g
      WRITE(14,*) 'loi18d=',loi18d
      WRITE(14,*) 'loi19a=',loi19a
      WRITE(14,*) 'loi19b=',loi19b
      WRITE(14,*) 'CGETOL=',CGETOL
      WRITE(14,*) 'LOGICAL MEAN=', MEAN
      WRITE(14,*) 'LOGICAL EXACT', EXACT
      WRITE(14,*) 'Precision machine=', preci
      WRITE(14,*) 'QUELQUES RESULTATS:'
      WRITE(14,*) 'Nombre d\'erreurs IFAIL2=1:',err1
      WRITE(14,*) 'Nombre d\'erreurs IFAIL2=2:',err2
      WRITE(14,*) 'Nombre d\'erreurs IFAIL2=3:',err3
      WRITE(14,*) 'Nombre d\'erreurs IFAIL2=4:',err4
      WRITE(14,*) 'Nombre d\'erreurs IFAIL2=5:',err5
      WRITE(14,*) 'Nombre d\'erreurs IFAIL2=6:',err6
      WRITE(14,*) 'Nombre d\'erreurs IFAIL2=7:',err7
      WRITE(14,*) 'Nombre d\'erreurs IFAIL2=8:',err8
      WRITE(14,*) 'Temps de calcul de creation des donnees:', CPTIME
      CLOSE(UNIT=14)
      END
c-----------------------------------------------------------------------
c                                Fin du programme
c-----------------------------------------------------------------------
```

Programmes creerdat_MA.f

```
c     Debut-Commentaires
c     Nom de la sous-routine: cdatMA
c Entrees:
c--------
c voir plus bas
c Sorties:
c--------
c Le fichier data.txt
c Description:
c-----------
c Modele MA(q)
c Ce programme cree le fichier de donnees data.txt qui contient, en ligne:
c sigch2 et epschap
c Utilisation dans une fonction main:
c-----------------------------------
c      PROGRAM main
c     Les ** indiquent les endroits ou des changements peuvent etre necessaires
c     Nom du fichier en sortie                                          **
c      CHARACTER dataf*8
c     Nom du fichier de parametres en sortie                             **
```

```
c       CHARACTER paramf*17
c      Nombre d'echantillons souhaites                                  **
c       INTEGER nbcle
c       PARAMETER(nbcle=10)
c      rang d'arret dans la random shock method de Burn                 **
c       INTEGER marret
c       PARAMETER(marret=200)
c      Induction period dans la methode de Burn                         **
c       INTEGER Mind
c       PARAMETER(Mind=200)
c      ecart-type des erreurs                                           **
c       DOUBLE PRECISION sigma
c       PARAMETER(sigma=1)
c      moyenne dans mon modele MA                                      **
c       DOUBLE PRECISION mu
c       PARAMETER(mu=0)
c      parametre de la khi-deux                                         **
c       INTEGER df1
c       PARAMETER(df1=2)
c      parametre de la student                                          **
c       INTEGER df2
c       PARAMETER(df2=5)
c      parametre de la skew-normale                                     **
c       DOUBLE PRECISION lambda
c       PARAMETER(lambda=2.0)
c        DOUBLE PRECISION loi5b, loi5k, loi6p, loi6q
c        DOUBLE PRECISION loi7g, loi7d, loi8p, loi8q, loi9a, loi9b
c        DOUBLE PRECISION loi10b, loi10l, loi11a, loi11k, loi13g, loi13d
c        DOUBLE PRECISION loi14l, loi16p, loi16d, loi17p, loi17m
c        DOUBLE PRECISION loi18g, loi18d, loi19a, loi19b
c        PARAMETER(loi18g=1.0,loi18d=1.0, loi19a=1.1, loi19b=0.5)
c        PARAMETER(loi16p=0.2,loi16d=5.0, loi17p=0.2,loi17m=3.0)
c        PARAMETER(loi10b=0.2,loi10l=1, loi11a=2.0,loi11k=0.5)
c        PARAMETER(loi13g=1.0,loi13d=1.0, loi14l=0.7)
c        PARAMETER(loi7g=0,loi7d=1, loi8p=2, loi8q=2, loi9a=0,loi9b=2)
c        PARAMETER(loi5b=1, loi5k=1.8, loi6p=4, loi6q=1)
c      loi des erreurs                                                  **
c      si loi=0 : Normale(0,sigma^2)
c      si loi=1 : Khi2 centree (df1)
c      si loi=2 : Student (df2)
c      si loi=3 : Skew-Normale(lambda)
c      si loi=4 : Laplace
c      si loi=5 : Weibull(b,k)
c      si loi=6 : Gamma(p,q)
c      si loi=7 : Log-Normale(g,d)
c      si loi=8 : Beta(p,q)
c      si loi=9 : Uniform(a,b)
c      si loi=10 : Shifted exp (l,b)
c      si loi=11 : Pareto(a,k)
c      si loi=12 : Shifted Pareto
c      si loi=13 : SU(g,d)
c      si loi=14 : TU(l)
c      si loi=15 : Logistic
c      si loi=16 : SC(p,d)
c      si loi=17 : LC(p,m)
c      si loi=18 : SB(g,d)
c      si loi=19 : S(a,b)
c       INTEGER loi
c       PARAMETER(loi=0)
c      the number of autoregressive coefficients supplied: p=0 puisque modele MA(q)
c       INTEGER p
c       PARAMETER(p=0)
c      the number of moving-average coefficients supplied
     **
c       INTEGER q
```

```
c       PARAMETER(q=2)
c      rm=max(p,q)
c       INTEGER rm
c       PARAMETER(rm=q)
c      nombre d'observations dans mon echantillon                          **
c      Si on change la valeur de nT, il faut aller modifier
c      la valeur dans FORMAT a la fin du programme
c       INTEGER nT
c       PARAMETER(nT=100)
c      the moving-average coefficients of the model
c       DOUBLE PRECISION teta(2)
c       DOUBLE PRECISION tetach(q)
c      vecteur des donnees cree
c       DOUBLE PRECISION donees(nT)
c Parametres pour G05EGF: simulation
c      the autoregressive coefficients of the model=phi      Input
c      Ici on met A(1) pour que la routine NAG G05EGF fonctionne
c       DOUBLE PRECISION A(1)
c      the moving-average coefficients of the model=teta     Input
c       DOUBLE PRECISION B(q+1)
c      le vecteur des innovations      Output
c       DOUBLE PRECISION R(nT)
c Parametres pour G13DCF: estimation
c      the number of initial parameter estimates   Input                   **
c      mettre p+q+1 si MEAN=.TRUE.
c       INTEGER NPAR
c       PARAMETER(NPAR=p+q)
c      first dimension of the array CM       Input                    **
c      mettre p+q+1 si MEAN=.TRUE.
c       INTEGER ICM
c       PARAMETER(ICM=p+q)
c      Workspace
c       INTEGER IW(NPAR+rm+3)
c      initial parameter estimates     Input/Output
c       DOUBLE PRECISION PAR(NPAR)
c     W(i,t) must be set equal to the observation at time t     Input
c      of the ith series
c       DOUBLE PRECISION W(1,nT)
c      the accuracy to which the solution in PAR and QQ is required     Input      **
c       DOUBLE PRECISION CGETOL
c       PARAMETER(CGETOL=0.0001)
c      residual at time t for series i, for i = 1,2,...,k      Output
c       DOUBLE PRECISION V(1,nT)
c      estimated first derivative of the log-likelihood function     Output
c       DOUBLE PRECISION G(NPAR)
c      estimate of the correlation coefficient between the ith and     Output
c      jth elements in the PAR array
c       DOUBLE PRECISION CM(ICM,NPAR)
c      Workspace
c       DOUBLE PRECISION WORK((5+3*nT+2*rm)+(2*p+q+2*rm*(rm+2)+9)+
c      +                    (NPAR+1)*(5*(NPAR+1)+29)/2+(rm+1)**2)
c     MEAN must be set to .TRUE. if components of mu are to     Input          **
c      be estimated and .FALSE. if all elements of mu are to be taken as zero
c       LOGICAL MEAN
c       PARAMETER(MEAN=.FALSE.)
c     PARHLD(i) must be set to .TRUE., if PAR(i) is to be       Input
c      held constant at its input value and .FALSE., if PAR(i) is a
c      free parameter, for i = 1,2,...,NPAR.
c       LOGICAL PARHLD(NPAR)
c     EXACT must be set equal to .TRUE. if the user wishes     Input          **
c      the routine to compute exact maximum likelihood estimates.
c     EXACT must be set equal to .FALSE. if only conditional
c      likelihood estimates are required.
c       LOGICAL EXACT
c      utile dans simARM.f
```


```
c       DOUBLE PRECISION YtMn(Mind+nT)
c      utile dans simARM.f
c       DOUBLE PRECISION Atnq(nT+Mind+q),EspSU,EspSB,EspS
cc        the mean of the time series      Input
c       DOUBLE PRECISION E
c       PARAMETER(E=mu)
cc        the number of autoregressive coefficients supplied      Input
c       INTEGER NA
c       PARAMETER(NA=p)
cc        the number of moving-average coefficients supplied      Input
c       INTEGER NB
c       PARAMETER(NB=q+1)
cc        the dimension of the array R: vecteur des innovations      Input
c       INTEGER NR
c       PARAMETER(NR=nT)
cc        the number of observed time series, k  (chez moi k=1)  Input
c       INTEGER K
c       PARAMETER(K=1)
cc        the number of observations in each time series, n (chez moi=nT)   Input
c       INTEGER N2
c       PARAMETER(N=nT)
cc        the number of AR parameter matrices, p     Input
c       INTEGER IP
c       PARAMETER(IP=p)
cc        the number of MA parameter matrices, q    Input
c       INTEGER IQ
c       PARAMETER(IQ=q)
cc        the first dimension of the arrays QQ, W and V     Input
c       INTEGER IK
c       PARAMETER(IK=1)
cc        the maximum number of likelihood evaluations to be     Input
cc        permitted by the search procedure
c       INTEGER MAXCAL
c       INTEGER MAXCA2
c       PARAMETER(MAXCAL=40*NPAR*(NPAR+5))
cc        which quantities are to be printed     Input
c       INTEGER ISHOW
c       PARAMETER(ISHOW=0)
cc        dimension of the array WORK     Input
c       INTEGER LWORK
c       PARAMETER(LWORK=(5+3*(nT)+2*rm)+(2*p+q+2*rm*(rm+2)+9)+
c      +                    (NPAR+1)*(5*(NPAR+1)+29)/2+(rm+1)**2)
cc        dimension of the array IW     Input
c       INTEGER LIW
c       PARAMETER(LIW=NPAR+rm+3)
cc        QQ(i,j) must be set equal to an initial estimate of     Input/Output
cc        the (i,j)th element of the covariance matrix of the residual series
c       DOUBLE PRECISION QQ(IK,K)
c-------------------------------------------------------------
c Fin des declarations des variables
c-------------------------------------------------------------
c     Nom du fichier de donnees en sortie
     **
c       dataf='data.txt'
c     Nom du fichier de parametres en sortie                                **
c       paramf='para.txt'
c     Il faut rentrer les valeurs des coefficients teta(i)                   **
c       teta(1)=0.1
c       teta(2)=0.2
c       CALL cdatMA(dataf,paramf,nbcle,marret,Mind,sigma,mu,df1,df2,
c      + lambda,loi5b, loi5k, loi6p, loi6q,
c       +  loi7g, loi7d, loi8p, loi8q, loi9a, loi9b,
c       +  loi10b, loi10l, loi11a, loi11k, loi13g, loi13d,
c       +  loi14l, loi16p, loi16d, loi17p, loi17m, loi18g, loi18d,
c       +  loi19a, loi19b, loi,p,q,rm,nT,teta,tetach,donees,A,B,R,NPAR,
```

```
c      + ICM,IW,PAR,W,CGETOL,V,G,CM,WORK,MEAN,PARHLD,EXACT,
c      + YtMn,Atnq,EspSU,EspSB,EspS,
c      +E,NA,NB,NR,K,N2,IP,IQ,IK,MAXCAL,ISHOW,LWORK,LIW,QQ)
c        END
c        INCLUDE 'mean.f'
c        INCLUDE 'simulMA.f'
c        INCLUDE 'rskew.f'
c        INCLUDE 'rlap.f'
c        INCLUDE 'rpare.f'
c        INCLUDE 'rspare.f'
c        INCLUDE 'rSU.f'
c        INCLUDE 'rTU.f'
c        INCLUDE 'rSC.f'
c        INCLUDE 'rLC.f'
c        INCLUDE 'rSB.f'
c        INCLUDE 'rS.f'
c        INCLUDE 'creerdat_MA.f'
c-----------------------------------------------------------------------
c     Instructions de compilation: alias f77='fort77'
c f77 -c nom_du_fichier_contenant_la_fonction_main.f
c f77 -c creerdat_MA.f -lnag
c f77 nom_du_fichier_contenant_la_fonction_main.o creerdat_MA.o -lnag
c Fonctions exterieures appellees:
c G05EGF, G05EWF et  G13DCF de la librairie NAG Mark16 et simMA.f
c     Auteur: Pierre Lafaye de Micheaux
c     Date: 15/02/2001
c     Fin-Commentaires
      SUBROUTINE cdatMA(dataf,paramf,nbcle,marret,Mind,sigma,mu,df1,df2,
     + lambda,loi5b,loi5k,loi6p,loi6q,loi7g,loi7d,loi8p,loi8q,loi9a,
     +loi9b,loi10b,loi10l,loi11a,loi11k,loi13g,loi13d,loi14l,loi16p,
     +loi16d,loi17p,loi17m,loi18g,loi18d,loi19a,loi19b,loi,p,q,rm,nT,
     + teta,tetach,donees,A,B,R,NPAR,
     + ICM,IW,PAR,W,CGETOL,V,G,CM,WORK,MEAN,PARHLD,EXACT,
     + YtMn,Atnq,EspSU,EspSB,EspS,
     +E,NA,NB,NR,K,N2,IP,IQ,IK,MAXCAL,ISHOW,LWORK,LIW,QQ)
c-----------------------------------------------------------------------
c Debut des Declarations des variables
c-----------------------------------------------------------------------
c-----------------------------------------------------------------------------------------------
c Debut des parametres a changer avant compilation et execution du programme
c-----------------------------------------------------------------------------------------------
      DOUBLE PRECISION ybarre,meanp
c     Nom du fichier en sortie
      CHARACTER dataf*8
c     Nom du fichier de parametres en sortie
      CHARACTER paramf*17
      INTEGER i,j,l
c     Compteur des nombres d'erreurs
      INTEGER err1
      INTEGER err2
      INTEGER err3
      INTEGER err4
      INTEGER err5
      INTEGER err6
      INTEGER err7
      INTEGER err8
c     Precision machine
      DOUBLE PRECISION preci, X02AJF
      EXTERNAL X02AJF
c     Temps de calcul
      DOUBLE PRECISION CPTIME, S1, S2, X05BAF
      EXTERNAL X05BAF
c     Nombre d'echantillons souhaites
      INTEGER nbcle
c     rang d'arret dans la random shock method de Burn
```

```
      INTEGER marret
c     Induction period dans la methode de Burn
      INTEGER Mind
c     ecart-type des erreurs
      DOUBLE PRECISION sigma
      DOUBLE PRECISION sigch2
c     moyenne dans mon modele MA
      DOUBLE PRECISION mu
      DOUBLE PRECISION much
c     parametre de la khi-deux
      INTEGER df1
c     parametre de la student
      INTEGER df2
c     parametre de la skew-normale
      DOUBLE PRECISION lambda
      DOUBLE PRECISION loi5b, loi5k, loi6p, loi6q
      DOUBLE PRECISION loi7g, loi7d, loi8p, loi8q, loi9a, loi9b
      DOUBLE PRECISION loi10b, loi10l, loi11a, loi11k, loi13g, loi13d
      DOUBLE PRECISION loi14l, loi16p, loi16d, loi17p, loi17m
      DOUBLE PRECISION loi18g, loi18d, loi19a, loi19b
c     loi des erreurs
c     si loi=0 : Normale(0,sigma^2)
c     si loi=1 : Khi2 centree (df1)
c     si loi=2 : Student (df2)
c     si loi=3 : Skew-Normale(lambda)
c     si loi=4 : Laplace
c     si loi=5 : Weibull(b,k)
c     si loi=6 : Gamma(p,q)
c     si loi=7 : Log-Normale(g,d)
c     si loi=8 : Beta(p,q)
c     si loi=9 : Uniform(a,b)
c     si loi=10 : Shifted exp (l,b)
c     si loi=11 : Pareto(a,k)
c     si loi=12 : Shifted Pareto
c     si loi=13 : SU(g,d)
c     si loi=14 : TU(l)
c     si loi=15 : Logistic
c     si loi=16 : SC(p,d)
c     si loi=17 : LC(p,m)
c     si loi=18 : SB(g,d)
c     si loi=19 : S(a,b)
      INTEGER loi
c     the number of autoregressive coefficients supplied: p=0 puisque modele MA(q)
      INTEGER p
c     the number of moving-average coefficients supplied
      INTEGER q
c     rm=max(p,q)
      INTEGER rm
c     nombre d'observations dans mon echantillon
c     Si on change la valeur de nT, il faut aller modifier
c     la valeur dans FORMAT a la fin du programme
      INTEGER nT
c     the moving-average coefficients of the model
      DOUBLE PRECISION teta(2)
      DOUBLE PRECISION tetach(q)
c     vecteur des donnees cree
      DOUBLE PRECISION donees(nT)
c Parametres pour G05EGF: simulation
      EXTERNAL G05EGF, G05EWF
      DOUBLE PRECISION G05EWF
c     the mean of the time series      Input
      DOUBLE PRECISION E
c     the number of autoregressive coefficients supplied      Input
      INTEGER NA
c     the autoregressive coefficients of the model=phi      Input
```

```
c     Ici on met A(1) pour que la routine NAG G05EGF fonctionne
      DOUBLE PRECISION A(1)
c     the number of moving-average coefficients supplied      Input
      INTEGER NB
c     the moving-average coefficients of the model=teta       Input
      DOUBLE PRECISION B(NB)
c     the dimension of the array R: vecteur des innovations   Input
      INTEGER NR
c     le vecteur des innovations      Output
      DOUBLE PRECISION R(NR)
c     the proportion of the variance of a term in the series      Output
c     that is due to the moving-average (error) terms in the model.
c     The smaller this is, the nearer is the model to
c     non-stationarity
      DOUBLE PRECISION VAR
c     Traitement des erreurs
      INTEGER IFAIL1
c Parametres pour G13DCF: estimation
      EXTERNAL G13DCF
c     the number of observed time series, k  (chez moi k=1)  Input
      INTEGER K
c     the number of observations in each time series, n (chez moi=nT)   Input
      INTEGER N2
c     the number of AR parameter matrices, p     Input
      INTEGER IP
c     the number of MA parameter matrices, q    Input
      INTEGER IQ
c     the number of initial parameter estimates    Input
c     mettre p+q+1 si MEAN=.TRUE.
      INTEGER NPAR
c     the first dimension of the arrays QQ, W and V     Input
      INTEGER IK
c     the frequency with which the automatic monitoring    Input
c     routine is to be called
      INTEGER IPRINT
c     the maximum number of likelihood evaluations to be     Input
c     permitted by the search procedure
      INTEGER MAXCAL
      INTEGER MAXCA2
c     which quantities are to be printed     Input
      INTEGER ISHOW
c     number of iterations performed by the search routine     Output
      INTEGER NITER
c     first dimension of the array CM        Input
c     mettre p+q+1 si MEAN=.TRUE.
      INTEGER ICM
c     dimension of the array WORK     Input
      INTEGER LWORK
c     dimension of the array IW     Input
      INTEGER LIW
c     Workspace
      INTEGER IW(LIW)
c     Traitement des erreurs    Input/Output
      INTEGER IFAIL2
c     initial parameter estimates     Input/Output
      DOUBLE PRECISION PAR(NPAR)
c     QQ(i,j) must be set equal to an initial estimate of     Input/Output
c     the (i,j)th element of the covariance matrix of the residual series
      DOUBLE PRECISION QQ(IK,K)
c     W(i,t) must be set equal to the observation at time t     Input
c     of the ith series
      DOUBLE PRECISION W(IK,N2)
c     the accuracy to which the solution in PAR and QQ is required     Input
      DOUBLE PRECISION CGETOL
c     value of the log-likelihood function corresponding to the final     Output
```

```
c      point held in PAR and QQ
       DOUBLE PRECISION RLOGL
c      residual at time t for series i, for i = 1,2,...,k      Output
       DOUBLE PRECISION V(IK,N2)
c      estimated first derivative of the log-likelihood function     Output
       DOUBLE PRECISION G(NPAR)
c      estimate of the correlation coefficient between the ith and    Output
c      jth elements in the PAR array
       DOUBLE PRECISION CM(ICM,NPAR)
c      Workspace
       DOUBLE PRECISION WORK(LWORK)
c      MEAN must be set to .TRUE. if components of mu are to     Input
c      be estimated and .FALSE. if all elements of mu are to be taken as zero
       LOGICAL MEAN
c      PARHLD(i) must be set to .TRUE., if PAR(i) is to be       Input
c      held constant at its input value and .FALSE., if PAR(i) is a
c      free parameter, for i = 1,2,...,NPAR.
       LOGICAL PARHLD(NPAR)
c      EXACT must be set equal to .TRUE. if the user wishes     Input
c      the routine to compute exact maximum likelihood estimates.
c      EXACT must be set equal to .FALSE. if only conditional
c      likelihood estimates are required.
       LOGICAL EXACT
c      utile dans simARM.f
       DOUBLE PRECISION YtMn(Mind+nT)
c      utile dans simARM.f
       DOUBLE PRECISION Atnq(nT+Mind+q),EspSU,EspSB,EspS
c-----------------------------------------------------------------
c Fin des Declarations des variables
c-----------------------------------------------------------------
       err1=0
       err2=0
       err3=0
       err4=0
       err5=0
       err6=0
       err7=0
       err8=0
c-----------------------------------------------------------------------
c                                  Debut du programme
c-----------------------------------------------------------------------
       B(1)=sigma
       DO 10, j=1,q
          B(j+1)=teta(j)*B(1)
 10       CONTINUE
       S1=X05BAF()
       OPEN(UNIT=12, FILE=dataf , STATUS='NEW' )
c-----------------------------------------------------------------------

c                                  q>0 et p=0: modele MA(q)
c-----------------------------------------------------------------------

       preci=X02AJF()
       DO 70, i=1,nbcle
       WRITE(6,*) nbcle-i
       IPRINT=-1
       EXACT=.TRUE.
c-----------------------------------------------------------------------

c                                  On simule les donnees:
c-----------------------------------------------------------------------

c donees=[Y1,...,YT]
       IF (loi .EQ. 0) THEN
          IFAIL1=0
```

```
c     ATTENTION!!! ICI JE N'AI PAS PU ENLEVER la moyenne aux epsilon comme dans
     simulARMA
c     C'est un probleme qu'il faudra resoudre quand mu sera different de 0
         CALL G05EGF(E, A, NA, B, NB, R, NR, VAR, IFAIL1)
c     A la sortie de G05EGF, R contient:
c     NA+0.5, NB+0.5, NA+NB+4+MAX(NA,NB)+0.5, mu, phi, sigma, teta, les NB valeurs
     passees de epsilon
         DO 20, j=1, nT
c     G05EWF va aussi modifier(mettre a jour) R en sortie
            donees(j)=G05EWF(R, NR, IFAIL1)
 20      CONTINUE
       ENDIF
       IF (loi .NE. 0) THEN
          CALL simMA (q,nT,marret,Mind,loi,df1,df2,lambda,loi5b,
      +  loi5k, loi6p, loi6q, loi7g, loi7d, loi8p, loi8q, loi9a, loi9b,
      +  loi10b, loi10l, loi11a, loi11k, loi13g, loi13d,
      +  loi14l, loi16p, loi16d, loi17p, loi17m, loi18g, loi18d,
      +  loi19a, loi19b,
      +        teta,YtMn,Atnq,EspSU,EspSB,EspS)
       DO 25, j=1,nT
          donees(j)=YtMn(j+Mind)
 25      CONTINUE
       ENDIF
       ybarre=meanp(nT,donees)
c-------------------------------------------------------------------------------

c                                    On estime les parametres:
c-------------------------------------------------------------------------------

c     Je mets les nT donnees dans W
         DO 30, j=1, nT
            W(1,j)=donees(j)
 30         CONTINUE
         DO 40, j=1,q
         PAR(j)=-teta(j)
         PARHLD(j)=.FALSE.
 40      CONTINUE
         IF (MEAN .EQV. .TRUE.) PAR(q+1)=mu
         IF (MEAN .EQV. .TRUE.) PARHLD(q+1)=.FALSE.
c     estimation de la variance des residus
       QQ(1,1)=sigma*sigma
       MAXCA2=MAXCAL
 45    IFAIL2=-1
c     Calcul des estimateurs du maximum de vraisemblance exact et des residus
       CALL G13DCF(K, N2, IP, IQ, MEAN, PAR, NPAR, QQ, IK,
      +                    W, PARHLD, EXACT, IPRINT, CGETOL,
      +                    MAXCA2, ISHOW, NITER, RLOGL, V, G, CM,
      +                    ICM, WORK, LWORK, IW, LIW, IFAIL2)
       IF (IFAIL2 .EQ. 1) err1=err1+1
       IF (IFAIL2 .EQ. 2) err2=err2+1
       IF (IFAIL2 .EQ. 3) err3=err3+1
       IF (IFAIL2 .EQ. 4) err4=err4+1
       IF (IFAIL2 .EQ. 5) err5=err5+1
       IF (IFAIL2 .EQ. 6) err6=err6+1
       IF (IFAIL2 .EQ. 7) err7=err7+1
       IF (IFAIL2 .EQ. 8) err8=err8+1
c     Permet d'eliminer le cas d'erreur IFAIL=4
       IF (IFAIL2 .EQ. 4) MAXCA2=MAXCA2+100
       IF (IFAIL2 .EQ. 4) GOTO 45
       DO 50, j=1,q
          tetach(j)=-PAR(j)
 50      CONTINUE
       IF (MEAN .EQV. .TRUE.) much=PAR(q+1)
       sigch2=QQ(1,1)
c     Mon vecteur des tetachapeau
```

```
c       WRITE(6,*) 'tetachap=',tetach
c      Ma variance estimee
c       WRITE(6,*) 'sigchap2=',sigch2
c      Mon vecteur de residus
c       WRITE(6,*) 'Residus:',V
c      Ma moyenne estimee
c       WRITE(6,*) 'Moyennechap=',much
c      On ecrit les donnees dans le fichier s'il n'y a pas eu d'erreur
c      c'est a dire qu'on ne garde que les bons echantillons
c       IF (IFAIL2 .EQ. 0) WRITE(12,60) sigch2, ((V(l,j),j=1,nT),l=1,1)
c      On ecrit toutes donnees dans le fichier sauf s'il y a eu des IFAIL1,2,3
       IF ((IFAIL2 .EQ. 0) .OR. (IFAIL2 .GT. 4)) WRITE(12,60) sigch2,
      + ((V(l,j),j=1,nT),l=1,1)
 60    FORMAT(SP,201D20.12)
c      FIN DE LA BOUCLE
 70    CONTINUE
       CLOSE(UNIT=12)
c      On compte le nombre d'echantillons reellement fabriques dans
c      le cas ou on elimine les mauvais echantillons
c       nbcle=nbcle-(err1+err2+err3+err5+err6+err7+err8)
c      On compte le nombre d'echantillons reellement fabriques dans
c      le cas ou on garde tous les echantillons (sauf IFAIL1,2,3)
       nbcle=nbcle-(err1+err2+err3)
       S2=X05BAF()
       CPTIME=S2-S1
       OPEN(UNIT=14, FILE=paramf , STATUS='NEW' )
       WRITE(14,*) 'Fichier '// paramf
       WRITE(14,*) 'p=',p
       WRITE(14,*) 'q=',q
       WRITE(14,*) 'Teta=',teta
       WRITE(14,*) 'nT=',nT
       WRITE(14,*) 'loi=',loi
       WRITE(14,*) 'nbcle=',nbcle
       WRITE(14,*) 'marret=', marret
       WRITE(14,*) 'Mind=', Mind
       WRITE(14,*) 'sigma=',sigma
       WRITE(14,*) 'mu=', mu
       WRITE(14,*) 'df1=',df1
       WRITE(14,*) 'df2=',df2
       WRITE(14,*) 'lambda=',lambda
       WRITE(14,*) 'loi5b=',loi5b
       WRITE(14,*) 'loi5k=',loi5k
       WRITE(14,*) 'loi6p=',loi6p
       WRITE(14,*) 'loi6q=',loi6q
       WRITE(14,*) 'loi7g=',loi7g
       WRITE(14,*) 'loi7d=',loi7d
       WRITE(14,*) 'loi8p=',loi8p
       WRITE(14,*) 'loi8q=',loi8q
       WRITE(14,*) 'loi9a=',loi9a
       WRITE(14,*) 'loi9b=',loi9b
       WRITE(14,*) 'loi10l=',loi10l
       WRITE(14,*) 'loi10b=',loi10b
       WRITE(14,*) 'loi11a=',loi11a
       WRITE(14,*) 'loi11k=',loi11k
       WRITE(14,*) 'loi13g=',loi13g
       WRITE(14,*) 'loi13d=',loi13d
       WRITE(14,*) 'loi14l=',loi14l
       WRITE(14,*) 'loi16p=',loi16p
       WRITE(14,*) 'loi16d=',loi16d
       WRITE(14,*) 'loi17p=',loi17p
       WRITE(14,*) 'loi17m=',loi17m
       WRITE(14,*) 'loi18g=',loi18g
       WRITE(14,*) 'loi18d=',loi18d
       WRITE(14,*) 'loi19a=',loi19a
       WRITE(14,*) 'loi19b=',loi19b
```

```
      WRITE(14,*) 'CGETOL=',CGETOL
      WRITE(14,*) 'LOGICAL MEAN=', MEAN
      WRITE(14,*) 'LOGICAL EXACT', EXACT
      WRITE(14,*) 'Precision machine=', preci
      WRITE(14,*) 'QUELQUES RESULTATS:'
      WRITE(14,*) 'Nombre d\'erreurs IFAIL2=1:',err1
      WRITE(14,*) 'Nombre d\'erreurs IFAIL2=2:',err2
      WRITE(14,*) 'Nombre d\'erreurs IFAIL2=3:',err3
      WRITE(14,*) 'Nombre d\'erreurs IFAIL2=4:',err4
      WRITE(14,*) 'Nombre d\'erreurs IFAIL2=5:',err5
      WRITE(14,*) 'Nombre d\'erreurs IFAIL2=6:',err6
      WRITE(14,*) 'Nombre d\'erreurs IFAIL2=7:',err7
      WRITE(14,*) 'Nombre d\'erreurs IFAIL2=8:',err8
      WRITE(14,*) 'Temps de calcul de creation des donnees:', CPTIME
      CLOSE(UNIT=14)
      END
c-----------------------------------------------------------------------
c                                Fin du programme
c-----------------------------------------------------------------------
```

Programmes creerdat_ARMA.f

```
c     Debut-Commentaires
c     Nom de la sous-routine: cdARMA
c Entrees:
c--------
c voir plus bas
c Sorties:
c--------
c Le fichier data.txt
c Description:
c-----------
c Modele ARMA(p,q)
c Ce programme cree le fichier de donnees data.txt qui contient, en ligne:
c sigch2 et epschap
c Utilisation dans une fonction main:
c-------------------------------------
c       PROGRAM main
cc       Les ** indiquent les endroits ou des changements peuvent etre necessaires
cc       Nom du fichier de donnees en sortie                                  **
c       CHARACTER dataf*8
cc       Nom du fichier de parametres en sortie                               **
c       CHARACTER paramf*17
cc       Nombre d'echantillons souhaites                                     **
c       INTEGER nbcle
c       PARAMETER(nbcle=10)
cc       rang d'arret dans la random shock method de Burn                    **
c       INTEGER marret
c       PARAMETER(marret=200)
cc       Induction period dans la methode de Burn                            **
c       INTEGER Mind
c       PARAMETER(Mind=200)
cc       ecart-type des erreurs                                              **
c       DOUBLE PRECISION sigma
c       PARAMETER(sigma=1)
cc       moyenne dans mon modele ARMA                                        **
c       DOUBLE PRECISION mu
c       PARAMETER(mu=0)
cc       parametre de la khi-deux                                            **
c       INTEGER df1
c       PARAMETER(df1=2)
cc       parametre de la student                                             **
c       INTEGER df2
c       PARAMETER(df2=5)
cc       parametre de la skew-normale                                        **
```

```
c       DOUBLE PRECISION lambda
c       PARAMETER(lambda=2.0)
c       DOUBLE PRECISION loi5b, loi5k, loi6p, loi6q
c       DOUBLE PRECISION loi7g, loi7d, loi8p, loi8q, loi9a, loi9b
c       DOUBLE PRECISION loi10b, loi10l, loi11a, loi11k, loi13g, loi13d
c       DOUBLE PRECISION loi14l, loi16p, loi16d, loi17p, loi17m
c       DOUBLE PRECISION loi18g, loi18d, loi19a, loi19b
c       PARAMETER(loi18g=1.0,loi18d=1.0, loi19a=1.1, loi19b=0.5)
c       PARAMETER(loi16p=0.2,loi16d=5.0, loi17p=0.2,loi17m=3.0)
c       PARAMETER(loi10b=0.2,loi10l=1, loi11a=2.0,loi11k=0.5)
c       PARAMETER(loi13g=1.0,loi13d=1.0, loi14l=0.7)
c       PARAMETER(loi7g=0,loi7d=1, loi8p=2, loi8q=2, loi9a=0,loi9b=2)
c       PARAMETER(loi5b=1, loi5k=1.8, loi6p=4, loi6q=1)
cc       loi des erreurs                                                **
cc       si loi=0 : Normale(0,sigma^2)
cc       si loi=1 : Khi2 centree (df1)
cc       si loi=2 : Student (df2)
cc       si loi=3 : Skew-Normale(lambda)
cc       si loi=4 : Laplace
cc       si loi=5 : Weibull(b,k)
cc       si loi=6 : Gamma(p,q)
cc       si loi=7 : Log-Normale(g,d)
cc       si loi=8 : Beta(p,q)
cc       si loi=9 : Uniform(a,b)
cc       si loi=10 : Shifted exp (l,b)
cc       si loi=11 : Pareto(a,k)
cc       si loi=12 : Shifted Pareto
cc       si loi=13 : SU(g,d)
cc       si loi=14 : TU(l)
cc       si loi=15 : Logistic
cc       si loi=16 : SC(p,d)
cc       si loi=17 : LC(p,m)
cc       si loi=18 : SB(g,d)
cc       si loi=19 : S(a,b)
c       INTEGER loi
c       PARAMETER(loi=2)
cc       the number of autoregressive coefficients supplied             **
c       INTEGER p
c       PARAMETER(p=2)
cc       the number of moving-average coefficients supplied             **
c       INTEGER q
c       PARAMETER(q=2)
cc       rm=max(p,q)                                                    **
c       INTEGER rm
c       PARAMETER(rm=2)
cc       nombre d'observations dans mon echantillon                     **
cc       Si on change la valeur de nT, il faut aller modifier
cc       la valeur dans FORMAT a la fin du programme creerdat_ARMA.f
c       INTEGER nT
c       PARAMETER(nT=100)
cc       the autoregressive coefficients of the model
c       DOUBLE PRECISION phi(2)
c       DOUBLE PRECISION phich(p)
cc       the moving-average coefficients of the model
c       DOUBLE PRECISION teta(2)
c       DOUBLE PRECISION tetach(q)
cc       vecteur des donnees cree
c       DOUBLE PRECISION donees(nT)
cc       Parametres pour G05EGF: simulation
cc       the autoregressive coefficients of the model=phi     Input
c       DOUBLE PRECISION A(p)
cc       the moving-average coefficients of the model=teta     Input
c       DOUBLE PRECISION B(q+1)
cc       le vecteur des innovations     Output
c       DOUBLE PRECISION R(nT)
```


```
cc       Parametres pour G13DCF: estimation
cc       the number of initial parameter estimates   Input                    **
cc       mettre p+q+1 si MEAN=.TRUE. , p+q si MEAN=.FALSE.
c        INTEGER NPAR
c        PARAMETER(NPAR=p+q)
cc       first dimension of the array CM       Input                          **
cc       mettre p+q+1 si MEAN=.TRUE. , p+q si MEAN=.FALSE.
c        INTEGER ICM
c        PARAMETER(ICM=p+q)
cc       Workspace
c        INTEGER IW(NPAR+rm+3)
cc       initial parameter estimates    Input/Output
c        DOUBLE PRECISION PAR(NPAR)
cc       W(i,t) must be set equal to the observation at time t    Input
cc       of the ith series
c        DOUBLE PRECISION W(1,nT)
cc       the accuracy to which the solution in PAR and QQ is required    Input
     **
c        DOUBLE PRECISION CGETOL
c        PARAMETER(CGETOL=0.0001)
cc       residual at time t for series i, for i = 1,2,...,k     Output
c        DOUBLE PRECISION V(1,nT)
cc       estimated first derivative of the log-likelihood function    Output
c        DOUBLE PRECISION G(NPAR)
cc       estimate of the correlation coefficient between the ith and    Output
cc       jth elements in the PAR array
c        DOUBLE PRECISION CM(ICM,NPAR)
cc       Workspace
c        DOUBLE PRECISION WORK((5+3*nT+2*rm)+(2*p+q+2*rm*(rm+2)+9)+
c       +                    (NPAR+1)*(5*(NPAR+1)+29)/2+(rm+1)**2)
cc       MEAN must be set to .TRUE. if components of mu are to    Input
     **
cc       be estimated and .FALSE. if all elements of mu are to be taken as zero
c        LOGICAL MEAN
c        PARAMETER(MEAN=.FALSE.)
cc       PARHLD(i) must be set to .TRUE., if PAR(i) is to be       Input
cc       held constant at its input value and .FALSE., if PAR(i) is a
cc       free parameter, for i = 1,2,...,NPAR.
c        LOGICAL PARHLD(NPAR)
cc       EXACT must be set equal to .TRUE. if the user wishes    Input
     **
cc       the routine to compute exact maximum likelihood estimates.
cc       EXACT must be set equal to .FALSE. if only conditional
cc       likelihood estimates are required.
c        LOGICAL EXACT
cc       Contient les p donnees initiales de la serie
c        DOUBLE PRECISION Wtip(p)
cc       marret+p innovations genere par shock.f
c        DOUBLE PRECISION shocks(marret+p)
cc       defini dans Burn, calcule par ARMpsi.f
c        DOUBLE PRECISION psi(marret+1)
cc       utile dans simARM.f de longueur marret                              **
c        DOUBLE PRECISION phi2(marret)
cc       utile dans simARM.f de longueur marret                             **
c        DOUBLE PRECISION teta2(marret)
cc       utile dans simARM.f
c        DOUBLE PRECISION YtpMn(p+Mind+nT)
cc       utile dans simARM.f
c        DOUBLE PRECISION Atnq(nT+Mind+q),EspSU,EspSB,EspS
cc       the mean of the time series      Input
c        DOUBLE PRECISION E
c        PARAMETER(E=mu)
cc       the number of autoregressive coefficients supplied     Input
c        INTEGER NA
c        PARAMETER(NA=p)
```

```
cc        the number of moving-average coefficients supplied      Input
c        INTEGER NB
c        PARAMETER(NB=q+1)
cc        the dimension of the array R: vecteur des innovations      Input
c        INTEGER NR
c        PARAMETER(NR=nT)
cc        the number of observed time series, k  (chez moi k=1)  Input
c        INTEGER K
c        PARAMETER(K=1)
cc        the number of observations in each time series, n (chez moi=nT)   Input
c        INTEGER N2
c        PARAMETER(N=nT)
cc        the number of AR parameter matrices, p     Input
c        INTEGER IP
c        PARAMETER(IP=p)
cc        the number of MA parameter matrices, q    Input
c        INTEGER IQ
c        PARAMETER(IQ=q)
cc        the first dimension of the arrays QQ, W and V     Input
c        INTEGER IK
c        PARAMETER(IK=1)
cc        the maximum number of likelihood evaluations to be     Input
cc        permitted by the search procedure
c        INTEGER MAXCAL
c        INTEGER MAXCA2
c        PARAMETER(MAXCAL=40*NPAR*(NPAR+5))
cc        which quantities are to be printed     Input
c        INTEGER ISHOW
c        PARAMETER(ISHOW=0)
cc        dimension of the array WORK     Input
c        INTEGER LWORK
c        PARAMETER(LWORK=(5+3*(nT)+2*rm)+(2*p+q+2*rm*(rm+2)+9)+
c       +                    (NPAR+1)*(5*(NPAR+1)+29)/2+(rm+1)**2)
cc        dimension of the array IW     Input
c        INTEGER LIW
c        PARAMETER(LIW=NPAR+rm+3)
cc        QQ(i,j) must be set equal to an initial estimate of     Input/Output
cc        the (i,j)th element of the covariance matrix of the residual series
c        DOUBLE PRECISION QQ(IK,K)
cc-----------------------------------------------------------------
cc Fin des declarations des variables
cc-----------------------------------------------------------------
cc        Nom du fichier de donnees en sortie
     **
c        dataf='data.txt'
cc        Nom du fichier de parametres en sortie
     **
c        paramf='para.txt'
cc        Il faut rentrer les valeurs des coefficients phi(i)
     **
c        phi(1)=0.1
c        phi(2)=0.5
cc        Il faut rentrer les valeurs des coefficients teta(i)
     **
c        teta(1)=0.1
c        teta(2)=0.2
c        CALL cdARMA(dataf,paramf,nbcle,marret,Mind,sigma,mu,df1,df2,
c       + lambda, loi5b, loi5k, loi6p, loi6q,
c       +  loi7g, loi7d, loi8p, loi8q, loi9a, loi9b,
c       +  loi10b, loi10l, loi11a, loi11k, loi13g, loi13d,
c       +  loi14l, loi16p, loi16d, loi17p, loi17m, loi18g, loi18d,
c       +  loi19a, loi19b,loi,p,q,rm,nT,phi,phich,teta,tetach,donees,A,B,
c       + R,NPAR,ICM,IW,PAR,W,CGETOL,V,G,CM,WORK,MEAN,PARHLD,EXACT,Wtip,
c       + shocks,psi,phi2,teta2,YtpMn,Atnq,EspSU,EspSB,EspS,
c       +E,NA,NB,NR,K,N2,IP,IQ,IK,MAXCAL,ISHOW,LWORK,LIW)
```

```
c       END
c       INCLUDE 'mean.f'
c       INCLUDE 'simulARMA.f'
c       INCLUDE 'ARMpsi.f'
c       INCLUDE 'shock.f'
c       INCLUDE 'rskew.f'
c       INCLUDE 'rlap.f'
c       INCLUDE 'rpare.f'
c       INCLUDE 'rspare.f'
c       INCLUDE 'rSU.f'
c       INCLUDE 'rTU.f'
c       INCLUDE 'rSC.f'
c       INCLUDE 'rLC.f'
c       INCLUDE 'rSB.f'
c       INCLUDE 'rS.f'
c       INCLUDE 'creerdat_ARMA.f'
c-------------------------------------------------
c Instructions de compilation: alias f77='fort77'
c f77 -c nom_du_fichier_contenant_la_fonction_main.f
c f77 -c creerdat_ARMA.f -lnag
c f77 nom_du_fichier_contenant_la_fonction_main.o creerdat_ARMA.o -lnag
c Fonctions exterieures appelees:
c G05EGF, G05EWF et  G13DCF de la librairie NAG Mark16 et simARMA.f
c Auteur: Pierre Lafaye de Micheaux
c Date: 15/02/2001
c Fin-Commentaires
      SUBROUTINE cdARMA(dataf,paramf,nbcle,marret,Mind,sigma,mu,df1,df2,
     + lambda,loi5b,loi5k,loi6p,loi6q,loi7g,loi7d,loi8p,loi8q,loi9a,
     +loi9b,loi10b,loi10l,loi11a,loi11k,loi13g,loi13d,loi14l,loi16p,
     +loi16d,loi17p,loi17m,loi18g,loi18d,loi19a,loi19b,loi,p,q,rm,nT,
     + phi,phich,teta,tetach,donees,A,
     + B,R,NPAR,ICM,IW,PAR,W,CGETOL,V,G,CM,WORK,MEAN,PARHLD,EXACT,
     + Wtip,shocks,psi,phi2,teta2,YtpMn,Atnq,EspSU,EspSB,EspS,
     +E,NA,NB,NR,K,N2,IP,IQ,IK,MAXCAL,ISHOW,LWORK,LIW,QQ)
c-------------------------------------------------------
c Debut des Declarations des variables
c-------------------------------------------------------
      DOUBLE PRECISION ybarre,meanp
c     Nom du fichier de donnees en sortie
      CHARACTER dataf*8
c     Nom du fichier de parametres en sortie
      CHARACTER paramf*17
      INTEGER i,j,l
c     Compteurs des nombres d'erreurs
      INTEGER err1
      INTEGER err2
      INTEGER err3
      INTEGER err4
      INTEGER err5
      INTEGER err6
      INTEGER err7
      INTEGER err8
c     Precision machine
      DOUBLE PRECISION preci, X02AJF
      EXTERNAL X02AJF
c     Temps de calcul
      DOUBLE PRECISION CPTIME, S1, S2, X05BAF
      EXTERNAL X05BAF
c     Nombre d'echantillons souhaites
      INTEGER nbcle
c     rang d'arret dans la random shock method de Burn
      INTEGER marret
c     Induction period dans la methode de Burn
      INTEGER Mind
c     ecart-type des erreurs
```

```
      DOUBLE PRECISION sigma
      DOUBLE PRECISION sigch2
c     moyenne dans mon modele ARMA
      DOUBLE PRECISION mu
      DOUBLE PRECISION much
c     parametre de la khi-deux
      INTEGER df1
c     parametre de la student
      INTEGER df2
c     parametre de la skew-normale
      DOUBLE PRECISION lambda
      DOUBLE PRECISION loi5b, loi5k, loi6p, loi6q
      DOUBLE PRECISION loi7g, loi7d, loi8p, loi8q, loi9a, loi9b
      DOUBLE PRECISION loi10b, loi10l, loi11a, loi11k, loi13g, loi13d
      DOUBLE PRECISION loi14l, loi16p, loi16d, loi17p, loi17m
      DOUBLE PRECISION loi18g, loi18d, loi19a, loi19b
c     loi des erreurs
c     si loi=0 : Normale(0,sigma^2)
c     si loi=1 : Khi2 centree (df1)
c     si loi=2 : Student (df2)
c     si loi=3 : Skew-Normale(lambda)
c     si loi=4 : Laplace
c     si loi=5 : Weibull(b,k)
c     si loi=6 : Gamma(p,q)
c     si loi=7 : Log-Normale(g,d)
c     si loi=8 : Beta(p,q)
c     si loi=9 : Uniform(a,b)
c     si loi=10 : Shifted exp (l,b)
c     si loi=11 : Pareto(a,k)
c     si loi=12 : Shifted Pareto
c     si loi=13 : SU(g,d)
c     si loi=14 : TU(l)
c     si loi=15 : Logistic
c     si loi=16 : SC(p,d)
c     si loi=17 : LC(p,m)
c     si loi=18 : SB(g,d)
c     si loi=19 : S(a,b)
      INTEGER loi
c     the number of autoregressive coefficients supplied
      INTEGER p
c     the number of moving-average coefficients supplied
      INTEGER q
c     rm=max(p,q)                                                    **
      INTEGER rm
c     nombre d'observations dans mon echantillon
c     Si on change la valeur de nT, il faut aller modifier
c     la valeur dans FORMAT a la fin du programme
      INTEGER nT
c     the autoregressive coefficients of the model
      DOUBLE PRECISION phi(2)
      DOUBLE PRECISION phich(p)
c     the moving-average coefficients of the model
      DOUBLE PRECISION teta(2)
      DOUBLE PRECISION tetach(q)
c     vecteur des donnees cree
      DOUBLE PRECISION donees(nT)
c Parametres pour G05EGF: simulation
      EXTERNAL G05EGF, G05EWF
      DOUBLE PRECISION G05EWF
c     the mean of the time series      Input
      DOUBLE PRECISION E
c     the number of autoregressive coefficients supplied     Input
      INTEGER NA
c     the autoregressive coefficients of the model=phi     Input
      DOUBLE PRECISION A(NA)
```


```
c     the number of moving-average coefficients supplied      Input
      INTEGER NB
c     the moving-average coefficients of the model=teta       Input
      DOUBLE PRECISION B(NB)
c     the dimension of the array R: vecteur des innovations   Input
      INTEGER NR
c     le vecteur des innovations      Output
      DOUBLE PRECISION R(NR)
c     the proportion of the variance of a term in the series      Output
c     that is due to the moving-average (error) terms in the model.
c     The smaller this is, the nearer is the model to
c     non-stationarity
      DOUBLE PRECISION VAR
c     Traitement des erreurs
      INTEGER IFAIL1
c Parametres pour G13DCF: estimation
      EXTERNAL G13DCF
c     the number of observed time series, k  (chez moi k=1)  Input
      INTEGER K
c     the number of observations in each time series, n (chez moi=nT)   Input
      INTEGER N2
c     the number of AR parameter matrices, p     Input
      INTEGER IP
c     the number of MA parameter matrices, q    Input
      INTEGER IQ
c     the number of initial parameter estimates    Input
c     mettre p+q+1 si MEAN=.TRUE. , p+q si MEAN=.FALSE.
      INTEGER NPAR
c     the first dimension of the arrays QQ, W and V     Input
      INTEGER IK
c     the frequency with which the automatic monitoring    Input
c     routine is to be called
      INTEGER IPRINT
c     the maximum number of likelihood evaluations to be     Input
c     permitted by the search procedure
      INTEGER MAXCAL
      INTEGER MAXCA2
c     which quantities are to be printed     Input
      INTEGER ISHOW
c     number of iterations performed by the search routine     Output
      INTEGER NITER
c     first dimension of the array CM       Input
c     mettre p+q+1 si MEAN=.TRUE. , p+q si MEAN=.FALSE.
      INTEGER ICM
c     dimension of the array WORK     Input
      INTEGER LWORK
c     dimension of the array IW     Input
      INTEGER LIW
c     Workspace
      INTEGER IW(LIW)
c     Traitement des erreurs    Input/Output
      INTEGER IFAIL2
c     initial parameter estimates     Input/Output
      DOUBLE PRECISION PAR(NPAR)
c     QQ(i,j) must be set equal to an initial estimate of     Input/Output
c     the (i,j)th element of the covariance matrix of the residual series
      DOUBLE PRECISION QQ(IK,K)
c     W(i,t) must be set equal to the observation at time t     Input
c     of the ith series
      DOUBLE PRECISION W(IK,N2)
c     the accuracy to which the solution in PAR and QQ is required     Input
      DOUBLE PRECISION CGETOL
c     value of the log-likelihood function corresponding to the final     Output
c     point held in PAR and QQ
      DOUBLE PRECISION RLOGL
```

```
c     residual at time t for series i, for i = 1,2,...,k      Output
      DOUBLE PRECISION V(IK,N2)
c     estimated first derivative of the log-likelihood function     Output
      DOUBLE PRECISION G(NPAR)
c     estimate of the correlation coefficient between the ith and    Output
c     jth elements in the PAR array
      DOUBLE PRECISION CM(ICM,NPAR)
c     Workspace
      DOUBLE PRECISION WORK(LWORK)
c     MEAN must be set to .TRUE. if components of mu are to     Input
c     be estimated and .FALSE. if all elements of mu are to be taken as zero
      LOGICAL MEAN
c     PARHLD(i) must be set to .TRUE., if PAR(i) is to be       Input
c     held constant at its input value and .FALSE., if PAR(i) is a
c     free parameter, for i = 1,2,...,NPAR.
      LOGICAL PARHLD(NPAR)
c     EXACT must be set equal to .TRUE. if the user wishes    Input
c     the routine to compute exact maximum likelihood estimates.
c     EXACT must be set equal to .FALSE. if only conditional
c     likelihood estimates are required.
      LOGICAL EXACT
c     Contient les p donnees initiales de la serie
      DOUBLE PRECISION Wtip(p)
c     marret+p innovations genere par shock.f
      DOUBLE PRECISION shocks(marret+p)
c     defini dans Burn, calcule par ARMpsi.f
      DOUBLE PRECISION psi(marret+1)
c     utile dans simARM.f de longueur marret
      DOUBLE PRECISION phi2(marret)
c     utile dans simARM.f de longueur marret
      DOUBLE PRECISION teta2(marret)
c     utile dans simARM.f
      DOUBLE PRECISION YtpMn(p+Mind+nT)
c     utile dans simARM.f
      DOUBLE PRECISION Atnq(nT+Mind+q),EspSU,EspSB,EspS
c-----------------------------------------------------------------------
c Fin des Declarations des variables
c-----------------------------------------------------------------------
      err1=0
      err2=0
      err3=0
      err4=0
      err5=0
      err6=0
      err7=0
      err8=0
c-----------------------------------------------------------------------
c                                Debut du programme
c-----------------------------------------------------------------------
      DO 10, j=1,p
         A(j)=phi(j)
 10      CONTINUE
      B(1)=sigma
      DO 20, j=1,q
         B(j+1)=teta(j)*B(1)
 20      CONTINUE
      S1=X05BAF()
      OPEN(UNIT=12, FILE=dataf , STATUS='NEW' )
c-----------------------------------------------------------------------

c                            q>0 et p>0: modele ARMA(p,q)
c-----------------------------------------------------------------------

      preci=X02AJF()
      DO 100, i=1,nbcle
```

```
      WRITE(6,*) nbcle-i
      IPRINT=-1
      EXACT=.TRUE.
c-----------------------------------------------------------------------

c                                On simule les donnees:
c-----------------------------------------------------------------------

c donees=[Y1,...,YT]
      IF (loi .EQ. 0) THEN
      IFAIL1=0
c     ATTENTION!!! ICI JE N'AI PAS PU ENLEVER la moyenne aux epsilon comme dans
     simulARMA
c     C'est un probleme qu'il faudra resoudre quand mu sera different de 0
         CALL G05EGF(E, A, NA, B, NB, R, NR, VAR, IFAIL1)
c     A la sortie de G05EGF, R contient:
c     NA+0.5, NB+0.5, NA+NB+4+MAX(NA,NB)+0.5, mu, phi, sigma, teta, les NB valeurs
     passees de epsilon
         DO 30, j=1, nT
c     G05EWF va aussi modifier(mettre a jour) R en sortie
            donees(j)=G05EWF(R, NR, IFAIL1)
 30      CONTINUE
      ENDIF
      IF (loi .NE. 0) THEN
         CALL simARM (p,q,nT,marret,Mind,loi,df1,df2,lambda,loi5b,
     +  loi5k, loi6p, loi6q, loi7g, loi7d, loi8p, loi8q, loi9a, loi9b,
     +  loi10b, loi10l, loi11a, loi11k, loi13g, loi13d,
     +  loi14l, loi16p, loi16d, loi17p, loi17m, loi18g, loi18d,
     +  loi19a, loi19b,Wtip,shocks,psi,phi,
     +  teta,phi2,teta2,YtpMn,Atnq,EspSU,EspSB,EspS)
      DO 35, j=1,nT
         donees(j)=YtpMn(j+Mind+p)
 35      CONTINUE
      ENDIF
      ybarre=meanp(nT,donees)
c-----------------------------------------------------------------------

c                                   On estime les parametres:
c-----------------------------------------------------------------------

c     Je mets les nT donnees dans W
         DO 40, j=1, nT
            W(1,j)=donees(j)
 40         CONTINUE
         DO 50, j=1, p
            PAR(j)=phi(j)
            PARHLD(j)=.FALSE.
 50         CONTINUE
         DO 60, j=1 ,q
         PAR(j+p)=-teta(j)
         PARHLD(j+p)=.FALSE.
 60      CONTINUE
         IF (MEAN) PAR(p+q+1)=mu
         IF (MEAN) PARHLD(p+q+1)=.FALSE.
c     estimation de la variance des residus
      QQ(1,1)=sigma*sigma
      MAXCA2=MAXCAL
 65   IFAIL2=-1
c     Calcul des estimateurs du maximum de vraisemblance exact et des residus
      CALL G13DCF(K, N2, IP, IQ, MEAN, PAR, NPAR, QQ, IK,
     +                      W, PARHLD, EXACT, IPRINT, CGETOL,
     +                      MAXCA2, ISHOW, NITER, RLOGL, V, G, CM,
     +                      ICM, WORK, LWORK, IW, LIW, IFAIL2)
      IF (IFAIL2 .EQ. 1) err1=err1+1
      IF (IFAIL2 .EQ. 2) err2=err2+1
```

```
      IF (IFAIL2 .EQ. 3) err3=err3+1
      IF (IFAIL2 .EQ. 4) err4=err4+1
      IF (IFAIL2 .EQ. 5) err5=err5+1
      IF (IFAIL2 .EQ. 6) err6=err6+1
      IF (IFAIL2 .EQ. 7) err7=err7+1
      IF (IFAIL2 .EQ. 8) err8=err8+1
c     Permet d'eliminer le cas d'erreur IFAIL=4
      IF (IFAIL2 .EQ. 4) MAXCA2=MAXCA2+100
      IF (IFAIL2 .EQ. 4) GOTO 65
      DO 70, j=1,p
         phich(j)=PAR(j)
 70      CONTINUE
      DO 80, j=1,q
         tetach(j)=-PAR(j+p)
 80      CONTINUE
      IF (MEAN) much=PAR(p+q+1)
      sigch2=QQ(1,1)
c     Mon vecteur des phichapeau
c      WRITE(6,*) 'phichap=',phich
c     Mon vecteur des tetachapeau
c      WRITE(6,*) 'tetachap=',tetach
c     Ma variance estimee
c      WRITE(6,*) 'sigchap2=',sigch2
c     Mon vecteur de residus
c      WRITE(6,*) 'Residus:',V
c     Ma moyenne estimee
c      WRITE(6,*) 'Moyennechap=',much
c     On ecrit les donnees dans le fichier s'il n'y a pas eu d'erreur
c     c'est a dire qu'on ne garde que les bons echantillons
c      IF (IFAIL2 .EQ. 0) WRITE(12,60) sigch2, ((V(l,j),j=1,nT),l=1,1)
c     On ecrit toutes donnees dans le fichier sauf s'il y a eu des IFAIL1,2,3
      IF ((IFAIL2 .EQ. 0) .OR. (IFAIL2 .GT. 4)) WRITE(12,90) sigch2,
     + ((V(l,j),j=1,nT),l=1,1)
 90   FORMAT(SP,51D20.12)                                                **
c     FIN DE LA BOUCLE
 100  CONTINUE
      CLOSE(UNIT=12)
c     On compte le nombre d'echantillons reellement fabriques dans
c     le cas ou on elimine les mauvais echantillons
c      nbcle=nbcle-(err1+err2+err3+err5+err6+err7+err8)
c     On compte le nombre d'echantillons reellement fabriques dans
c     le cas ou on garde tous les echantillons (sauf IFAIL1,2,3)
      nbcle=nbcle-(err1+err2+err3)
      S2=X05BAF()
      CPTIME=S2-S1
      OPEN(UNIT=14, FILE=paramf , STATUS='NEW' )
      WRITE(14,*) 'Fichier '// paramf
      WRITE(14,*) 'p=',p
      WRITE(14,*) 'q=',q
      WRITE(14,*) 'Phi=',phi
      WRITE(14,*) 'Teta=',teta
      WRITE(14,*) 'nT=',nT
      WRITE(14,*) 'loi=',loi
      WRITE(14,*) 'nbcle=',nbcle
      WRITE(14,*) 'marret=', marret
      WRITE(14,*) 'Mind=', Mind
      WRITE(14,*) 'sigma=',sigma
      WRITE(14,*) 'mu=', mu
      WRITE(14,*) 'df1=',df1
      WRITE(14,*) 'df2=',df2
      WRITE(14,*) 'lambda=',lambda
      WRITE(14,*) 'loi5b=',loi5b
      WRITE(14,*) 'loi5k=',loi5k
      WRITE(14,*) 'loi6p=',loi6p
      WRITE(14,*) 'loi6q=',loi6q
```


```
      WRITE(14,*) 'loi7g=',loi7g
      WRITE(14,*) 'loi7d=',loi7d
      WRITE(14,*) 'loi8p=',loi8p
      WRITE(14,*) 'loi8q=',loi8q
      WRITE(14,*) 'loi9a=',loi9a
      WRITE(14,*) 'loi9b=',loi9b
      WRITE(14,*) 'loi10l=',loi10l
      WRITE(14,*) 'loi10b=',loi10b
      WRITE(14,*) 'loi11a=',loi11a
      WRITE(14,*) 'loi11k=',loi11k
      WRITE(14,*) 'loi13g=',loi13g
      WRITE(14,*) 'loi13d=',loi13d
      WRITE(14,*) 'loi14l=',loi14l
      WRITE(14,*) 'loi16p=',loi16p
      WRITE(14,*) 'loi16d=',loi16d
      WRITE(14,*) 'loi17p=',loi17p
      WRITE(14,*) 'loi17m=',loi17m
      WRITE(14,*) 'loi18g=',loi18g
      WRITE(14,*) 'loi18d=',loi18d
      WRITE(14,*) 'loi19a=',loi19a
      WRITE(14,*) 'loi19b=',loi19b
      WRITE(14,*) 'CGETOL=',CGETOL
      WRITE(14,*) 'LOGICAL MEAN=', MEAN
      WRITE(14,*) 'LOGICAL EXACT', EXACT
      WRITE(14,*) 'Precision machine=', preci
      WRITE(14,*) 'QUELQUES RESULTATS:'
      WRITE(14,*) 'Nombre d\'erreurs IFAIL2=1:',err1
      WRITE(14,*) 'Nombre d\'erreurs IFAIL2=2:',err2
      WRITE(14,*) 'Nombre d\'erreurs IFAIL2=3:',err3
      WRITE(14,*) 'Nombre d\'erreurs IFAIL2=4:',err4
      WRITE(14,*) 'Nombre d\'erreurs IFAIL2=5:',err5
      WRITE(14,*) 'Nombre d\'erreurs IFAIL2=6:',err6
      WRITE(14,*) 'Nombre d\'erreurs IFAIL2=7:',err7
      WRITE(14,*) 'Nombre d\'erreurs IFAIL2=8:',err8
      WRITE(14,*) 'Temps de calcul de creation des donnees:', CPTIME
      CLOSE(UNIT=14)
      END
c-----------------------------------------------------------------------
c                              Fin du programme
c-----------------------------------------------------------------------
```

A.7. LE PROGRAMME CALCSTAT.F

Programme calcstat.f

```
c     Debut-Commentaires
c     Nom de la sous-routine: calcstat
c Entrees:
c--------
c voir plus bas
c Sorties:
c--------
c Le fichier resultat.txt
c Description:
c-----------
c     Cette sous-routine calcule les statistiques du test lisse sur les ARMA,
c     d'Anderson Darling et de Brockwell et Davis a partir d'un fichier de donnees
     data.txt
c     contenant les donnees simulees d'un modele ARMA
c     Ces donnees sont fabriquees par l'une des sous-routines creerdat_AR.f,
     creerdat_MA.f
c     ou creerdat_ARMA.f, il s'agit de sigchap2 et epschap sur chaque ligne
c     Les resultats sont stockes dans le fichier resultat.txt
c Utilisation dans une fonction main:
c-------------------------------------
```

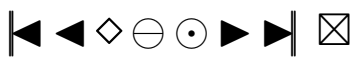
```
c       PROGRAM main
cc       Les ** indiquent les endroits ou des changements peuvent etre necessaires
c       DOUBLE PRECISION  khi(10), khi2(10)
cc       Si isa=.TRUE. on prend les polynomes modifies avec les ak  **
cc       Si isa=.FALSE. on prend les polynomes modifies sans les ak
c       LOGICAL isa
cc       Quantiles Ledwi1 et Ledwi2 pour alpha                          **
c       DOUBLE PRECISION QuLed1, QuLed2
c       PARAMETER(QuLed1=3.692,QuLed2=4.932)
cc       Quantiles Ledwi1 et Ledwi2 pour alpha2                          **
c       DOUBLE PRECISION QuLe12, QuLe22
c       PARAMETER(QuLe12=3.692,QuLe22=4.932)
cc       Quantiles Brockwell et Davis, et Anderson Darling, et JB pour alpha       **
c       DOUBLE PRECISION QuBD, QuAD, QuJB
c       PARAMETER(QuBD=0.9786,QuAD=1.743,QuJB=4.61)
cc       Quantiles Brockwell et Davis, et Anderson Darling, et JB pour alpha2       **
c       DOUBLE PRECISION QuBD2, QuAD2, QuJB2
c       PARAMETER(QuBD2=0.9786,QuAD2=1.743, QuJB2=5.99)
cc       Niveau du test                                          **
c       DOUBLE PRECISION alpha, alpha2
c       PARAMETER(alpha=0.1,alpha2=0.05)
cc       Nombre de polynomes                                     **
c       INTEGER Kp
c       PARAMETER(Kp=10)
cc       ordre du modele AR (1 ou 2)                             **
c       INTEGER p
c       PARAMETER(p=2)
cc       ordre du modele MA (1 ou 2)                             **
c       INTEGER q
c       PARAMETER(q=2)
cc       Taille des echantillons                                 **
c       INTEGER nT
c       PARAMETER(nT=100)
c       DOUBLE PRECISION phi(2), teta(2)
c       INTEGER n,m,nbcle
c       PARAMETER(m=nT+1)
cc       n=nbcle=nombre de lignes dans le fichier de donnees           **
cc       on peu faire wc-l data.txt pour le connaitre
c       PARAMETER(n=990,nbcle=990)
c       CHARACTER filer*8, filew*12                                    **
c       DOUBLE PRECISION valeur(n,m)
c       DOUBLE PRECISION sch2vc(n)
c       DOUBLE PRECISION epscha(n,nT)
c       INTEGER compt(Kp)
c       INTEGER compt2(Kp)
c       DOUBLE PRECISION statn(n,Kp)
c       INTEGER KoLed1(n)
c       INTEGER KoLed2(n)
c       DOUBLE PRECISION Ledwi1(Kp)
c       DOUBLE PRECISION Ledwi2(Kp-1)
c       DOUBLE PRECISION snLed1(n)
c       DOUBLE PRECISION snLed2(n)
c       DOUBLE PRECISION stanBD(n)
c       DOUBLE PRECISION stanAD(n)
c       DOUBLE PRECISION stanJB(n)
c       DOUBLE PRECISION Z(nT)
c       DOUBLE PRECISION D(nT)
c       DOUBLE PRECISION Davant(nT), E2(nT)
c       DOUBLE PRECISION U(nT)
c       DOUBLE PRECISION hUetoi(nT,Kp)
c       DOUBLE PRECISION Uavant(nT)
c       DOUBLE PRECISION hKetoi(Kp)
c       DOUBLE PRECISION hU(nT,Kp)
c       DOUBLE PRECISION hK(Kp)
c       DOUBLE PRECISION vecteu(Kp)
```


```
c       DOUBLE PRECISION vectoi(Kp)
c       DOUBLE PRECISION stat(Kp)
c       DOUBLE PRECISION Zavant(nT)
c       DOUBLE PRECISION BDa(nT)
c       DOUBLE PRECISION BDb(nT)
c       DOUBLE PRECISION BDc(nT)
c       DOUBLE PRECISION BDd(nT)
c       DOUBLE PRECISION Davan2(nT)
c       DOUBLE PRECISION ADa(nT)
c       DOUBLE PRECISION ADb(nT)
c       DOUBLE PRECISION Puiss(Kp)
c       DOUBLE PRECISION Puiss2(Kp)
c       DOUBLE PRECISION snsort(n,Kp)
c       DOUBLE PRECISION snsorv(n)
c       DOUBLE PRECISION Qucalc(Kp)
c       DOUBLE PRECISION snLe1s(n)
c       DOUBLE PRECISION snLe2s(n)
c       DOUBLE PRECISION snBDso(n)
c       DOUBLE PRECISION snADso(n)
c       DOUBLE PRECISION snJBso(n)
c       DOUBLE PRECISION matric(nbcle,18)
c       CHARACTER paramf*9
c       INTEGER loi
c       INTEGER dnT
c       PARAMETER(dnT=Kp)
c       DOUBLE PRECISION QLed1u,QLe12u
c       PARAMETER(QLed1u=QuLed1,QLe12u=QuLe12)
c       DOUBLE PRECISION QLed2u,QLe22u
c       PARAMETER(QLed2u=QuLed2,QLe22u=QuLe22)
c       DOUBLE PRECISION QBDu,QBD2u
c       PARAMETER(QBDu=QuBD,QBD2u=QuBD2)
c       DOUBLE PRECISION QADu,QAD2u
c       PARAMETER(QADu=QuAD,QAD2u=QuAD2)
c       DOUBLE PRECISION QJBu,QJB2u
c       PARAMETER(QJBu=QuJB,QJB2u=QuJB2)
cc        Quantiles de la loi du Khi2 (K=10;alpha=0.1)                    **
cc       a (de)commenter si necessaire
c       khi(1)=2.71
c       khi(2)=4.61
c       khi(3)=6.25
c       khi(4)=7.78
c       khi(5)=9.24
c       khi(6)=10.64
c       khi(7)=12.02
c       khi(8)=13.36
c       khi(9)=14.68
c       khi(10)=15.99
cc        Quantiles de la loi du Khi2 (K=10;alpha=0.05)                    **
cc       a (de)commenter si necessaire
c       khi2(1)=3.84
c       khi2(2)=5.99
c       khi2(3)=7.81
c       khi2(4)=9.49
c       khi2(5)=11.07
c       khi2(6)=12.59
c       khi2(7)=14.07
c       khi2(8)=15.51
c       khi2(9)=16.92
c       khi2(10)=18.31
cc        Quantiles utilises pour le calcul de la stat(vrais quantiles calcules sous H0
     ), pour alpha  **
cc       a (de)commenter si necessaire
c       khi(1)=2.394426
c       khi(2)=4.392086
c       khi(3)=5.954882
```

```
c       khi(4)=7.577112
c       khi(5)=9.020887
c       khi(6)=10.523610
c       khi(7)=11.927158
c       khi(8)=13.508102
c       khi(9)=14.852025
c       khi(10)=16.271442
cc       Quantiles utilises pour le calcul de la stat(vrais quantiles calcules sous H0
     ), pour alpha2  **
cc       a (de)commenter si necessaire
c       khi2(1)=2.394426
c       khi2(2)=4.392086
c       khi2(3)=5.954882
c       khi2(4)=7.577112
c       khi2(5)=9.020887
c       khi2(6)=10.523610
c       khi2(7)=11.927158
c       khi2(8)=13.508102
c       khi2(9)=14.852025
c       khi2(10)=16.271442
cc       Valeurs des parametres phi et teta                          **
c       phi(1)=0.1
c       phi(2)=0.5
c       teta(1)=0.1
c       teta(2)=0.2
cc       On peut aussi changer le nom du fichier en entree (data.txt) et     **
cc       le nom du fichier en sortie (resultat.txt)
c       filer='data.txt'
c       filew='resultat.txt'
c       paramf='param.txt'
c       loi=0
c       CALL calcstat(isa,khi,khi2,QuLed1,QuLe12,QuLed2,QuLe22,QuBD,
c      + QuBD2,QuAD,QuAD2,alpha,Kp,p,q,nT
c      +     ,phi,teta,n,nbcle,m,filer,filew,valeur,sch2vc,epscha,compt,
c      + compt2,statn,KoLed1,KoLed2,Ledwi1,Ledwi2,snLed1,snLed2,stanBD,
c      +stanAD,stanJB
c      +    Z,D,Davant,E2,U,hUetoi,Uavant,hKetoi,hU,hK,vecteu,vectoi,stat,
c      +    Zavant,BDa,BDb,BDc,BDd,Davan2,ADa,ADb,Puiss,Puiss2,
c      +snsort,snsorv,
c      +    Qucalc,snLe1s,snLe2s,snBDso,snADso,snJBso,matric,paramf,loi,
c      + dnT,QLed1u,QLe12u,QLed2u,QLe22u,QBDu,QBD2u,QADu,QAD2u,QJBu,QJB2u)
c       END
c       INCLUDE 'H1isa.f'
c       INCLUDE 'H2isa.f'
c       INCLUDE 'H3isa.f'
c       INCLUDE 'H4isa.f'
c       INCLUDE 'H5isa.f'
c       INCLUDE 'H6isa.f'
c       INCLUDE 'H7isa.f'
c       INCLUDE 'H8isa.f'
c       INCLUDE 'H9isa.f'
c       INCLUDE 'H10isa.f'
c       INCLUDE 'H1etoile.f'
c       INCLUDE 'H2etoile.f'
c       INCLUDE 'H3etoile.f'
c       INCLUDE 'H4etoile.f'
c       INCLUDE 'H5etoile.f'
c       INCLUDE 'H6etoile.f'
c       INCLUDE 'H7etoile.f'
c       INCLUDE 'H8etoile.f'
c       INCLUDE 'H9etoile.f'
c       INCLUDE 'H10etoile.f'
c       INCLUDE 'H1.f'
c       INCLUDE 'H2.f'
c       INCLUDE 'H3.f'
```


```
c        INCLUDE 'H4.f'
c        INCLUDE 'H5.f'
c        INCLUDE 'H6.f'
c        INCLUDE 'H7.f'
c        INCLUDE 'H8.f'
c        INCLUDE 'H9.f'
c        INCLUDE 'H10.f'
c        INCLUDE 'qnorm.f'
c        INCLUDE 'pnorm.f'
c        INCLUDE 'min.f'
c        INCLUDE 'max.f'
c        INCLUDE 'mean.f'
c        INCLUDE 'var.f'
c-------------------------------------
c     Instructions de compilation: alias f77='fort77'
c f77 -c nom_du_fichier_contenant_la_fonction_main.f
c f77 -c calcstat.f -lnag
c f77 nom_du_fichier_contenant_la_fonction_main.o calcsat.o -lnag
c     Fonctions exterieures appelees:
c     M01CAF NAG Mark16 routine
c     Auteur: Pierre Lafaye de Micheaux
c     Date: 15/02/2001
c     Fin-Commentaires
      SUBROUTINE calcstat(isa,khi,khi2,QuLed1,QuLe12,QuLed2,QuLe22,QuBD,
     + QuBD2,QuAD,QuAD2,QuJB,QuJB2,alpha,alpha2,Kp,p,q,nT,phi,teta,n,
     + nbcle,m,filer,filew,valeur,sch2vc,epscha,compt,compt2,statn,
     + KoLed1,KoLed2,Ledwi1,Ledwi2,snLed1,snLed2,stanBD,stanAD,stanJB,
     + Z,D,Davant,E2,U,hUetoi,Uavant,hKetoi,hU,hK,vecteu,vectoi,stat,
     + Zavant,BDa,BDb,BDc,BDd,Davan2,ADa,ADb,Puiss,Puiss2,snsort,snsorv,
     + Qucalc,snLe1s,snLe2s,snBDso,snADso,snJBso,matric,paramf,loi,
     + dnT,QLed1u,QLe12u,QLed2u,QLe22u,QBDu,QBD2u,QADu,QAD2u,QJBu,QJB2u)
c-------------------------------------------------------------
c Debut des Declarations des variables
c-------------------------------------------------------------
      CHARACTER paramf*17
      LOGICAL isa
      DOUBLE precision meanp,mini, maxi, var
      DOUBLE PRECISION  khi(10), khi2(10)
c     Quantiles Ledwi1 et Ledwi2
      DOUBLE PRECISION QuLed1, QuLed2,QuLe12,QuLe22
c     Quantiles Brockwell et Davis, et Anderson Darling, et Jarque et Bera
      DOUBLE PRECISION QuBD, QuAD,QuBD2,QuAD2,QuJB,QuJB2
c     Niveau du test
      DOUBLE PRECISION alpha,alpha2
c     Nombre de polynomes
      INTEGER Kp
c     ordre du modele AR (1 ou 2)
      INTEGER p
c     ordre du modele MA (1 ou 2)
      INTEGER q
c     Taille des echantillons
      INTEGER nT
      DOUBLE PRECISION phi(2), teta(2)
      INTEGER i,j,k
c     n=nombre de lignes dans le fichier de donnees
c     on peu faire wc -l data.txt pour le connaitre
      INTEGER n,m,nbcle
      DOUBLE PRECISION valeur(n,m)
      DOUBLE PRECISION sch2vc(n)
      DOUBLE PRECISION epscha(n,nT)
      INTEGER compt(Kp),compt2(Kp)
      INTEGER coLed1, coLed2,coLe12,coLe22
      INTEGER contBD, contAD,conBD2,conAD2, contJB, conJB2
      DOUBLE PRECISION statn(n,Kp)
      INTEGER KoLed1(n)
```

```
INTEGER KoLed2(n)
INTEGER dnT
DOUBLE PRECISION Ledwi1(dnT)
DOUBLE PRECISION Ledwi2(dnT-1)
DOUBLE PRECISION penal
DOUBLE PRECISION snLed1(n)
DOUBLE PRECISION snLed2(n)
DOUBLE PRECISION stanBD(n)
DOUBLE PRECISION stanAD(n), stanJB(n)
DOUBLE PRECISION Qtutil(10),Qtuti2(10)
DOUBLE PRECISION QLed1u,QLe12u
DOUBLE PRECISION QLed2u,QLe22u
DOUBLE PRECISION QBDu,QBD2u
DOUBLE PRECISION QADu,QAD2u
DOUBLE PRECISION QJBu,QJB2u
DOUBLE PRECISION sigch2
DOUBLE PRECISION Z(nT)
DOUBLE PRECISION D(nT)
DOUBLE PRECISION Davant(nT), E2(nT)
DOUBLE PRECISION mu, sigma
DOUBLE PRECISION U(nT)
DOUBLE PRECISION hUetoi(nT,Kp)
DOUBLE PRECISION Uavant(nT)
DOUBLE PRECISION hKetoi(Kp)
DOUBLE PRECISION hKtemp
DOUBLE PRECISION hU(nT,Kp)
DOUBLE PRECISION hK(Kp)
DOUBLE PRECISION vecteu(Kp)
DOUBLE PRECISION inter
DOUBLE PRECISION mxLed1
INTEGER ordre
INTEGER vKoLe1
DOUBLE PRECISION mxLed2
INTEGER ordre2
INTEGER vKoLe2
DOUBLE PRECISION vectoi(Kp)
DOUBLE PRECISION stat(Kp)
DOUBLE PRECISION stLed1
DOUBLE PRECISION stLed2
DOUBLE PRECISION Zavant(nT)
INTEGER nTul
DOUBLE PRECISION BDa(nT)
DOUBLE PRECISION BDb(nT)
DOUBLE PRECISION BDc(nT)
DOUBLE PRECISION BDd(nT)
DOUBLE PRECISION statBD
DOUBLE PRECISION sumbd, sumc, sumd2
DOUBLE PRECISION Davan2(nT)
DOUBLE PRECISION ADa(nT)
DOUBLE PRECISION ADb(nT)
DOUBLE PRECISION statAD
DOUBLE PRECISION sumAD
DOUBLE PRECISION Puiss(Kp),Puiss2(Kp)
DOUBLE PRECISION PsLed1, PsLed2, PuisAD, PuisBD, PuisJB
DOUBLE PRECISION PsLe12, PsLe22, PuiAD2, PuiBD2, PuiJB2
INTEGER val, val2
DOUBLE PRECISION snsort(n,Kp)
INTEGER nul
DOUBLE PRECISION snsorv(n)
DOUBLE PRECISION Qucalc(Kp),Qucal2(10)
DOUBLE PRECISION snLe1s(n)
DOUBLE PRECISION QLed1c, QLed12
DOUBLE PRECISION snLe2s(n)
DOUBLE PRECISION QLed2c, QLed22
DOUBLE PRECISION snBDso(n)
```

```
      DOUBLE PRECISION QBDcal, QBDca2
      DOUBLE PRECISION snADso(n), snJBso(n)
      DOUBLE PRECISION QADcal, QADca2, QJBcal, QJBca2
      DOUBLE PRECISION rgLed1(2)
      DOUBLE PRECISION rgLed2(2)
      DOUBLE PRECISION rgBD(2)
      DOUBLE PRECISION rgAD(2), rgJB(2)
      DOUBLE PRECISION mecLe1(2)
      DOUBLE PRECISION mecLe2(2)
      DOUBLE PRECISION mecBD(2)
      DOUBLE PRECISION mecAD(2), mecJB(2)
      DOUBLE PRECISION matric(nbcle,18)
      INTEGER frKLe1(10)
      INTEGER frKLe2(10)
      CHARACTER filer*8, filew*21, filtab*19, filexc*24
      DOUBLE PRECISION moy
      DOUBLE PRECISION m2, m3,m4, g1, g2, g1a, g1b
      DOUBLE PRECISION statJB
      INTEGER IFAIL
      filexc=filew//'.xc'
      filtab='./SIMUL/tableau.txt'
      penal=DLOG(DBLE(nT))
      coLed1=INT(0)
      coLed2=INT(0)
      contBD=INT(0)
      contAD=INT(0)
      coLe12=INT(0)
      coLe22=INT(0)
      conBD2=INT(0)
      conAD2=INT(0)
      IF (p .EQ. 0) THEN
         phi(1)=DBLE(0.0)
         phi(2)=DBLE(0.0)
         ENDIF
      IF (p .EQ. 1) THEN
         phi(2)=DBLE(0.0)
         ENDIF
      IF (q .EQ. 0) THEN
         teta(1)=DBLE(0.0)
         teta(2)=DBLE(0.0)
         ENDIF
      IF (q .EQ. 1) THEN
         teta(2)=DBLE(0.0)
         ENDIF
c-----------------------------------------------------------
c Fin des Declarations des variables
c-----------------------------------------------------------
c-----------------------------------------------------------------------
c                              Debut du programme
c-----------------------------------------------------------------------
c     On met les donnees du fichier dans la matrice valeurs
         OPEN(UNIT=10, FILE=filer , STATUS='OLD' )
         i=INT(1)
      DO  WHILE (2 .GT. 1)
         READ(10,7,END=10) (valeur(i,j),j=1,m)
         i=INT(i+1)
      END DO
 7    FORMAT(BN,51D20.12)                                          **
 10   WRITE(6,*) 'Votre fichier de donnees contient',i-1,' echantillons'
      IF ((i-1) .NE. nbcle) THEN
      WRITE(6,*) 'Rectifiez la valeur de n et nbcle a',i-1,'
     + au debut du prog.'
      STOP
      ENDIF
      CLOSE(10)
```

```
c     creation du vecteur des sigchap2
      DO 20, i=1,nbcle
         sch2vc(i)=valeur(i,1)
 20      CONTINUE
c     creation de la matrice des epschap
      DO 40, i=1,nbcle
         DO 30, j=1,nT
            epscha(i,j)=valeur(i,j+1)
 30         CONTINUE
 40      CONTINUE
c     On remplit Qtutil, vecteur des quantiles utilises
      DO 50,i=1,10
         Qtutil(i)=khi(i)
         Qtuti2(i)=khi2(i)
 50      CONTINUE
c     On commence la grande boucle-----------------------------------------
      DO 9000, j=1,nbcle
c     on affiche un compteur a l'ecran
      WRITE(6,*) nbcle-j
      sigch2=sch2vc(j)
      DO 60, i=1,nT
         Z(i)=epscha(j,i)
 60      CONTINUE
      DO 70, i=1,nT
         D(i)=Z(i)/DSQRT(sigch2)
 70      CONTINUE
      mu=DBLE(0.0)
      sigma=DBLE(1.0)
c     On sauvegarde les valeurs de D dans Davant
      DO 80, i=1,nT
         Davant(i)=D(i)
 80      CONTINUE
c     remplace D par pnorm(D)
      CALL pnorm(nT,D,mu,sigma)
c     affecte a E le resultat
      DO 90, i=1,nT
         E2(i)=D(i)
 90      CONTINUE
c     remet les bonnes valeurs dans D
      DO 100, i=1,nT
         D(i)=Davant(i)
 100      CONTINUE
c     U=[U1,...,UT]
      DO 110, i=1,nT
         U(i)=DBLE(2.0)*E2(i)-DBLE(1.0)
 110      CONTINUE
c     hKetoile'[,k] est le vecteur de longueur nT (hketoile(U1),...,hketoile(UT))
c     On remplit la matrice hUetoile
c     i=1...................
c     On sauvegarde les valeurs de U dans Uavant
      DO 120, i=1,nT
         Uavant(i)=U(i)
 120      CONTINUE
      IF (isa .EQV. .TRUE.) THEN
c     calcul de H1isa(U), remplace U par H1isa(U)
      CALL H1isa(nT,U)
      ENDIF
      IF (isa .EQV. .FALSE.) THEN
c     calcul de H1etoi(U), remplace U par H1etoi(U)
      CALL H1etoi(nT,U)
      ENDIF
c     affecte a hUetoile[,1] le resultat
      DO 130, i=1,nT
         hUetoi(i,1)=U(i)
 130      CONTINUE
```

```
c     remet les bonnes valeurs dans U
      DO 140, i=1,nT
          U(i)=Uavant(i)
 140      CONTINUE
c     i=2...................
c     On sauvegarde les valeurs de U dans Uavant
      DO 150, i=1,nT
         Uavant(i)=U(i)
 150      CONTINUE
      IF (isa .EQV. .TRUE.) THEN
c     calcul de H2isa(U), remplace U par H2isa(U)
      CALL H2isa(nT,U)
      ENDIF
      IF (isa .EQV. .FALSE.) THEN
c     calcul de H2etoi(U), remplace U par H2etoi(U)
      CALL H2etoi(nT,U)
      ENDIF
c     affecte a hUetoile[,2] le resultat
      DO 160, i=1,nT
         hUetoi(i,2)=U(i)
 160      CONTINUE
c     remet les bonnes valeurs dans U
      DO 170, i=1,nT
          U(i)=Uavant(i)
 170      CONTINUE
c     i=3...................
c     On sauvegarde les valeurs de U dans Uavant
      DO 180, i=1,nT
         Uavant(i)=U(i)
 180      CONTINUE
      IF (isa .EQV. .TRUE.) THEN
c     calcul de H3isa(U), remplace U par H3isa(U)
      CALL H3isa(nT,U)
      ENDIF
      IF (isa .EQV. .FALSE.) THEN
c     calcul de H3etoi(U), remplace U par H3etoi(U)
      CALL H3etoi(nT,U)
      ENDIF
c     affecte a hUetoile[,3] le resultat
      DO 190, i=1,nT
         hUetoi(i,3)=U(i)
 190      CONTINUE
c     remet les bonnes valeurs dans U
      DO 200, i=1,nT
          U(i)=Uavant(i)
 200   CONTINUE
c     i=4...................
c     On sauvegarde les valeurs de U dans Uavant
      DO 210, i=1,nT
         Uavant(i)=U(i)
 210      CONTINUE
      IF (isa .EQV. .TRUE.) THEN
c     calcul de H4isa(U), remplace U par H4isa(U)
      CALL H4isa(nT,U)
      ENDIF
      IF (isa .EQV. .FALSE.) THEN
c     calcul de H4etoi(U), remplace U par H4etoi(U)
      CALL H4etoi(nT,U)
      ENDIF
c     affecte a hUetoile[,4] le resultat
      DO 220, i=1,nT
         hUetoi(i,4)=U(i)
 220      CONTINUE
c     remet les bonnes valeurs dans U
      DO 230, i=1,nT
```

```
          U(i)=Uavant(i)
 230   CONTINUE
c      i=5...................
c      On sauvegarde les valeurs de U dans Uavant
       DO 240, i=1,nT
          Uavant(i)=U(i)
 240      CONTINUE
       IF (isa .EQV. .TRUE.) THEN
c      calcul de H5isa(U), remplace U par H5isa(U)
       CALL H5isa(nT,U)
       ENDIF
       IF (isa .EQV. .FALSE.) THEN
c      calcul de H5etoi(U), remplace U par H5etoi(U)
       CALL H5etoi(nT,U)
       ENDIF
c      affecte a hUetoile[,5] le resultat
       DO 250, i=1,nT
          hUetoi(i,5)=U(i)
 250      CONTINUE
c      remet les bonnes valeurs dans U
       DO 260, i=1,nT
          U(i)=Uavant(i)
 260   CONTINUE
c      i=6...................
c      On sauvegarde les valeurs de U dans Uavant
       DO 270, i=1,nT
          Uavant(i)=U(i)
 270      CONTINUE
       IF (isa .EQV. .TRUE.) THEN
c      calcul de H6isa(U), remplace U par H6isa(U)
       CALL H6isa(nT,U)
       ENDIF
       IF (isa .EQV. .FALSE.) THEN
c      calcul de H6etoi(U), remplace U par H6etoi(U)
       CALL H6etoi(nT,U)
       ENDIF
c      affecte a hUetoile[,6] le resultat
       DO 280, i=1,nT
          hUetoi(i,6)=U(i)
 280      CONTINUE
c      remet les bonnes valeurs dans U
       DO 290, i=1,nT
          U(i)=Uavant(i)
 290   CONTINUE
c      i=7...................
c      On sauvegarde les valeurs de U dans Uavant
       DO 300, i=1,nT
          Uavant(i)=U(i)
 300      CONTINUE
       IF (isa .EQV. .TRUE.) THEN
c      calcul de H7isa(U), remplace U par H7isa(U)
       CALL H7isa(nT,U)
       ENDIF
       IF (isa .EQV. .FALSE.) THEN
c      calcul de H7etoi(U), remplace U par H7etoi(U)
       CALL H7etoi(nT,U)
       ENDIF
c      affecte a hUetoile[,7] le resultat
       DO 310, i=1,nT
          hUetoi(i,7)=U(i)
 310      CONTINUE
c      remet les bonnes valeurs dans U
       DO 320, i=1,nT
          U(i)=Uavant(i)
 320   CONTINUE
```

```
c     i=8...................
c     On sauvegarde les valeurs de U dans Uavant
      DO 330, i=1,nT
         Uavant(i)=U(i)
 330      CONTINUE
      IF (isa .EQV. .TRUE.) THEN
c     calcul de H8isa(U), remplace U par H8isa(U)
      CALL H8isa(nT,U)
      ENDIF
      IF (isa .EQV. .FALSE.) THEN
c     calcul de H8etoi(U), remplace U par H8etoi(U)
      CALL H8etoi(nT,U)
      ENDIF
c     affecte a hUetoile[,8] le resultat
      DO 340, i=1,nT
         hUetoi(i,8)=U(i)
 340      CONTINUE
c     remet les bonnes valeurs dans U
      DO 350, i=1,nT
          U(i)=Uavant(i)
 350   CONTINUE
c     i=9...................
c     On sauvegarde les valeurs de U dans Uavant
      DO 360, i=1,nT
         Uavant(i)=U(i)
 360      CONTINUE
      IF (isa .EQV. .TRUE.) THEN
c     calcul de H9isa(U), remplace U par H9isa(U)
      CALL H9isa(nT,U)
      ENDIF
      IF (isa .EQV. .FALSE.) THEN
c     calcul de H9etoi(U), remplace U par H9etoi(U)
      CALL H9etoi(nT,U)
      ENDIF
c     affecte a hUetoile[,9] le resultat
      DO 370, i=1,nT
         hUetoi(i,9)=U(i)
 370      CONTINUE
c     remet les bonnes valeurs dans U
      DO 380, i=1,nT
          U(i)=Uavant(i)
 380   CONTINUE
c     i=10..................
c     On sauvegarde les valeurs de U dans Uavant
      DO 390, i=1,nT
         Uavant(i)=U(i)
 390      CONTINUE
      IF (isa .EQV. .TRUE.) THEN
c     calcul de H10isa(U), remplace U par H10isa(U)
      CALL H10isa(nT,U)
      ENDIF
      IF (isa .EQV. .FALSE.) THEN
c     calcul de H10eto(U), remplace U par H10eto(U)
      CALL H10eto(nT,U)
      ENDIF
c     affecte a hUetoile[,10] le resultat
      DO 400, i=1,nT
         hUetoi(i,10)=U(i)
 400  CONTINUE
c     remet les bonnes valeurs dans U
      DO 410, i=1,nT
          U(i)=Uavant(i)
 410   CONTINUE
c     On cree la matrice hKetoile
      DO 430, i=1,Kp
```


```
          hKtemp=DBLE(0.0)
          DO 420, k=1,nT
              hKtemp=hKtemp+hUetoi(k,i)
 420          CONTINUE
          hKetoi(i)=hKtemp/(DSQRT(DBLE(nT)))
 430      CONTINUE
c     hK' c'est le h~Kchap du memoire:hK[k]=(1/sqrt(nT))*\sum_{t=1}^{nT} h_k (U_t)
c     On remplit la matrice hU
c     i=1.....................
c     On sauvegarde les valeurs de U dans Uavant
      DO 440, i=1,nT
          Uavant(i)=U(i)
 440   CONTINUE
c     calcul de H1(U), remplace U par H1(U)
      CALL H1(nT,U)
c     affecte a hU[,1] le resultat
      DO 450, i=1,nT
          hU(i,1)=U(i)
 450   CONTINUE
c     remet les bonnes valeurs dans U
      DO 460, i=1,nT
          U(i)=Uavant(i)
 460   CONTINUE
c     i=2.....................
c     On sauvegarde les valeurs de U dans Uavant
      DO 470, i=1,nT
          Uavant(i)=U(i)
 470   CONTINUE
c     calcul de H2(U), remplace U par H2(U)
      CALL H2(nT,U)
c     affecte a hU[,2] le resultat
      DO 480, i=1,nT
          hU(i,2)=U(i)
 480   CONTINUE
c     remet les bonnes valeurs dans U
      DO 490, i=1,nT
          U(i)=Uavant(i)
 490   CONTINUE
c     i=3.....................
c     On sauvegarde les valeurs de U dans Uavant
      DO 500, i=1,nT
          Uavant(i)=U(i)
 500   CONTINUE
c     calcul de H3(U), remplace U par H3(U)
      CALL H3(nT,U)
c     affecte a hU[,3] le resultat
      DO 510, i=1,nT
          hU(i,3)=U(i)
 510   CONTINUE
c     remet les bonnes valeurs dans U
      DO 520, i=1,nT
          U(i)=Uavant(i)
 520   CONTINUE
c     i=4.....................
c     On sauvegarde les valeurs de U dans Uavant
      DO 530, i=1,nT
          Uavant(i)=U(i)
 530   CONTINUE
c     calcul de H4(U), remplace U par H4(U)
      CALL H4(nT,U)
c     affecte a hU[,4] le resultat
      DO 540, i=1,nT
          hU(i,4)=U(i)
 540   CONTINUE
c     remet les bonnes valeurs dans U
```

```
      DO 550, i=1,nT
          U(i)=Uavant(i)
 550   CONTINUE
c     i=5.....................
c     On sauvegarde les valeurs de U dans Uavant
      DO 560, i=1,nT
          Uavant(i)=U(i)
 560   CONTINUE
c     calcul de H5(U), remplace U par H5(U)
      CALL H5(nT,U)
c     affecte a hU[,5] le resultat
      DO 570, i=1,nT
          hU(i,5)=U(i)
 570   CONTINUE
c     remet les bonnes valeurs dans U
      DO 580, i=1,nT
          U(i)=Uavant(i)
 580   CONTINUE
c     i=6.....................
c     On sauvegarde les valeurs de U dans Uavant
      DO 590, i=1,nT
          Uavant(i)=U(i)
 590   CONTINUE
c     calcul de H6(U), remplace U par H6(U)
      CALL H6(nT,U)
c     affecte a hU[,6] le resultat
      DO 600, i=1,nT
          hU(i,6)=U(i)
 600   CONTINUE
c     remet les bonnes valeurs dans U
      DO 610, i=1,nT
          U(i)=Uavant(i)
 610   CONTINUE
c     i=7.....................
c     On sauvegarde les valeurs de U dans Uavant
      DO 620, i=1,nT
          Uavant(i)=U(i)
 620   CONTINUE
c     calcul de H7(U), remplace U par H7(U)
      CALL H7(nT,U)
c     affecte a hU[,7] le resultat
      DO 630, i=1,nT
          hU(i,7)=U(i)
 630   CONTINUE
c     remet les bonnes valeurs dans U
      DO 640, i=1,nT
          U(i)=Uavant(i)
 640   CONTINUE
c     i=8.....................
c     On sauvegarde les valeurs de U dans Uavant
      DO 650, i=1,nT
          Uavant(i)=U(i)
 650   CONTINUE
c     calcul de H8(U), remplace U par H8(U)
      CALL H8(nT,U)
c     affecte a hU[,8] le resultat
      DO 660, i=1,nT
          hU(i,8)=U(i)
 660   CONTINUE
c     remet les bonnes valeurs dans U
      DO 670, i=1,nT
          U(i)=Uavant(i)
 670   CONTINUE
c     i=9.....................
c     On sauvegarde les valeurs de U dans Uavant
```

```
      DO 680, i=1,nT
          Uavant(i)=U(i)
 680   CONTINUE
c      calcul de H9(U), remplace U par H9(U)
      CALL H9(nT,U)
c      affecte a hU[,9] le resultat
      DO 690, i=1,nT
          hU(i,9)=U(i)
 690   CONTINUE
c      remet les bonnes valeurs dans U
      DO 700, i=1,nT
          U(i)=Uavant(i)
 700   CONTINUE
c      i=10.....................
c      On sauvegarde les valeurs de U dans Uavant
      DO 710, i=1,nT
          Uavant(i)=U(i)
 710   CONTINUE
c      calcul de H10(U), remplace U par H10(U)
      CALL H10(nT,U)
c      affecte a hU[,10] le resultat
      DO 720, i=1,nT
          hU(i,10)=U(i)
 720   CONTINUE
c      remet les bonnes valeurs dans U
      DO 730, i=1,nT
          U(i)=Uavant(i)
 730   CONTINUE
c      On cree la matrice hK
       hKtemp=DBLE(0.0)
       DO 750, i=1,Kp
          hKtemp=DBLE(0.0)
          DO 740, k=1,nT
             hKtemp=hKtemp+hU(k,i)
 740         CONTINUE
          hK(i)=hKtemp/DSQRT(DBLE(nT))
 750      CONTINUE
c      On calcule le K optimal par la methode de Ledwina: (critere S2): (KL1)
          inter=DBLE(0.0)
          DO 760, i=1,Kp
c mettre vecteu(i)=(hK(i))**2 si on veut calculer hKchapeau avec les polynomes non
     modifies
c mettre vecteu(i)=(hKetoi(i))**2 si on veut calculer hKchapeau avec les polynomes
     modifies
             vecteu(i)=(hKetoi(i))**2
 760   CONTINUE
       DO 770, k=1,dnT
             inter=inter+vecteu(k)
             Ledwi1(k)=inter-DBLE(k)*penal
 770   CONTINUE
       mxLed1=Ledwi1(1)
       DO 780, i=1,(dnT-1)
             IF (Ledwi1(i+1) .GT. mxLed1) mxLed1=Ledwi1(i+1)
 780   CONTINUE
        DO 790, i=1,dnT
             Ledwi1(i)=Ledwi1(i)-mxLed1
 790   CONTINUE
       DO 810, i=1,dnT
             IF (Ledwi1(i) .EQ. DBLE(0.0)) THEN
                ordre=i
                GO TO 815
             ENDIF
 810   CONTINUE
 815   vKoLe1=ordre
       KoLed1(j)=vKoLe1
```

```
c     On calcule le K optimal par la methode de Ledwina: (critere S2): (KL2)
       inter=DBLE(0.0)
      DO 820, k=2,dnT
          inter=inter+vecteu(k)
          Ledwi2(k-1)=inter-DBLE(k)*penal
 820  CONTINUE
      mxLed2=Ledwi2(1)
      DO 830, i=1,(dnT-2)
            IF (Ledwi2(i+1) .GT. mxLed2) mxLed2=Ledwi2(i+1)
 830  CONTINUE
       DO 840, i=1,(dnT-1)
            Ledwi2(i)=Ledwi2(i)-mxLed2
 840        CONTINUE
      DO 860, i=1,(dnT-1)
            IF (Ledwi2(i) .EQ. DBLE(0.0)) THEN
               ordre2=i
              GO TO 865
            ENDIF
 860  CONTINUE
 865  vKoLe2=ordre2+1
      KoLed2(j)=vKoLe2
c      stat[k]:c'est la statistique de mon test calculee avec les hketoile
      DO 870, i=1,Kp
          vectoi(i)=(hKetoi(i))**2
 870      CONTINUE
      inter=DBLE(0.0)
      DO 880, k=1,Kp
          inter=inter+vectoi(k)
          stat(k)=inter
          IF (stat(k) .GT. khi(k)) compt(k)=compt(k)+1
          IF (stat(k) .GT. khi2(k)) compt2(k)=compt2(k)+1
 880      CONTINUE
c     Calcul de statLedwi1
      stLed1=stat(vKoLe1)
      IF (stLed1 .GT. QuLed1) coLed1=coLed1+1
      IF (stLed1 .GT. QuLe12) coLe12=coLe12+1
      snLed1(j)=stLed1
c     Calcul de statLedwi2
      stLed2=stat(vKoLe2)
      IF (stLed2 .GT. QuLed2) coLed2=coLed2+1
      IF (stLed2 .GT. QuLe22) coLe22=coLe22+1
      snLed2(j)=stLed2
      DO 890, i=1,Kp
          statn(j,i)=stat(i)
 890      CONTINUE
c     On calcule la stat de BD
      DO 900, i=1,nT
            Zavant(i)=Z(i)
 900      CONTINUE
          nTul=nT
          IFAIL=0
      CALL M01CAF(Z, 1, nTul, 'A', IFAIL)
      DO 910, i=1,nT
          BDa(i)=Z(i)
 910      CONTINUE
          moy=DBLE(0.0)
      DO 920, i=1,nT
          Z(i)=Zavant(i)
          moy=moy+Z(i)
 920      CONTINUE
          moy=moy/DBLE(nT)
      DO 930, i=1,nT
          BDb(i)=BDa(i)
 930      CONTINUE
      DO 940, i=1,nT
```

```
          BDc(i)=(BDb(i)-moy)**2
 940      CONTINUE
      DO 950, i=1,nT
          BDd(i)=(DBLE(i)-DBLE(0.375))/(DBLE(nT)+DBLE(0.250))
 950      CONTINUE
      CALL qnorm(nT,BDd,DBLE(0),DBLE(1))
      sumbd=DBLE(0.0)
      sumc=DBLE(0.0)
      sumd2=DBLE(0.0)
      DO 960, i=1,nT
          sumbd=sumbd+(BDb(i)*BDd(i))
 960      CONTINUE
      DO 970, i=1,nT
          sumc=sumc+BDc(i)
 970      CONTINUE
      DO 980, i=1,nT
          sumd2=sumd2+(BDd(i))**2
 980      CONTINUE
c     J'ai rajoute ca sur la demande de Ducharme
          sumc=DBLE(nT)*sigch2
          statBD=(sumbd**2)/(sumc*sumd2)
          IF (statBD .LT. QuBD) contBD=contBD+1
          IF (statBD .LT. QuBD2) conBD2=conBD2+1
          stanBD(j)=statBD
c     On calcule la stat de AD
      DO 990, i=1,nT
          Davan2(i)=D(i)
 990      CONTINUE
      CALL pnorm(nT,D,DBLE(0),DBLE(1))
      DO 1000, i=1,nT
          ADa(i)=D(i)
 1000     CONTINUE
      DO 1010, i=1,nT
          D(i)=Davan2(i)
 1010     CONTINUE
          IFAIL=0
      CALL M01CAF(ADa, 1, nTul, 'A', IFAIL)
      DO 1020, i=1,nT
          ADb(i)=ADa(i)
 1020     CONTINUE
      sumAD=DBLE(0.0)
      DO 1030, i=1,nT
          sumAD=sumAD+(DBLE(2.0)*DBLE(i)-DBLE(1.0))*DLOG(ADb(i))+(DBLE(2)
     +      *DBLE(nT)+DBLE(1.0)-DBLE(2.0)*DBLE(i))*DLOG(DBLE(1.0)-ADb(i))
 1030     CONTINUE
          statAD=-(DBLE(nT)+sumAD/DBLE(nT))
          statAD=statAD*(DBLE(1.0)+DBLE(0.75)/DBLE(nT)+DBLE(2.25)
     +          /(DBLE(nT*nT)))
          IF (statAD .GT. QuAD) contAD=contAD+1
          IF (statAD .GT. QuAD2) conAD2=conAD2+1
          stanAD(j)=statAD
c     On calcule la stat de Jarque et Bera
          m2=DBLE(0.0)
          m3=DBLE(0.0)
          m4=DBLE(0.0)
          DO 1035, i=1,nT
              m2=m2+(Z(i))**2
              m3=m3+(Z(i))**3
              m4=m4+(Z(i))**4
 1035     CONTINUE
              m2=m2/DBLE(nT)
              m3=m3/DBLE(nT)
              m4=m4/DBLE(nT)
              g1a=m3**2
              g1b=m2**3
```


```
            g1=g1a/g1b
            g2=m4/(m2**2)
            statJB=DBLE(nT)*(g1/DBLE(6)+((g2-DBLE(3))**2)/DBLE(24))
     $       + nT*((DBLE(3.0)*(moy**2))/(DBLE(2.0)*m2)-(m3*moy)/(m2**2))
          IF (statJB .GT. QuJB) contJB=contJB+1
          IF (statJB .GT. QuJB2) conJB2=conJB2+1
          stanJB(j)=statJB
c     Fin de la grande boucle
 9000     CONTINUE
c-------------------------------------------------------------------
c ON TERMINE PAR LA PUISSANCE, LES QUANTILES ET ON SORT LES VALEURS
c-------------------------------------------------------------------
c     On calcule la puissance des tests en %
      DO 1040, k=1,Kp
         Puiss(k)=(DBLE(compt(k))*DBLE(100))/DBLE(nbcle)
         Puiss2(k)=(DBLE(compt2(k))*DBLE(100))/DBLE(nbcle)
 1040    CONTINUE
      PsLed1=(DBLE(coLed1)*DBLE(100))/DBLE(nbcle)
      PsLe12=(DBLE(coLe12)*DBLE(100))/DBLE(nbcle)
      PsLed2=(DBLE(coLed2)*DBLE(100))/DBLE(nbcle)
      PsLe22=(DBLE(coLe22)*DBLE(100))/DBLE(nbcle)
      PuisBD=(DBLE(contBD)*DBLE(100))/DBLE(nbcle)
      PuiBD2=(DBLE(conBD2)*DBLE(100))/DBLE(nbcle)
      PuisAD=(DBLE(contAD)*DBLE(100))/DBLE(nbcle)
      PuiAD2=(DBLE(conAD2)*DBLE(100))/DBLE(nbcle)
      PuisJB=(DBLE(contJB)*DBLE(100))/DBLE(nbcle)
      PuiJB2=(DBLE(conJB2)*DBLE(100))/DBLE(nbcle)
c     On calcule les Quantiles
      val=IDInt(DBLE(nbcle)*alpha)
      val2=IDInt(DBLE(nbcle)*alpha2)
      DO 1060, i=1,nbcle
         DO 1050, k=1,Kp
            snsort(i,k)=DBLE(0.0)
 1050       CONTINUE
 1060    CONTINUE
      nul=nbcle
      DO 1070, i=1,nbcle
         snsorv(i)=DBLE(0.0)
 1070    CONTINUE
      DO 1100, k=1,Kp
         DO 1080, i=1,nbcle
            snsorv(i)=statn(i,k)
 1080       CONTINUE
            IFAIL=0
          CALL M01CAF(snsorv, 1, nul, 'A', IFAIL)
          DO 1090, i=1,nbcle
             snsort(i,k)=snsorv(i)
 1090        CONTINUE
 1100    CONTINUE
       DO 1110, i=1,Kp
          Qucalc(i)=snsort(nbcle-val,i)
          Qucal2(i)=snsort(nbcle-val2,i)
 1110     CONTINUE
       DO 1120, i=1,nbcle
          snLe1s(i)=snLed1(i)
 1120     CONTINUE
          IFAIL=0
       CALL M01CAF(snLe1s, 1, nul, 'A', IFAIL)
       QLed1c=snLe1s(nbcle-val)
       QLed12=snLe1s(nbcle-val2)
       DO 1130, i=1,nbcle
          snLe2s(i)=snLed2(i)
 1130     CONTINUE
          IFAIL=0
       CALL M01CAF(snLe2s, 1, nul, 'A', IFAIL)
```

```
      QLed2c=snLe2s(nbcle-val)
      QLed22=snLe2s(nbcle-val2)
      DO 1140, i=1,nbcle
         snBDso(i)=stanBD(i)
 1140    CONTINUE
         IFAIL=0
      CALL M01CAF(snBDso, 1, nul, 'A', IFAIL)
      QBDcal=snBDso(val)
      QBDca2=snBDso(val2)
      DO 1150, i=1,nbcle
         snADso(i)=stanAD(i)
 1150    CONTINUE
         IFAIL=0
      CALL M01CAF(snADso, 1, nul, 'A', IFAIL)
      QADcal=snADso(nbcle-val)
      QADca2=snADso(nbcle-val2)
      DO 1155, i=1,nbcle
         snJBso(i)=stanJB(i)
 1155 CONTINUE
      IFAIL=0
      CALL M01CAF(snJBso, 1, nul, 'A', IFAIL)
      QJBcal=snJBso(nbcle-val)
      QJBca2=snJBso(nbcle-val2)
c     On calcule les rangs des valeurs des statistiques
      rgLed1(1)=mini(nbcle,snLed1)
      rgLed1(2)=maxi(nbcle,snLed1)
      rgLed2(1)=mini(nbcle,snLed2)
      rgLed2(2)=maxi(nbcle,snLed2)
      rgBD(1)=mini(nbcle,stanBD)
      rgBD(2)=maxi(nbcle,stanBD)
      rgAD(1)=mini(nbcle,stanAD)
      rgAD(2)=maxi(nbcle,stanAD)
      rgJB(1)=mini(nbcle,stanJB)
      rgJB(2)=maxi(nbcle,stanJB)
c     On calcule les moyennes et ecart-types des valeurs des statistiques
      mecLe1(1)=meanp(nbcle,snLed1)
      mecLe1(2)=DSQRT(var(nbcle,snLed1))
      mecLe2(1)=meanp(nbcle,snLed2)
      mecLe2(2)=DSQRT(var(nbcle,snLed2))
      mecBD(1)=meanp(nbcle,stanBD)
      mecBD(2)=DSQRT(var(nbcle,stanBD))
      mecAD(1)=meanp(nbcle,stanAD)
      mecAD(2)=DSQRT(var(nbcle,stanAD))
      mecJB(1)=meanp(nbcle,stanJB)
      mecJB(2)=DSQRT(var(nbcle,stanJB))
c     On calcule le nombre de fois ou chaque polynome a ete choisi par les procedures
    de Ledwina
      DO 1160, i=1,10
         frKLe1(i)=0
 1160    CONTINUE
      DO 1180, i=1,10
         DO 1170, k=1,nbcle
            IF (KoLed1(k) .EQ. i) frKLe1(i)=frKLe1(i)+1
 1170       CONTINUE
 1180    CONTINUE
      DO 1190, i=1,10
         frKLe2(i)=0
 1190    CONTINUE
      DO 1210, i=1,10
         DO 1200, k=1,nbcle
            IF (KoLed2(k) .EQ. i) frKLe2(i)=frKLe2(i)+1
 1200       CONTINUE
 1210    CONTINUE
         DO 1220, i=1,nbcle
            matric(i,1)=i
```

```
              matric(i,2)=statn(i,1)
              matric(i,3)=statn(i,2)
              matric(i,4)=statn(i,3)
              matric(i,5)=statn(i,4)
              matric(i,6)=statn(i,5)
              matric(i,7)=statn(i,6)
              matric(i,8)=statn(i,7)
              matric(i,9)=statn(i,8)
              matric(i,10)=statn(i,9)
              matric(i,11)=statn(i,10)
              matric(i,12)=KoLed1(i)
              matric(i,13)=snLed1(i)
              matric(i,14)=KoLed2(i)
              matric(i,15)=snLed2(i)
              matric(i,16)=stanBD(i)
              matric(i,17)=stanAD(i)
              matric(i,18)=stanJB(i)
 1220         CONTINUE
c       On ecrit les resultats dans resultat.ijkl.xc
        OPEN(UNIT=17, FILE=filexc, STATUS='NEW')
           WRITE(17,*) 'numero-simul R1 R2 R3 R4 R5 R6 R7 R8 R9 R10 Kchap1
     + Ledwi1 Kchap2 Ledwi2 BD AD JB'
           DO 1230, i=1,nbcle
              WRITE(17,1235) (matric(i,k),k=1,18)
1230      CONTINUE
1235         FORMAT(F6.0,10F14.8,F6.0,F14.8,F6.0,4F14.8)
        CLOSE(UNIT=17)
c       On ecrit quelques resultats dans para.ijkl
        OPEN(UNIT=14, FILE=paramf , STATUS='OLD', ACCESS='append' )
        WRITE(14,*) 'Puiss avec alpha=',alpha,':'
        WRITE(14,*) Puiss
        WRITE(14,*) 'Puiss avec alpha2=',alpha2,':'
        WRITE(14,*) Puiss2
        WRITE(14,*) 'Quantutilises avec alpha=',alpha,':'
        WRITE(14,*) Qtutil
        WRITE(14,*) 'Quantutilises avec alpha2=',alpha2,':'
        WRITE(14,*) Qtuti2
        WRITE(14,*) 'Quantcalcules avec alpha=', alpha, ':'
        WRITE(14,*) Qucalc
        WRITE(14,*) 'Quantcalcules avec alpha2=',alpha2, ':'
        WRITE(14,*) Qucal2
        WRITE(14,*) 'QuantLedwi1utilise avec alpha=', alpha, ':'
        WRITE(14,*) QLed1u
        WRITE(14,*) 'QuantLedwi1utilise avec alpha2=', alpha2, ':'
        WRITE(14,*) QLe12u
        WRITE(14,*) 'QuantLedwi1calcule avec alpha=', alpha, ':'
        WRITE(14,*) QLed1c
        WRITE(14,*) 'QuantLedwi1calcule avec alpha2=', alpha2, ':'
        WRITE(14,*) QLed12
        WRITE(14,*) 'PuissLedwi1 avec alpha=', alpha, ':'
        WRITE(14,*) PsLed1
        WRITE(14,*) 'PuissLedwi1 avec alpha2=', alpha2, ':'
        WRITE(14,*) PsLe12
        WRITE(14,*) 'freqKLedwi1:'
        WRITE(14,*) frKLe1
        WRITE(14,*) 'QuantLedwi2utilise avec alpha=', alpha, ':'
        WRITE(14,*) QLed2u
        WRITE(14,*) 'QuantLedwi2utilise avec alpha2=', alpha2, ':'
        WRITE(14,*) QLe22u
        WRITE(14,*) 'QuantLedwi2calcule avec alpha=', alpha, ':'
        WRITE(14,*) QLed2c
        WRITE(14,*) 'QuantLedwi2calcule avec alpha2=', alpha2, ':'
        WRITE(14,*) QLed22
        WRITE(14,*) 'PuissLedwi2 avec alpha=', alpha, ':'
        WRITE(14,*) PsLed2
```


```
      WRITE(14,*) 'PuissLedwi2 avec alpha2=', alpha2, ':'
      WRITE(14,*) PsLe22
      WRITE(14,*) 'freqKLedwi2:'
      WRITE(14,*) frKLe2
      WRITE(14,*) 'QuantBDutilise avec alpha=', alpha, ':'
      WRITE(14,*) QBDu
      WRITE(14,*) 'QuantBDutilise avec alpha2=', alpha2, ':'
      WRITE(14,*) QBD2u
      WRITE(14,*) 'QuantBDcalcule avec alpha=', alpha, ':'
      WRITE(14,*) QBDcal
      WRITE(14,*) 'QuantBDcalcule avec alpha2=', alpha2, ':'
      WRITE(14,*) QBDca2
      WRITE(14,*) 'PuissBD avec alpha=', alpha, ':'
      WRITE(14,*) PuisBD
      WRITE(14,*) 'PuissBD avec alpha2=', alpha2, ':'
      WRITE(14,*) PuiBD2
      WRITE(14,*) 'QuantADutilise avec alpha=', alpha, ':'
      WRITE(14,*) QADu
      WRITE(14,*) 'QuantADutilise avec alpha2=', alpha2, ':'
      WRITE(14,*) QAD2u
      WRITE(14,*) 'QuantADcalcule avec alpha=', alpha, ':'
      WRITE(14,*) QADcal
      WRITE(14,*) 'QuantADcalcule avec alpha2=', alpha2, ':'
      WRITE(14,*) QADca2
      WRITE(14,*) 'PuissAD avec alpha=', alpha, ':'
      WRITE(14,*) PuisAD
      WRITE(14,*) 'PuissAD avec alpha2=', alpha2, ':'
      WRITE(14,*) PuiAD2
      WRITE(14,*) 'QuantJButilise avec alpha=', alpha, ':'
      WRITE(14,*) QJBu
      WRITE(14,*) 'QuantJButilise avec alpha2=', alpha2, ':'
      WRITE(14,*) QJB2u
      WRITE(14,*) 'QuantJBcalcule avec alpha=', alpha, ':'
      WRITE(14,*) QJBcal
      WRITE(14,*) 'QuantJBcalcule avec alpha2=', alpha2, ':'
      WRITE(14,*) QJBca2
      WRITE(14,*) 'PuissJB avec alpha=', alpha, ':'
      WRITE(14,*) PuisJB
      WRITE(14,*) 'PuissJB avec alpha2=', alpha2, ':'
      WRITE(14,*) PuiJB2
      CLOSE(UNIT=14)
c     On ecrit les resultats dans tableau.txt
      OPEN(UNIT=13, FILE=filtab , STATUS='OLD' , ACCESS='append')
      WRITE(13,1500) nbcle, nT, p, q,loi,alpha, phi(1), phi(2),
     + teta(1),
     + teta(2),
     + Puiss(1),
     + Puiss(2), Puiss(3), Puiss(4), Puiss(5), PsLed1, PsLed2, PuisBD,
     + PuisAD, PuisJB, QLed1c, QLed2c
      WRITE(13,1500) nbcle, nT, p, q,loi,alpha2, phi(1), phi(2),
     + teta(1)
     + , teta(2),
     + Puiss2(1), Puiss2(2), Puiss2(3), Puiss2(4), Puiss2(5), PsLe12,
     + PsLe22, PuiBD2, PuiAD2, PuiJB2, QLed12, QLed22
 1500 FORMAT(I6,I4,2I2,I3,F7.3,16F7.2)
      CLOSE(UNIT=13)
      END
c-------------------------------------------------------------------------
c                        Fin du programme
c-------------------------------------------------------------------------
```

A.8. LES PROGRAMMES SIMULARMAPQ.F

Programme simulaAR.f

```
c Debut-Commentaires
c Nom de la sous-routine: simAR
c Entrees:
c p (entier) ordre de la partie AR
c n (entier) longueur de la serie souhaitee
c marret (entier) rang d'arret dans la random shock method
c Mind (entier) induction period
c loi (entier) specifie la loi des erreurs ou innovations
c df1 (entier)
c df2 (entier)
c lambda (double)
c loi5b, loi5k, loi6p, loi6q
c loi7g, loi7d, loi8p, loi8q, loi9a, loi9b
c loi10b, loi10l, loi11a, loi11k, loi13g, loi13d
c loi14l, loi16p, loi16d, loi17p, loi17m
c loi18g, loi18d, loi19a, loi19b
c Wtip (vecteur, double) de longeur p qui contiendra les p donnees initiales de la
     serie
c shocks (vecteur, double) des marret+p innovations genere par shock.f
c psi (vecteur, double) de longeur marret+1 tel que defini dans Burn, calcule par
     ARpsi.f
c phi (vecteur, double) de longueur p
c phi2 (vecteur, double) de longueur marret
c YtpMn (vecteur, double) de longueur p+Mind+n
c Atn (vecteur, double) de longueur n+Mind
c EspSU (double) : esperance de la loi SU
c EspSB (double) : esperance de la loi S
c EspS (double) : esperance de la loi S
c Sorties: void
c Description:
c Ce programme realise les etapes 2 et 3 de Simulation Algorithm 1, de l'article de
     Burn
c Cette sous-routine remplace le vecteur YtpMn en entree par p+M+n donnees simulees
c d'un modele AR(p) de vecteur d'innovations de loi donnee par l'entier loi ,
c les p premieres valeurs sont les p valeurs initiales de l'etape 1 de Burn,
c les M valeurs suivantes sont les valeurs a ecarter de la warm-up period de longueur
     Mind
c donc seules les n dernieres valeurs devront etre conservees pour la suite de la
     simulation
c elle renvoie aussi le vecteur shocks des marret+p innovations utilisees dans l'
     algorithme
c Initialisation 1 de Burn, ainsi que le vecteur Atn des n+Mind random shocks At de l'
     etape 2 de Burn.
c Utilisation dans une fonction main:
c-------------------------------------
c       PROGRAM main
c       INTEGER p, n, marret, Mind, loi, df1, df2
c       DOUBLE PRECISION lambda, EspSU, EspSB, EspS
c       PARAMETER(p=2,n=100,marret=200,Mind=200,loi=1,
c      +     df1=2,df2=5,lambda=2.0, EspSU=0, EspSB=0, EspS=0)
c       DOUBLE PRECISION loi5b, loi5k, loi6p, loi6q
c       DOUBLE PRECISION loi7g, loi7d, loi8p, loi8q, loi9a, loi9b
c       DOUBLE PRECISION loi10b, loi10l, loi11a, loi11k, loi13g, loi13d
c       DOUBLE PRECISION loi14l, loi16p, loi16d, loi17p, loi17m
c       DOUBLE PRECISION loi18g, loi18d, loi19a, loi19b
c       PARAMETER(loi18g=1.0,loi18d=1.0, loi19a=1.1, loi19b=0.5)
c       PARAMETER(loi16p=0.2,loi16d=5.0, loi17p=0.2,loi17m=3.0)
c       PARAMETER(loi10b=0.2,loi10l=1, loi11a=2.0,loi11k=0.5)
c       PARAMETER(loi13g=1.0,loi13d=1.0, loi14l=0.7)
c       PARAMETER(loi7g=0,loi7d=1, loi8p=2, loi8q=2, loi9a=0,loi9b=2)
c       PARAMETER(loi5b=1, loi5k=1.8, loi6p=4, loi6q=1)
c       DOUBLE PRECISION Wtip(p), shocks(marret+p), psi(marret+1)
c       DOUBLE PRECISION phi(2), YtpMn(p+Mind+n), Atn(n+Mind)
c       DOUBLE PRECISION phi2(marret)
cc       Initialisation de phi
```

```
c        phi(1)=0.1
c        phi(2)=0.2
c        CALL G05CBF(0)
cc        Calcul de YtpMn, shocks, Atn
c        CALL simAR(p,n,marret,Mind,loi,df1,df2,lambda,loi5b, loi5k, loi6p,
c       +  loi6q, loi7g, loi7d, loi8p, loi8q, loi9a, loi9b,
c       +  loi10b, loi10l, loi11a, loi11k, loi13g, loi13d,
c       +  loi14l, loi16p, loi16d, loi17p, loi17m, loi18g, loi18d,
c       +  loi19a, loi19b,
c       +  Wtip,shocks,psi,phi,phi2,YtpMn,Atn,EspSU,EspSB,EspS)
cc        Affichage des valeurs de YtpMn
c        WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de YtpMn'
c        WRITE(UNIT=6,FMT=*) YtpMn
cc        Affichage des valeurs de shocks
c        WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de shocks'
c        WRITE(UNIT=6,FMT=*) shocks
cc        Affichage des valeurs de Atn
c        WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de Atn'
c        WRITE(UNIT=6,FMT=*) Atn
c        END
c        INCLUDE 'rskew.f'
c        INCLUDE 'rlap.f'
c        INCLUDE 'ARpsi.f'
c        INCLUDE 'shock.f'
c        INCLUDE 'rpare.f'
c        INCLUDE 'rspare.f'
c        INCLUDE 'rSU.f'
c        INCLUDE 'rTU.f'
c        INCLUDE 'rSC.f'
c        INCLUDE 'rLC.f'
c        INCLUDE 'rSB.f'
c        INCLUDE 'rS.f'
c-------------------------------------------
c Instructions de compilation: alias f77='fort77'
c f77 -c nom_du_fichier_contenant_la_fonction_main.f
c f77 -c simAR.f -lnag
c f77 nom_du_fichier_contenant_la_fonction_main.o simAR.o -lnag
c Fonctions exterieures appelees:
c ARpsi.f, shock.f, rskew.f, rlap.f  et G05DEF, G05DHF, G05DJF de NAG Mark16
c Auteur: Pierre Lafaye de Micheaux
c Date: 15/02/2001
c Fin-Commentaires
      SUBROUTINE simAR(p,n,marret,Mind,loi,df1,df2,lambda,loi5b,
     +  loi5k, loi6p, loi6q, loi7g, loi7d, loi8p, loi8q, loi9a, loi9b,
     +  loi10b, loi10l, loi11a, loi11k, loi13g, loi13d,
     +  loi14l, loi16p, loi16d, loi17p, loi17m, loi18g, loi18d,
     +  loi19a, loi19b, Wtip,shocks,psi,phi,phi2,YtpMn,Atn,
     +  EspSU,EspSB,EspS)
      INTEGER l, p,n,marret,Mind,loi,df1,df2
      DOUBLE PRECISION Wtip(p), shocks(marret+p),psi(marret+1)
      DOUBLE PRECISION phi(2),phi2(marret)
      DOUBLE PRECISION YtpMn(p+Mind+n)
      DOUBLE PRECISION Atn(n+Mind)
      DOUBLE PRECISION pi, a, b, lambda, Eweibu, EspSU, EspSB, EspS
      DOUBLE PRECISION G05DEF, G05DHF, S14AAF
      DOUBLE PRECISION G05DJF, G05DPF, G05DCF
      DOUBLE PRECISION G05DAF, G05DBF, rS
      DOUBLE PRECISION loi5b, loi5k, loi6p, loi6q
      DOUBLE PRECISION loi7g, loi7d, loi8p, loi8q, loi9a, loi9b
      DOUBLE PRECISION loi10b, loi10l, loi11a, loi11k, loi13g, loi13d
      DOUBLE PRECISION loi14l, loi16p, loi16d, loi17p, loi17m
      DOUBLE PRECISION loi18g, loi18d, loi19a, loi19b
c Calcul des valeurs de psi
      CALL ARpsi(marret,p,psi,phi,phi2)
c Calcul de shocks et Wtip
```

```
      CALL shock(p, marret, loi, Wtip, shocks, psi,
     +  df1, df2, lambda, loi5b, loi5k, loi6p, loi6q,
     +  loi7g, loi7d, loi8p, loi8q, loi9a, loi9b,
     +  loi10b, loi10l, loi11a, loi11k, loi13g, loi13d,
     +  loi14l, loi16p, loi16d, loi17p, loi17m, loi18g, loi18d,
     +  loi19a, loi19b)
c Etape 2 de l'algorithme: on genere n+Mind random shocks
      l=n+Mind
      pi=DBLE(3.14159265358979)
      a=DBLE(0.0)
      b=DBLE(1.0)
c     Ca c'est inutile puisqu'on n'appelle jamais simulAR.f avec loi=0
c      IF (loi .EQ. 0) THEN
c            CALL G05FDF(a,b,l,Atn)
c      DO 10, i=1,l
c            Atn(i)=Atn(i)-a
c 10      CONTINUE
c      ENDIF
      IF (loi .EQ. 1) THEN
         DO 20, i=1,l
         IFAIL=0
            Atn(i)=G05DHF(df1,IFAIL)-DBLE(df1)
 20         CONTINUE
      ENDIF
      IF (loi .EQ. 2) THEN
         DO 30, i=1,l
         IFAIL=0
c                                    l'esperance d'une student est nulle
            Atn(i)=G05DJF(df2,IFAIL)
 30         CONTINUE
      ENDIF
      IF (loi .EQ. 3) THEN
         CALL rskew(l,Atn,lambda)
         DO 40, i=1,l
            Atn(i)=Atn(i) -
     +           dsqrt(DBLE(2.0)/pi)*(lambda/(sqrt(1+lambda*lambda)))
 40      CONTINUE
      ENDIF
      IF (loi .EQ. 4) THEN
c                        l'esperance d'une loi de Laplace est nulle
         CALL rlap(l,Atn)
      ENDIF
      IF (loi .EQ. 5) THEN
         IFAIL=0
         Eweibu=S14AAF(DBLE(1+1/loi5k),IFAIL)/loi5b
         DO 50, i=1,l
         IFAIL=0
            Atn(i)=G05DPF(loi5k,(loi5b)**(-loi5k),IFAIL)-Eweibu
 50         CONTINUE
      ENDIF
      IF (loi .EQ. 6) THEN
         IFAIL=0
         CALL G05FFF(loi6p,loi6q,l,Atn,IFAIL)
         DO 60, i=1,l
            Atn(i)=Atn(i)-loi6p*loi6q
 60      CONTINUE
      ENDIF
      IF (loi .EQ. 7) THEN
         DO 70, i=1,l
            Atn(i)=G05DEF(-loi7g/loi7d,1/loi7d)
     +                          -dexp(-loi7g/loi7d+0.5/(loi7d**2))
 70         CONTINUE
      ENDIF
      IF (loi .EQ. 8) THEN
         IFAIL=0
```

```
         CALL G05FEF(loi8p,loi8q,l,Atn,IFAIL)
         DO 80, i=1,l
            Atn(i)=Atn(i)-loi8p/(loi8p+loi8q)
 80      CONTINUE
      ENDIF
      IF (loi .EQ. 9) THEN
         DO 90, i=1,l
            Atn(i)=G05DAF(loi9a,loi9b)-(loi9a+loi9b)/2
 90      CONTINUE
      ENDIF
      IF (loi .EQ. 10) THEN
         DO 100, i=1,l
            Atn(i)=G05DBF(1/loi10b)-1/loi10b
 100      CONTINUE
      ENDIF
      IF (loi .EQ. 11) THEN
         CALL rpare(loi11a,loi11k,l,Atn)
         DO 110, i=1,l
            Atn(i)=Atn(i)-loi11k*loi11a/(loi11a-1)
 110      CONTINUE
      ENDIF
      IF (loi .EQ. 12) THEN
         CALL rspare(l,Atn)
         DO 120, i=1,l
            Atn(i)=Atn(i)-1
 120      CONTINUE
      ENDIF
      IF (loi .EQ. 13) THEN
         CALL rSU(loi13g,loi13d,l,Atn)
         DO 130, i=1,l
            Atn(i)=Atn(i)-EspSU
 130      CONTINUE
      ENDIF
      IF (loi .EQ. 14) THEN
c                        l'esperance d'une loi TU(l) = 0
         CALL rTU(loi14l,l,Atn)
      ENDIF
      IF (loi .EQ. 15) THEN
         DO 140, i=1,l
c                  l'esperance d'une loi Logistic = 0
            Atn(i)=G05DCF(a,b)
 140      CONTINUE
      ENDIF
      IF (loi .EQ. 16) THEN
c                  l'esperance d'une loi SC(p,d) = 0
         CALL rSC(loi16p,loi16d,l,Atn)
      ENDIF
      IF (loi .EQ. 17) THEN
         CALL rLC(loi17p,loi17m,l,Atn)
         DO 150, i=1,l
            Atn(i)=Atn(i) -loi17p*loi17m
 150     CONTINUE
      ENDIF
      IF (loi .EQ. 18) THEN
         CALL rSB(loi18g,loi18d,l,Atn)
         DO 160, i=1,l
            Atn(i)=Atn(i)-EspSB
 160      CONTINUE
      ENDIF
      IF (loi .EQ. 19) THEN
         DO 170, i=1,l
            Atn(i)=rS(loi19a,loi19b)-EspS
 170      CONTINUE
      ENDIF
c Etape 3 de l'algorithme
```

```
c YtpMn:          Y(1-p), Y(2-p), ..., Y(0), Y(1) , ..., Y(Mind+n)
c YtpMn(i), i=:    1       2            p      p+1          p+Mind+n
      DO 180, i=1,(p+Mind+n)
         YtpMn(i)=DBLE(0.0)
 180  CONTINUE
c Wtp:             Y(1-p), Y(2-p), ..., Y(0)
c YtpMn(i), i=:    1       2            p
      DO 190, i=1,p
         YtpMn(i)=Wtip(i)
 190  CONTINUE
      DO 210,i=(p+1),(p+Mind+n)
         DO 200, k=1,p
         YtpMn(i)=YtpMn(i)+phi(k)*YtpMn(i-k)
 200     CONTINUE
c Atn:          eps(1), ..., eps(n+Mind)
c Atn(k), k=:     1            n+Mind
            YtpMn(i)=YtpMn(i)+Atn(-p+i)
 210       CONTINUE
      RETURN
      END
```

Programme simulaMA.f

```
c Debut-Commentaires
c Nom de la sous-routine: simMA
c Entrees:
c q (entier) ordre de la partie MA
c n (entier) longueur de la serie souhaitee
c marret (entier) rang d'arret dans la random shock method
c Mind (entier) induction period
c loi (entier) specifie la loi des erreurs ou innovations
c df1 (entier)
c df2 (entier)
c lambda (double)
c loi5b, loi5k, loi6p, loi6q
c loi7g, loi7d, loi8p, loi8q, loi9a, loi9b
c loi10b, loi10l, loi11a, loi11k, loi13g, loi13d
c loi14l, loi16p, loi16d, loi17p, loi17m
c loi18g, loi18d, loi19a, loi19b
c teta (vecteur, double) de longueur q
c YtMn (vecteur, double) de longueur Mind+n
c Atnq (vecteur, double) de longueur n+Mind+q
c EspSU (double) : esperance de la loi SU
c EspSB (double) : esperance de la loi S
c EspS (double) : esperance de la loi S
c Sorties: void
c Description:
c Ce programme realise les etapes 2 et 3 de Simulation Algorithm 1, de l'article de
     Burn
c Cette sous-routine remplace le vecteur YtMn en entree par M+n donnees simulees
c d'un modele MA(q) de vecteur d'innovations de loi donnee par l'entier loi ,
c les M valeurs suivantes sont les valeurs a ecarter de la warm-up period de longueur
     Mind
c donc seules les n dernieres valeurs devront etre conservees pour la suite de la
     simulation
c elle renvoie aussi le vecteur Atnq des n+Mind+q random shocks At de l'etape 2 de
     Burn.
c Utilisation dans une fonction main:
c-------------------------------------------
c       PROGRAM main
c       INTEGER q, n, marret, Mind, loi, df1, df2
c       DOUBLE PRECISION lambda, EspSU, EspSB, EspS
c       PARAMETER(q=2,n=100,marret=200,Mind=200,loi=1,
c      +     df1=2,df2=5,lambda=2.0, EspSU=0, EspSB=0, EspS=0)
c       DOUBLE PRECISION loi5b, loi5k, loi6p, loi6q
```

```
c       DOUBLE PRECISION loi7g, loi7d, loi8p, loi8q, loi9a, loi9b
c       DOUBLE PRECISION loi10b, loi10l, loi11a, loi11k, loi13g, loi13d
c       DOUBLE PRECISION loi14l, loi16p, loi16d, loi17p, loi17m
c       DOUBLE PRECISION loi18g, loi18d, loi19a, loi19b
c       PARAMETER(loi18g=1.0,loi18d=1.0, loi19a=1.1, loi19b=0.5)
c       PARAMETER(loi16p=0.2,loi16d=5.0, loi17p=0.2,loi17m=3.0)
c       PARAMETER(loi10b=0.2,loi10l=1, loi11a=2.0,loi11k=0.5)
c       PARAMETER(loi13g=1.0,loi13d=1.0, loi14l=0.7)
c       PARAMETER(loi7g=0,loi7d=1, loi8p=2, loi8q=2, loi9a=0,loi9b=2)
c       PARAMETER(loi5b=1, loi5k=1.8, loi6p=4, loi6q=1)
c       DOUBLE PRECISION teta(2), YtMn(Mind+n), Atnq(n+Mind+q)
cc       Initialisation de teta
c       teta(1)=0.1
c       teta(2)=0.5
c       CALL G05CBF(0)
cc       Calcul de YtMn, Atnq
c       CALL simMA(q,n,marret,Mind,loi,df1,df2,lambda,loi5b, loi5k, loi6p,
c      +  loi6q, loi7g, loi7d, loi8p, loi8q, loi9a, loi9b,
c      +  loi10b, loi10l, loi11a, loi11k, loi13g, loi13d,
c      +  loi14l, loi16p, loi16d, loi17p, loi17m, loi18g, loi18d,
c      +  loi19a, loi19b,
c      +       teta,YtMn,Atnq, EspSU,EspSB, EspS)
cc       Affichage des valeurs de YtMn
c       WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de YtMn'
c       WRITE(UNIT=6,FMT=*) YtMn
cc       Affichage des valeurs de Atnq
c       WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de Atnq'
c       WRITE(UNIT=6,FMT=*) Atnq
c       END
c       INCLUDE 'rskew.f'
c       INCLUDE 'rlap.f'
c       INCLUDE 'rpare.f'
c       INCLUDE 'rspare.f'
c       INCLUDE 'rSU.f'
c       INCLUDE 'rTU.f'
c       INCLUDE 'rSC.f'
c       INCLUDE 'rLC.f'
c       INCLUDE 'rSB.f'
c       INCLUDE 'rS.f'
c-------------------------------------
c Instructions de compilation: alias f77='fort77'
c f77 -c nom_du_fichier_contenant_la_fonction_main.f
c f77 -c simMA.f -lnag
c f77 nom_du_fichier_contenant_la_fonction_main.o simMA.o -lnag
c Fonctions exterieures appelees:
c rskew.f, rlap.f et G05DEF, G05DHF, G05DJF de NAG Mark16
c Auteur: Pierre Lafaye de Micheaux
c Date: 15/02/2001
c Fin-Commentaires
      SUBROUTINE simMA (q,n,marret,Mind,loi,df1,df2,lambda,loi5b,
     +  loi5k, loi6p, loi6q, loi7g, loi7d, loi8p, loi8q, loi9a, loi9b,
     +  loi10b, loi10l, loi11a, loi11k, loi13g, loi13d,
     +  loi14l, loi16p, loi16d, loi17p, loi17m, loi18g, loi18d,
     +  loi19a, loi19b,
     +       teta,YtMn,Atnq,EspSU,EspSB,EspS)
      INTEGER l,q,n,marret,Mind,loi,df1,df2
      DOUBLE PRECISION teta(2)
      DOUBLE PRECISION YtMn(Mind+n)
      DOUBLE PRECISION Atnq(n+Mind+q)
      DOUBLE PRECISION pi, a, b, lambda, Eweibu,EspSU, EspSB, EspS
      DOUBLE PRECISION G05DEF, G05DHF, S14AAF
      DOUBLE PRECISION G05DJF, G05DPF, G05DCF
      DOUBLE PRECISION G05DAF, G05DBF, rS
      DOUBLE PRECISION loi5b, loi5k, loi6p, loi6q
      DOUBLE PRECISION loi7g, loi7d, loi8p, loi8q, loi9a, loi9b
```

```
      DOUBLE PRECISION loi10b, loi10l, loi11a, loi11k, loi13g, loi13d
      DOUBLE PRECISION loi14l, loi16p, loi16d, loi17p, loi17m
      DOUBLE PRECISION loi18g, loi18d, loi19a, loi19b
c Etape 2 de l'algorithme: on genere n+Mind+q random shocks
      l=n+Mind+q
      pi=DBLE(3.14159265358979)
      a=DBLE(0.0)
      b=DBLE(1.0)
c     Ca c'est inutile puisqu'on n'appelle jamais simulMA.f avec loi=0
c      IF (loi .EQ. 0) THEN
c            CALL G05FDF(a,b,l,Atnq)
c      DO 10, i=1,l
c            Atnq(i)=Atnq(i)-a
c 10      CONTINUE
c      ENDIF
      IF (loi .EQ. 1) THEN
         DO 20, i=1,l
         IFAIL=0
            Atnq(i)=G05DHF(df1,IFAIL)-DBLE(df1)
 20         CONTINUE
      ENDIF
      IF (loi .EQ. 2) THEN
c                                    l'esperance d'une student est nulle
         DO 30, i=1,l
         IFAIL=0
            Atnq(i)=G05DJF(df2,IFAIL)
 30         CONTINUE
      ENDIF
      IF (loi .EQ. 3) THEN
         CALL rskew(l,Atnq,lambda)
         DO 40, i=1,l
            Atnq(i)=Atnq(i)-
     +           dsqrt(dble(2.0)/pi)*(lambda/(sqrt(1+lambda*lambda)))
 40      CONTINUE
      ENDIF
      IF (loi .EQ. 4) THEN
c                          l'esperance d'une loi de Laplace est nulle
         CALL rlap(l,Atnq)
      ENDIF
      IF (loi .EQ. 5) THEN
         IFAIL=0
         Eweibu=S14AAF(DBLE(1+1/loi5k),IFAIL)/loi5b
         DO 50, i=1,l
         IFAIL=0
            Atnq(i)=G05DPF(loi5k,(loi5b)**(-loi5k),IFAIL)-Eweibu
 50         CONTINUE
      ENDIF
      IF (loi .EQ. 6) THEN
         IFAIL=0
         CALL G05FFF(loi6p,loi6q,l,Atnq,IFAIL)
         DO 60, i=1,l
            Atnq(i)=Atnq(i)-loi6p*loi6q
 60      CONTINUE
      ENDIF
      IF (loi .EQ. 7) THEN
         DO 70, i=1,l
            Atnq(i)=G05DEF(-loi7g/loi7d,1/loi7d)
     +                    -dexp(-loi7g/loi7d+0.5/(loi7d**2))
 70         CONTINUE
      ENDIF
      IF (loi .EQ. 8) THEN
         IFAIL=0
         CALL G05FEF(loi8p,loi8q,l,Atnq,IFAIL)
         DO 80, i=1,l
            Atnq(i)=Atnq(i)-loi8p/(loi8p+loi8q)
```

```
 80         CONTINUE
      ENDIF
      IF (loi .EQ. 9) THEN
         DO 90, i=1,l
            Atnq(i)=G05DAF(loi9a,loi9b)-(loi9a+loi9b)/2
 90         CONTINUE
      ENDIF
      IF (loi .EQ. 10) THEN
         DO 100, i=1,l
            Atnq(i)=G05DBF(1/loi10b)-1/loi10b
 100         CONTINUE
      ENDIF
      IF (loi .EQ. 11) THEN
         CALL rpare(loi11a,loi11k,l,Atnq)
         DO 110, i=1,l
            Atnq(i)=Atnq(i)-loi11k*loi11a/(loi11a-1)
 110         CONTINUE
      ENDIF
      IF (loi .EQ. 12) THEN
         CALL rspare(l,Atnq)
         DO 120, i=1,l
            Atnq(i)=Atnq(i)-dble(1.0)
 120         CONTINUE
      ENDIF
      IF (loi .EQ. 13) THEN
         CALL rSU(loi13g,loi13d,l,Atnq)
         DO 130, i=1,l
            Atnq(i)=Atnq(i)-EspSU
 130         CONTINUE
      ENDIF
      IF (loi .EQ. 14) THEN
c                        l'esperance d'une loi TU(l) = 0
         CALL rTU(loi14l,l,Atnq)
      ENDIF
      IF (loi .EQ. 15) THEN
         DO 140, i=1,l
c                    l'esperance d'une loi Logistic = 0
            Atnq(i)=G05DCF(a,b)
 140         CONTINUE
      ENDIF
      IF (loi .EQ. 16) THEN
c                    l'esperance d'une loi SC(p,d) = 0
         CALL rSC(loi16p,loi16d,l,Atnq)
      ENDIF
      IF (loi .EQ. 17) THEN
         CALL rLC(loi17p,loi17m,l,Atnq)
         DO 150, i=1,l
            Atnq(i)=Atnq(i) -loi17p*loi17m
 150      CONTINUE
      ENDIF
      IF (loi .EQ. 18) THEN
         CALL rSB(loi18g,loi18d,l,Atnq)
         DO 160, i=1,l
            Atnq(i)=Atnq(i)-EspSB
 160         CONTINUE
      ENDIF
      IF (loi .EQ. 19) THEN
         DO 170, i=1,l
            Atnq(i)=rS(loi19a,loi19b)-EspS
 170         CONTINUE
      ENDIF
c Etape 3 de l'algorithme
c YtMn:          Y(1) , ..., Y(Mind+n)
c YtMn(i), i=:    1            Mind+n
      DO 180, i=1,(Mind+n)
```

```
          YtMn(i)=DBLE(0.0)
 180       CONTINUE
       DO 200,i=1,(Mind+n)
c Atnq:          eps(1-q), eps(2-q), ..., eps(0), eps(1), ..., eps(n+Mind)
c Atnq(k), k=:      1         2              q      q+1           n+Mind+q
          DO 190, k=1,q
             YtMn(i)=YtMn(i)+teta(k)*Atnq(q+i-k)
 190       CONTINUE
             YtMn(i)=YtMn(i)+Atnq(q+i)
 200      CONTINUE
       RETURN
       END
```

Programme simulaARMA.f

```
c Debut-Commentaires
c Nom de la sous-routine: simARM
c Entrees:
c p (entier) ordre de la partie AR
c q (entier) ordre de la partie MA
c n (entier) longueur de la serie souhaitee
c marret (entier) rang d'arret dans la random shock method
c Mind (entier) induction period
c loi (entier) specifie la loi des erreurs ou innovations
c df1 (entier)
c df2 (entier)
c lambda (double)
c loi5b, loi5k, loi6p, loi6q
c loi7g, loi7d, loi8p, loi8q, loi9a, loi9b
c loi10b, loi10l, loi11a, loi11k, loi13g, loi13d
c loi14l, loi16p, loi16d, loi17p, loi17m
c loi18g, loi18d, loi19a, loi19b
c Wtip (vecteur, double) de longueur p qui contiendra les p donnees initiales de la
     serie
c shocks (vecteur, double) des marret+p innovations genere par shock.f
c psi (vecteur, double) de longueur marret+1 tel que defini dans Burn, calcule par
     ARMpsi.f
c phi (vecteur, double) de longueur p
c teta (vecteur, double) de longueur q
c phi2 (vecteur, double) de longueur marret
c teta2 (vecteur, double) de longueur marret
c YtpMn (vecteur, double) de longueur p+Mind+n
c Atnq (vecteur, double) de longueur n+Mind+q
c EspSU (double) : esperance de la loi SU
c EspSB (double) : esperance de la loi S
c EspS (double) : esperance de la loi S
c Sorties: void
c Description:
c Ce programme realise les etapes 2 et 3 de Simulation Algorithm 1, de l'article de
     Burn
c Cette sous-routine remplace le vecteur YtpMn en entree par p+Mind+n donnees simulees
c d'un modele ARMA(p,q) de vecteur d'innovations de loi donnee par l'entier loi ,
c les p premieres valeurs sont les p valeurs initiales de l'etape 1 de Burn,
c les Mind valeurs suivantes sont les valeurs a ecarter de la warm-up period de
     longueur Mind
c donc seules les n dernieres valeurs devront etre conservees pour la suite de la
     simulation
c elle renvoie aussi le vecteur shocks des marret+p innovations utilisees dans l'
     algorithme
c Initialisation 1 de Burn, ainsi que le vecteur Atnq des n+Mind+q random shocks At de
      l'etape 2 de Burn.
c Utilisation dans une fonction main:
c-------------------------------------------
c       PROGRAM main
c       INTEGER p, q, n, marret, Mind, loi, df1, df2
```

```
c       DOUBLE PRECISION lambda, EspSU, EspSB, EspS
c       PARAMETER(p=2,q=2,n=100,marret=200,Mind=200,loi=1,
c      +     df1=2,df2=5,lambda=2.0, EspSU=0, EspSB=0, EspS=0)
c       DOUBLE PRECISION loi5b, loi5k, loi6p, loi6q
c       DOUBLE PRECISION loi7g, loi7d, loi8p, loi8q, loi9a, loi9b
c       DOUBLE PRECISION loi10b, loi10l, loi11a, loi11k, loi13g, loi13d
c       DOUBLE PRECISION loi14l, loi16p, loi16d, loi17p, loi17m
c       DOUBLE PRECISION loi18g, loi18d, loi19a, loi19b
c       PARAMETER(loi18g=1.0,loi18d=1.0, loi19a=1.1, loi19b=0.5)
c       PARAMETER(loi16p=0.2,loi16d=5.0, loi17p=0.2,loi17m=3.0)
c       PARAMETER(loi10b=0.2,loi10l=1, loi11a=2.0,loi11k=0.5)
c       PARAMETER(loi13g=1.0,loi13d=1.0, loi14l=0.7)
c       PARAMETER(loi7g=0,loi7d=1, loi8p=2, loi8q=2, loi9a=0,loi9b=2)
c       PARAMETER(loi5b=1, loi5k=1.8, loi6p=4, loi6q=1)
c       DOUBLE PRECISION Wtip(p), shocks(marret+p), psi(marret+1)
c       DOUBLE PRECISION phi(2), teta(2), YtpMn(p+Mind+n), Atnq(n+Mind+q)
c       DOUBLE PRECISION phi2(marret), teta2(marret)
cc       Initialisation de phi et teta
c       phi(1)=0.1
c       phi(2)=0.2
c       teta(1)=0.1
c       teta(2)=0.5
c       CALL G05CBF(0)
cc       Calcul de YtpMn, shocks, Atnq
c       CALL simARM (p,q,n,marret,Mind,loi,df1,df2,lambda,loi5b, loi5k,
c      +  loi6p, loi6q, loi7g, loi7d, loi8p, loi8q, loi9a, loi9b,
c      +  loi10b, loi10l, loi11a, loi11k, loi13g, loi13d,
c      +  loi14l, loi16p, loi16d, loi17p, loi17m, loi18g, loi18d,
c      +  loi19a, loi19b,
c      +  Wtip,shocks,psi,phi,teta,phi2,teta2,YtpMn,Atnq, EspSU,
c      +  EspSB, EspS )
cc       Affichage des valeurs de YtpMn
c       WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de YtpMn'
c       WRITE(UNIT=6,FMT=*) YtpMn
cc       Affichage des valeurs de shocks
c       WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de shocks'
c       WRITE(UNIT=6,FMT=*) shocks
cc       Affichage des valeurs de Atnq
c       WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de Atnq'
c       WRITE(UNIT=6,FMT=*) Atnq
c       END
c       INCLUDE 'rskew.f'
c       INCLUDE 'rlap.f'
c       INCLUDE 'ARMpsi.f'
c       INCLUDE 'shock.f'
c       INCLUDE 'rpare.f'
c       INCLUDE 'rspare.f'
c       INCLUDE 'rSU.f'
c       INCLUDE 'rTU.f'
c       INCLUDE 'rSC.f'
c       INCLUDE 'rLC.f'
c       INCLUDE 'rSB.f'
c       INCLUDE 'rS.f'
c-------------------------------------------------
c Instructions de compilation: alias f77='fort77'
c f77 -c nom_du_fichier_contenant_la_fonction_main.f
c f77 -c simARM.f -lnag
c f77 nom_du_fichier_contenant_la_fonction_main.o simARM.o -lnag
c Fonctions exterieures appelees:
c ARMpsi.f, shock.f, rskew.f, rlap.f et G05DEF, G05DHF, G05DJF de NAG Mark16
c Auteur: Pierre Lafaye de Micheaux
c Date: 15/02/2001
c Fin-Commentaires
      SUBROUTINE simARM (p,q,n,marret,Mind,loi,df1,df2,lambda,loi5b,
     +  loi5k, loi6p, loi6q, loi7g, loi7d, loi8p, loi8q, loi9a, loi9b,
```

```
     +  loi10b, loi10l, loi11a, loi11k, loi13g, loi13d, loi14l, loi16p,
     +  loi16d, loi17p, loi17m, loi18g, loi18d, loi19a, loi19b, Wtip,
     + shocks,psi,phi,teta,phi2,teta2,YtpMn,Atnq,EspSU,EspSB,EspS)
      INTEGER l, p,q,n,marret,Mind,loi,df1,df2
      DOUBLE PRECISION Wtip(p), shocks(marret+p),psi(marret+1)
      DOUBLE PRECISION phi(2),teta(2),phi2(marret)
      DOUBLE PRECISION teta2(marret),YtpMn(p+Mind+n)
      DOUBLE PRECISION Atnq(n+Mind+q)
      DOUBLE PRECISION pi, a, b, lambda, Eweibu ,EspSU, EspSB, EspS
      DOUBLE PRECISION G05DEF, G05DHF
      DOUBLE PRECISION G05DJF, G05DPF, G05DCF, S14AAF
      DOUBLE PRECISION G05DAF, G05DBF, rS
      DOUBLE PRECISION loi5b, loi5k, loi6p, loi6q
      DOUBLE PRECISION loi7g, loi7d, loi8p, loi8q, loi9a, loi9b
      DOUBLE PRECISION loi10b, loi10l, loi11a, loi11k, loi13g, loi13d
      DOUBLE PRECISION loi14l, loi16p, loi16d, loi17p, loi17m
      DOUBLE PRECISION loi18g, loi18d, loi19a, loi19b
c Calcul des valeurs de psi
      CALL ARMpsi(marret,p,q,psi,phi,teta, phi2,   teta2)
c Calcul de shocks et Wtip
      CALL shock(p, marret, loi, Wtip, shocks, psi,
     +                  df1, df2, lambda, loi5b, loi5k, loi6p, loi6q,
     +  loi7g, loi7d, loi8p, loi8q, loi9a, loi9b,
     +  loi10b, loi10l, loi11a, loi11k, loi13g, loi13d,
     +  loi14l, loi16p, loi16d, loi17p, loi17m, loi18g, loi18d,
     +  loi19a, loi19b)
c Etape 2 de l'algorithme: on genere n+Mind+q random shocks
      l=n+Mind+q
      pi=DBLE(3.14159265358979)
      a=DBLE(0.0)
      b=DBLE(1.0)
c Ca c'est inutile puisqu'on appelle jamais simulARMA avec loi=0
c      IF (loi .EQ. 0) THEN
c            CALL G05FDF(a,b,l,Atnq)
c      DO 10, i=1,l
c            Atnq(i)=Atnq(i)-a
c 10      CONTINUE
c      ENDIF
      IF (loi .EQ. 1) THEN
         DO 20, i=1,l
            IFAIL=0
            Atnq(i)=G05DHF(df1,IFAIL)-DBLE(df1)
 20         CONTINUE
      ENDIF
      IF (loi .EQ. 2) THEN
         DO 30, i=1,l
c                                    l'esperance d'une student est nulle
         IFAIL=0
            Atnq(i)=G05DJF(df2,IFAIL)
 30         CONTINUE
      ENDIF
      IF (loi .EQ. 3) THEN
         CALL rskew(l,Atnq,lambda)
         DO 40, i=1,l
            Atnq(i)=Atnq(i) -
     +           dsqrt(dble(2.0)/pi)*(lambda/(sqrt(1+lambda*lambda)))
 40      CONTINUE
      ENDIF
      IF (loi .EQ. 4) THEN
c                        l'esperance d'une loi de Laplace est nulle
         CALL rlap(l,Atnq)
      ENDIF
      IF (loi .EQ. 5) THEN
         IFAIL=0
         Eweibu=S14AAF(DBLE(1+1/loi5k),IFAIL)/loi5b
```

```
          DO 50, i=1,l
          IFAIL=0
              Atnq(i)=G05DPF(loi5k,(loi5b)**(-loi5k),IFAIL)-Eweibu
 50           CONTINUE
       ENDIF
       IF (loi .EQ. 6) THEN
          IFAIL=0
          CALL G05FFF(loi6p,loi6q,l,Atnq,IFAIL)
          DO 60, i=1,l
              Atnq(i)=Atnq(i) -loi6p*loi6q
 60       CONTINUE
       ENDIF
       IF (loi .EQ. 7) THEN
          DO 70, i=1,l
              Atnq(i)=G05DEF(-loi7g/loi7d,1/loi7d)
     +                        -dexp(-loi7g/loi7d+0.5/(loi7d**2))
 70           CONTINUE
       ENDIF
       IF (loi .EQ. 8) THEN
          IFAIL=0
          CALL G05FEF(loi8p,loi8q,l,Atnq,IFAIL)
          DO 80, i=1,l
              Atnq(i)=Atnq(i)-loi8p/(loi8p+loi8q)
 80           CONTINUE
       ENDIF
       IF (loi .EQ. 9) THEN
          DO 90, i=1,l
              Atnq(i)=G05DAF(loi9a,loi9b)-(loi9a+loi9b)/2
 90           CONTINUE
       ENDIF
       IF (loi .EQ. 10) THEN
          DO 100, i=1,l
              Atnq(i)=G05DBF(1/loi10b)-1/loi10b
 100           CONTINUE
       ENDIF
       IF (loi .EQ. 11) THEN
          CALL rpare(loi11a,loi11k,l,Atnq)
          DO 110, i=1,l
              Atnq(i)=Atnq(i)-loi11k*loi11a/(loi11a-1)
 110          CONTINUE
       ENDIF
       IF (loi .EQ. 12) THEN
          CALL rspare(l,Atnq)
          DO 120, i=1,l
              Atnq(i)=Atnq(i)-dble(1.0)
 120          CONTINUE
       ENDIF
       IF (loi .EQ. 13) THEN
          CALL rSU(loi13g,loi13d,l,Atnq)
          DO 130, i=1,l
              Atnq(i)=Atnq(i)-EspSU
 130          CONTINUE
       ENDIF
       IF (loi .EQ. 14) THEN
c                         l'esperance d'une loi TU(l) = 0
          CALL rTU(loi14l,l,Atnq)
       ENDIF
       IF (loi .EQ. 15) THEN
          DO 140, i=1,l
c                   l'esperance d'une loi Logistic = 0
              Atnq(i)=G05DCF(a,b)
 140           CONTINUE
       ENDIF
       IF (loi .EQ. 16) THEN
c                   l'esperance d'une loi SC(p,d) = 0
```

```
         CALL rSC(loi16p,loi16d,l,Atnq)
      ENDIF
      IF (loi .EQ. 17) THEN
         CALL rLC(loi17p,loi17m,l,Atnq)
         DO 150, i=1,l
            Atnq(i)=Atnq(i) -loi17p*loi17m
 150      CONTINUE
      ENDIF
      IF (loi .EQ. 18) THEN
         CALL rSB(loi18g,loi18d,l,Atnq)
         DO 160, i=1,l
            Atnq(i)=Atnq(i)-EspSB
 160        CONTINUE
      ENDIF
      IF (loi .EQ. 19) THEN
         DO 170, i=1,l
            Atnq(i)=rS(loi19a,loi19b)-EspS
 170         CONTINUE
      ENDIF
c Etape 3 de l'algorithme
c YtpMn:        Y(1-p), Y(2-p), ..., Y(0), Y(1) , ..., Y(Mind+n)
c YtpMn(i), i=:   1       2          p     p+1         p+Mind+n
      DO 180, i=1,(p+Mind+n)
         YtpMn(i)=DBLE(0.0)
 180     CONTINUE
c Wtp:           Y(1-p), Y(2-p), ..., Y(0)
c YtpMn(i), i=:    1       2          p
      DO 190, i=1,p
         YtpMn(i)=Wtip(i)
 190     CONTINUE
      DO 220,i=(p+1),(p+Mind+n)
         DO 200, k=1,p
         YtpMn(i)=YtpMn(i)+phi(k)*YtpMn(i-k)
 200     CONTINUE
c Atnq:        eps(1-q), eps(2-q), ..., eps(0), eps(1), ..., eps(n+Mind)
c Atnq(k), k=:     1         2              q      q+1          n+Mind+q
         DO 210, k=1,q
            YtpMn(i)=YtpMn(i)+teta(k)*Atnq(q-p+i-k)
 210      CONTINUE
            YtpMn(i)=YtpMn(i)+Atnq(q-p+i)
 220      CONTINUE
      RETURN
      END
```

A.9. LES PROGRAMMES POUR LA "RANDOM SHOCK METHOD" DE BURN

Programme ARpsi.f

```
c Debut-Commentaires
c Nom de la sous-routine: ARpsi
c Entrees:
c marret (entier), longueur du vecteur psi
c psi (double), un vecteur de marret nombres
c p (entier)
c phi (double), vecteur de longueur p
c phi2(marret)
c Sorties: void
c Description:
c Cette fonction ne renvoie rien mais modifie le vecteur psi,
c en le remplacant par les valeurs de psi(i) telles que definies dans Burn
c pour i=1, ..., marret+1
c Utilisation dans une fonction main:
c-------------------------------
```

```
c       PROGRAM main
c       INTEGER marret, p
c       PARAMETER(p=2)
cc       longueur du vecteur psi, a modifier dans le programme principal
c       PARAMETER(marret=50)
cc       initialisation de psi et phi
c       DOUBLE PRECISION psi(marret+1), phi(2)
cc       initialisation de phi2 de longueur p
c       DOUBLE PRECISION phi2(marret)
cc       affectation de valeurs a phi
c       phi(1)=0.1
c       phi(2)=0.2
cc       calcul de psi, remplace psi par ARpsi(m,p,psi,phi)
c       CALL ARpsi(marret,p,psi,phi,phi2)
cc       Affichage des valeurs de psi
c       WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de psi'
c       WRITE(UNIT=6,FMT=*) psi
c       END
c-------------------------------------------
c Instructions de compilation: alias f77='fort77'
c f77 -c nom_du_fichier_contenant_la_fonction_main.f
c f77 -c ARpsi.f
c f77 nom_du_fichier_contenant_la_fonction_main.o ARpsi.o
c Fonctions exterieures appelees:
c Auteur: Pierre Lafaye de Micheaux
c Date: 15/02/2001
c Fin-Commentaires
      SUBROUTINE ARpsi(marret,p,psi,phi,phi2)
      INTEGER marret, p, j, k
      DOUBLE PRECISION psi(marret+1), phi(2)
      DOUBLE PRECISION phi2(marret)
      DO 2, j=1,marret
          phi2(j)=DBLE(0.0)
          psi(j)=DBLE(0.0)
 2        CONTINUE
       psi(1)=DBLE(1.0)
       psi(marret+1)=DBLE(0.0)
      DO 3, j=1,p
          phi2(j)=phi(j)
 3    CONTINUE
      DO 20, j=1,marret
         DO 10, k=1,j
          psi(j+1)=psi(j+1)+phi2(k)*psi(j-k+1)
 10       CONTINUE
 20       CONTINUE
      RETURN
      END
```

Programme MApsi.f

```
c Debut-Commentaires
c Nom de la sous-routine: MApsi
c Entrees:
c marret (entier), longueur du vecteur psi
c psi (double), un vecteur de marret nombres
c q (entier)
c teta (double), vecteur de longueur q
c Sorties: void
c Description:
c Cette fonction ne renvoie rien mais modifie le vecteur psi,
c en le remplacant par les valeurs de psi(i) telles que definies dans Burn
c pour i=1, ..., marret+1
c Utilisation dans une fonction main:
c-------------------------------------------
c       PROGRAM main
```

```
c       INTEGER marret, q
c       PARAMETER(q=2)
cc       longueur du vecteur psi, a modifier dans le programme principal
c       PARAMETER(marret=50)
cc       initialisation de psi et teta
c       DOUBLE PRECISION psi(marret+1), teta(q)
cc       affectation de valeurs a teta
c       teta(1)=0.1
c       teta(2)=0.5
cc       calcul de psi, remplace psi par psi(m,q,psi,teta)
c       CALL MApsi(marret,q,psi,teta)
cc       Affichage des valeurs de psi
c       WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de psi'
c       WRITE(UNIT=6,FMT=*) psi
c       END
c-------------------------------------------
c Instructions de compilation: alias f77='fort77'
c f77 -c nom_du_fichier_contenant_la_fonction_main.f
c f77 -c MApsi.f
c f77 nom_du_fichier_contenant_la_fonction_main.o MApsi.o
c Fonctions exterieures appelees:
c Auteur: Pierre Lafaye de Micheaux
c Date: 15/02/2001
c Fin-Commentaires
      SUBROUTINE MApsi(marret,q,psi,teta)
      INTEGER marret, q, j
      DOUBLE PRECISION psi(marret+1), teta(q)
      psi(1)=DBLE(1.0)
      DO 10, j=1,q
         psi(j+1)=teta(j)
 10      CONTINUE
      DO 20, j=(q+1),marret
         psi(j+1)=DBLE(0.0)
 20      CONTINUE
      RETURN
      END
```

Programme ARMpsi.f

```
c Debut-Commentaires
c Nom de la sous-routine: ARMpsi
c Entrees:
c marret (entier), longueur du vecteur psi
c psi (double), un vecteur de marret nombres
c p (entier)
c phi (double), vecteur de longueur p
c q (entier)
c teta (double) vecteur de longueur q
c phi2 (double), de longueur marret
c teta2 (double), de longueur marret
c Sorties: void
c Description:
c Cette fonction ne renvoie rien mais modifie le vecteur psi,
c en le remplacant par les valeurs de psi(i) telles que definies dans Burn
c pour i=1, ..., marret+1
c Utilisation dans une fonction main:
c-------------------------------------------
c       PROGRAM main
c       INTEGER marret, p, q
c       PARAMETER(p=2, q=2)
cc       longueur du vecteur psi, a modifier dans le programme principal
c       PARAMETER(marret=50)
cc       initialisation de psi, phi et teta
c       DOUBLE PRECISION psi(marret+1), phi(2), teta(2)
cc       initialisation de phi2 et teta2
```

```
c       DOUBLE PRECISION phi2(marret), teta2(marret)
cc       affectation de valeurs a phi et teta
c       phi(1)=0.1
c       phi(2)=0.2
c       teta(1)=0.1
c       teta(2)=0.5
cc       calcul de psi(marret,p,q,psi,phi,teta), remplace psi par psi(m,p,q,psi,phi,
     teta)
c       CALL ARMpsi(marret,p,q,psi,phi,teta,phi2,teta2)
cc       Affichage des valeurs de psi
c       WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de psi'
c       WRITE(UNIT=6,FMT=*) psi
c       END
c-------------------------------------
c Instructions de compilation: alias f77='fort77'
c f77 -c nom_du_fichier_contenant_la_fonction_main.f
c f77 -c ARMpsi.f
c f77 nom_du_fichier_contenant_la_fonction_main.o ARMpsi.o
c Fonctions exterieures appelees:
c Auteur: Pierre Lafaye de Micheaux
c Date: 15/02/2001
c Fin-Commentaires
      SUBROUTINE ARMpsi(marret,p,q,psi,phi,teta,phi2,teta2)
      INTEGER marret, p, q, j, k
      DOUBLE PRECISION psi(marret+1), phi(2), teta(2)
      DOUBLE PRECISION phi2(marret), teta2(marret)
      DO 2, j=1,marret
         phi2(j)=DBLE(0.0)
         teta2(j)=DBLE(0.0)
         psi(j)=DBLE(0.0)
 2       CONTINUE
         psi(1)=DBLE(1.0)
         psi(marret+1)=DBLE(0.0)
      DO 3, j=1,p
         phi2(j)=phi(j)
 3    CONTINUE
      DO 4, j=1,q
         teta2(j)=teta(j)
 4    CONTINUE
      DO 20, j=1,marret
         DO 10, k=1,j
         psi(j+1)=psi(j+1)+phi2(k)*psi(j-k+1)
 10      CONTINUE
         psi(j+1)=psi(j+1)+teta2(j)
 20      CONTINUE
      RETURN
      END
```

Programme shock.f

```
c Debut-Commentaires
c Nom de la sous-routine: shock
c Entrees:
c p (entier), ordre de la partie AR
c marret (entier), rang d'arret dans la random shock method
c loi (entier), specifie la loi des innovations
c Wtip (double), vecteur de longueur p qui va contenir les p donnees
c          initiales de la serie
c shocks (double), vecteur des marret+p innovations genere
c psi (double), vecteur de longueur marret+1
c          tel que defini dans Burn
c df1 (entier)
c df2 (entier)
c lambda (double)
c loi5b, loi5k, loi6p, loi6q
```

```
c loi7g, loi7d, loi8p, loi8q, loi9a, loi9b
c loi10b, loi10l, loi11a, loi11k, loi13g, loi13d
c loi14l, loi16p, loi16d, loi17p, loi17m
c loi18g, loi18d, loi19a, loi19b
c Sorties: void
c Description:
c Ce programme realise la partie Initialisation Algoritm 1, de l'article de Burn
c Cette sous-routine remplace le vecteur Wtip en entree par p donnees simulees d'un
    modele ARMA(p,q)
c de vecteur d'innovations de loi donnee par l'entier loi
c elle renvoie aussi le vecteur shocks des marret+p innovations utilisees
c Utilisation dans une fonction main:
c-----------------------------------------
c       PROGRAM main
c       INTEGER p, marret, loi, df1, df2, i
c       DOUBLE PRECISION lambda, loi5b, loi5k, loi6p, loi6q
c       DOUBLE PRECISION loi7g, loi7d, loi8p, loi8q, loi9a, loi9b
c       DOUBLE PRECISION loi10b, loi10l, loi11a, loi11k, loi13g, loi13d
c       DOUBLE PRECISION loi14l, loi16p, loi16d, loi17p, loi17m
c       DOUBLE PRECISION loi18g, loi18d, loi19a, loi19b
c       PARAMETER(loi18g=1.0,loi18d=1.0, loi19a=1.1, loi19b=0.5)
c       PARAMETER(loi16p=0.2,loi16d=5.0, loi17p=0.2,loi17m=3.0)
c       PARAMETER(loi10b=0.2,loi10l=1, loi11a=2.0,loi11k=0.5)
c       PARAMETER(loi13g=1.0,loi13d=1.0, loi14l=0.7)
c       PARAMETER(loi7g=0,loi7d=1, loi8p=2, loi8q=2, loi9a=0,loi9b=2)
c       PARAMETER(loi5b=1, loi5k=1.8, loi6p=4, loi6q=1)
c       PARAMETER(p=2,marret=200,loi=1,df1=2,df2=5,lambda=2.0)
c       DOUBLE PRECISION Wtip(p), shocks(marret+p), psi(marret+1)
cc       Les valeurs de psi
c       DO 10, i=1,marret
c       psi(i)=0.1
c 10    CONTINUE
c       CALL G05CBF(0)
cc       Calcul de shocks et Wtip
c       CALL shock (p, marret, loi, Wtip, shocks, psi,
c      +         df1, df2, loi5b, loi5k, loi6p, loi6q, lambda,
c      +         loi7g, loi7d, loi8p, loi8q, loi9a, loi9b,
c      +         loi10b, loi10l, loi11a, loi11k, loi13g, loi13d,
c      +         loi14l, loi16p, loi16d, loi17p, loi17m, loi18g, loi18d,
c      +         loi19a, loi19b)
cc       Affichage des valeurs de Wtip
c       WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de Wtip'
c       WRITE(UNIT=6,FMT=*) Wtip
cc       Affichage des valeurs de shocks
c       WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de shocks'
c       WRITE(UNIT=6,FMT=*) shocks
c       END
c       INCLUDE 'rskew.f'
c       INCLUDE 'rlap.f'
c       INCLUDE 'rpare.f'
c       INCLUDE 'rspare.f'
c       INCLUDE 'rSU.f'
c       INCLUDE 'rTU.f'
c       INCLUDE 'rSC.f'
c       INCLUDE 'rLC.f'
c       INCLUDE 'rSB.f'
c       INCLUDE 'rS.f'
c-----------------------------------------
c Instructions de compilation: alias f77='fort77'
c f77 -c nom_du_fichier_contenant_la_fonction_main.f
c f77 -c shock.f -lnag
c f77 nom_du_fichier_contenant_la_fonction_main.o shock.o -lnag
c Fonctions exterieures appelees:
c rskew.f, rlap.f, rpare.f, rspare.f, rSU.f, rTU.f, rSC.f, rLC.f, rSB.f rS.f et
c G05DHF, G05DJF, G05DEF, G05DPF, G05FFF, G05FEF, G05DAF, G05DBF, G05DCF de NAG Mark16
```

```
c Auteur: Pierre Lafaye de Micheaux
c Date: 15/02/2001
c Fin-Commentaires
      SUBROUTINE shock (p, marret, loi, Wtip, shocks, psi,
     +  df1, df2, lambda, loi5b, loi5k, loi6p, loi6q,
     +  loi7g, loi7d, loi8p, loi8q, loi9a, loi9b,
     +  loi10b, loi10l, loi11a, loi11k, loi13g, loi13d,
     +  loi14l, loi16p, loi16d, loi17p, loi17m, loi18g, loi18d,
     +  loi19a, loi19b)
      INTEGER p, marret, loi, i, k, IFAIL, df1, df2, l
      DOUBLE PRECISION Wtip(p), shocks(marret+p), psi(marret+1)
      DOUBLE PRECISION pi, a, b, lambda, loi5b, loi5k, loi6p, loi6q
      DOUBLE PRECISION loi7g, loi7d, loi8p, loi8q, loi9a, loi9b
      DOUBLE PRECISION loi10b, loi10l, loi11a, loi11k, loi13g, loi13d
      DOUBLE PRECISION loi14l, loi16p, loi16d, loi17p, loi17m
      DOUBLE PRECISION loi18g, loi18d, loi19a, loi19b
      DOUBLE PRECISION G05DHF, G05DJF, G05DEF, G05DPF, G05DCF
      DOUBLE PRECISION G05DAF, G05DBF, rS
c 1. Generate marret+p random shocks At for t=1-marret, 2-marret, ..., p
c suivant la valeur de l'entier loi
c      si loi=0 : Normale(0,sigma^2)
c      si loi=1 : Khi2 centree (df1)
c      si loi=2 : Student (df2)
c      si loi=3 : Skew-Normale(lambda)
c      si loi=4 : Laplace
c      si loi=5 : Weibull(b,k)
c      si loi=6 : Gamma(p,q)
c      si loi=7 : Log-Normale(g,d)
c      si loi=8 : Beta(p,q)
c      si loi=9 : Uniform(a,b)
c      si loi=10 : Shifted exp (l,b)
c      si loi=11 : Pareto(a,k)
c      si loi=12 : Shifted Pareto
c      si loi=13 : SU(g,d)
c      si loi=14 : TU(l)
c      si loi=15 : Logistic
c      si loi=16 : SC(p,d)
c      si loi=17 : LC(p,m)
c      si loi=18 : SB(g,d)
c      si loi=19 : S(a,b)
      l=marret+p
      pi=DBLE(3.14159265358979)
      a=DBLE(0.0)
      b=DBLE(1.0)
c     Ca c'est inutile puisqu'on n'appelle jamais shock.f avec loi=0
c     Si jamais on decide de le faire il faudra introduire sigma a la
c     place de b juste (et uniquement) en dessous
c      IF (loi .EQ. 0) THEN
c            CALL G05FDF(a,b,l,shocks)
c       ENDIF
      IF (loi .EQ. 1) THEN
         DO 10, i=1,l
            IFAIL=0
            shocks(i)=G05DHF(df1,IFAIL)-DBLE(df1)
 10         CONTINUE
      ENDIF
      IF (loi .EQ. 2) THEN
         DO 20, i=1,l
         IFAIL=0
            shocks(i)=G05DJF(df2,IFAIL)
 20         CONTINUE
      ENDIF
      IF (loi .EQ. 3) THEN
         CALL rskew(l,shocks,lambda)
         DO 30, i=1,l
```

```
            shocks(i)=shocks(i) -
     +           dsqrt(dble(2.0)/pi)*(lambda/(dsqrt(1+lambda*lambda)))
30       CONTINUE
         ENDIF
         IF (loi .EQ. 4) THEN
            CALL rlap(l,shocks)
         ENDIF
         IF (loi .EQ. 5) THEN
            DO 50, i=1,l
            IFAIL=0
               shocks(i)=G05DPF(loi5k,(loi5b)**(-loi5k),IFAIL)
50          CONTINUE
         ENDIF
         IF (loi .EQ. 6) THEN
            IFAIL=0
            CALL G05FFF(loi6p,loi6q,l,shocks,IFAIL)
         ENDIF
         IF (loi .EQ. 7) THEN
            DO 60, i=1,l
               shocks(i)=G05DEF(-loi7g/loi7d,1/loi7d)
60          CONTINUE
         ENDIF
         IF (loi .EQ. 8) THEN
            IFAIL=0
            CALL G05FEF(loi8p,loi8q,l,shocks,IFAIL)
         ENDIF
         IF (loi .EQ. 9) THEN
            DO 70, i=1,l
               shocks(i)=G05DAF(loi9a,loi9b)
70          CONTINUE
         ENDIF
         IF (loi .EQ. 10) THEN
            DO 80, i=1,l
               shocks(i)=G05DBF(1/loi10b)+loi10l
80          CONTINUE
         ENDIF
         IF (loi .EQ. 11) THEN
            CALL rpare(loi11a,loi11k,l,shocks)
         ENDIF
         IF (loi .EQ. 12) THEN
            CALL rspare(l,shocks)
         ENDIF
         IF (loi .EQ. 13) THEN
            CALL rSU(loi13g,loi13d,l,shocks)
         ENDIF
         IF (loi .EQ. 14) THEN
            CALL rTU(loi14l,l,shocks)
         ENDIF
         IF (loi .EQ. 15) THEN
            DO 90, i=1,l
               shocks(i)=G05DCF(a,b)
90          CONTINUE
         ENDIF
         IF (loi .EQ. 16) THEN
            CALL rSC(loi16p,loi16d,l,shocks)
         ENDIF
         IF (loi .EQ. 17) THEN
            CALL rLC(loi17p,loi17m,l,shocks)
         ENDIF
         IF (loi .EQ. 18) THEN
            CALL rSB(loi18g,loi18d,l,shocks)
         ENDIF
         IF (loi .EQ. 19) THEN
            DO 100, i=1,l
               shocks(i)=rS(loi19a,loi19b)
```

```
 100          CONTINUE
      ENDIF
c 2. Generate p series values Wtip pour t=1,2, ..., p en utilisant
c le MA(marret) model et les shocks de l'etape 1
c A(t) <--> shocks(t+marret)
c psi(k) <--> psi(k+1)
      DO 570, i=1,p
         Wtip(i)=DBLE(0.0)
         DO 560, k=1,(marret+1)
            Wtip(i)=Wtip(i)+psi(k)*shocks(marret+i-k+1)
 560          CONTINUE
 570       CONTINUE
      RETURN
      END
```

A.10. LES PROGRAMMES POUR LA SIMULATION DE DIFFÉRENTES LOIS

Programme dlap.f

```
c Debut-Commentaires
c Nom de la sous-routine: dlap
c Entrees:
c n (entier), longueur du vecteur x
c x (double), un vecteur de n nombres
c (x est le vecteur des points auxquels on veut calculer la densite).
c Sorties: void
c Description:
c Cette fonction ne renvoie rien mais modifie le vecteur x,
c en le remplacant par le vecteur dlap(x),  densite
c d'une loi de Laplace aux points du vecteur x.
c Utilisation dans une fonction main:
c-------------------------------------
c       PROGRAM main
c       INTEGER n, i
cc       longueur du vecteur x, a modifier dans le programme principal
c       PARAMETER(n=3)
cc       initialisation de x, y et xavant
c       DOUBLE PRECISION x(n), y(n), xavant(n)
cc       affectation de valeurs a x
c       x(1)=1
c       x(2)=2
c       x(3)=3
cc       On sauvegarde les valeurs de x dans xavant
c       DO 10, i=1,n
c       xavant(i)=x(i)
c 10    CONTINUE
cc       calcul de dlap(x), remplace x par dlap(x)
c       CALL dlap(n,x)
cc       affecte a y le resultat
c       DO 20, i=1,n
c       y(i)=x(i)
c 20    CONTINUE
cc       remet les bonnes valeurs dans x
cc       affecte a y le resultat
c       DO 30, i=1,n
c       x(i)=xavant(i)
c 30    CONTINUE
cc       Affichage des valeurs de y=dlap(x)
c       WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de y'
c       WRITE(UNIT=6,FMT=*) y
cc       Affichage des valeurs de x
c       WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de x'
c       WRITE(UNIT=6,FMT=*) x
```

```
c       END
c-----------------------------------------------
c Instructions de compilation: alias f77='fort77'
c f77 -c nom_du_fichier_contenant_la_fonction_main.f
c f77 -c dlap.f
c f77 nom_du_fichier_contenant_la_fonction_main.o dlap.o
c Fonctions exterieures appelees:
c Auteur: Pierre Lafaye de Micheaux
c Date: 15/02/2001
c Fin-Commentaires
      SUBROUTINE dlap (n,x)
      INTEGER n,i
      DOUBLE PRECISION x(n)
      DO 10, i=1,n
         x(i)=DEXP(-ABS(x(i)))/(DBLE(2.0))
 10      CONTINUE
      RETURN
      END
```

Programme dnorm.f

```
c Debut-Commentaires
c Nom de la sous-routine: dnorm
c Entrees:
c n (entier), longueur du vecteur x
c x (double), un vecteur de n nombres
c (x est le vecteur des points auxquels on veut calculer la densite).
c Sorties: void
c Description:
c Cette fonction ne renvoie rien mais modifie le vecteur x,
c en le remplacant par le vecteur dnorm(x),  densite
c d'une loi de Normale aux points du vecteur x.
c Utilisation dans une fonction main:
c-----------------------------------------------
c       PROGRAM main
c       INTEGER n, i
cc       longueur du vecteur x, a modifier dans le programme principal
c       PARAMETER(n=3)
cc       initialisation de x, y et xavant
c       DOUBLE PRECISION x(n), y(n), xavant(n)
cc       affectation de valeurs a x
c       x(1)=1
c       x(2)=2
c       x(3)=3
cc       On sauvegarde les valeurs de x dans xavant
c       DO 10, i=1,n
c       xavant(i)=x(i)
c 10    CONTINUE
cc       calcul de dnorm(x), remplace x par dnorm(x)
c       CALL dnorm(n,x)
cc       affecte a y le resultat
c       DO 20, i=1,n
c       y(i)=x(i)
c 20    CONTINUE
cc       remet les bonnes valeurs dans x
cc       affecte a y le resultat
c       DO 30, i=1,n
c       x(i)=xavant(i)
c 30    CONTINUE
cc       Affichage des valeurs de y=dlap(x)
c       WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de y'
c       WRITE(UNIT=6,FMT=*) y
cc       Affichage des valeurs de x
c       WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de x'
c       WRITE(UNIT=6,FMT=*) x
```

```
c       END
c-----------------------------------------------
c Instructions de compilation: alias f77='fort77'
c f77 -c nom_du_fichier_contenant_la_fonction_main.f
c f77 -c dnorm.f
c f77 nom_du_fichier_contenant_la_fonction_main.o dnorm.o
c Fonctions exterieures appelees:
c Auteur: Pierre Lafaye de Micheaux
c Date: 15/02/2001
c Fin-Commentaires
      SUBROUTINE dnorm (n,x)
      INTEGER n,i
      DOUBLE PRECISION PI
      PARAMETER (PI = 3.14159265358979)
      DOUBLE PRECISION x(n)
      DO 10, i=1,n
         x(i)=(DEXP((-(x(i)**2))/DBLE(2.0)))/DSQRT(PI)
 10      CONTINUE
      RETURN
      END
```

Programme dskew.f

```
c Debut-Commentaires
c Nom de la sous-routine: dskew
c Entrees:
c n (entier), longueur du vecteur x
c lambda (lambda)
c x (double), un vecteur de n nombres
c (x est le vecteur des points auxquels on veut calculer la densite).
c Sorties: void
c Description:
c Cette fonction ne renvoie rien mais modifie le vecteur x,
c en le remplacant par le vecteur dskew(x),  densite
c d'une loi de Skew-Normale(lambda) aux points du vecteur x.
c Utilisation dans une fonction main:
c-----------------------------------------------
c       PROGRAM main
c       INTEGER n, i
c       DOUBLE PRECISION lambda
cc       longueur du vecteur x, a modifier dans le programme principal
c       PARAMETER(n=3)
cc       parametre de la loi Skew-Normale a modifier dans le programme principal
c       PARAMETER (lambda=5.0)
cc       initialisation de x, y et xavant
c       DOUBLE PRECISION x(n), y(n), xavant(n)
cc       affectation de valeurs a x
c       x(1)=1
c       x(2)=2
c       x(3)=3
cc       On sauvegarde les valeurs de x dans xavant
c       DO 10, i=1,n
c       xavant(i)=x(i)
c 10    CONTINUE
cc       calcul de dskew(x,lambda), remplace x par dskew(x,lambda)
c       CALL dskew(n,lambda,x)
cc       affecte a y le resultat
c       DO 20, i=1,n
c       y(i)=x(i)
c 20    CONTINUE
cc       remet les bonnes valeurs dans x
cc       affecte a y le resultat
c       DO 30, i=1,n
c       x(i)=xavant(i)
c 30    CONTINUE
```

```
cc        Affichage des valeurs de y=dskew(x,lambda)
c        WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de y'
c        WRITE(UNIT=6,FMT=*) y
cc        Affichage des valeurs de x
c        WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de x'
c        WRITE(UNIT=6,FMT=*) x
c        END
c-------------------------------------------
c Instructions de compilation: alias f77='fort77'
c f77 -c nom_du_fichier_contenant_la_fonction_main.f -lnag
c f77 -c dskew.f -lnag
c f77 nom_du_fichier_contenant_la_fonction_main.o dskew.o -lnag
c Fonctions exterieures appelees:
c S15ABF NAG Mark16 routine
c Auteur: Pierre Lafaye de Micheaux
c Date: 15/02/2001
c Fin-Commentaires
      SUBROUTINE dskew (n,lambda,x)
      INTEGER n,i, IFAIL
      DOUBLE PRECISION PI
      PARAMETER (PI = 3.14159265358979)
      DOUBLE PRECISION lambda,  x(n), z1, z2
      DOUBLE PRECISION  S15ABF
      EXTERNAL S15ABF
      DO 10, i=1,n
         z1=DBLE(2.0)*((DEXP((-(x(i)**2))/DBLE(2.0)))/
     +          DSQRT(DBLE(2.0)*PI))
         IFAIL=0
         z2=lambda*x(i)
         S15ABF(z2,IFAIL)
         x(i)=z1*z2
 10      CONTINUE
      RETURN
      END
```

Programme pnorm.f

```
c Debut-Commentaires
c Nom de la sous-routine: pnorm
c Entrees:
c n (enier), longueur du vecteur x
c x (double), vecteur des points
c auxquels on veut calculer la probabilite
c mu (double), moyenne
c sigma (double), ecart-type
c Sorties: void
c Description:
c Cette fonction ne renvoie rien mais modifie le vecteur x,
c en le remplacant par le vecteur pnorm_bis(x,mu,sigma) des probabilites
c d'une loi Normale(mu,sigma) aux points du vecteur x.
c Utilisation dans une fonction main:
c-------------------------------------------
c        PROGRAM main
c        INTEGER n, i
c        DOUBLE PRECISION mu, sigma
cc         longueur du vecteur x, a modifier dans le programme principal
c        PARAMETER(n=3)
cc         parametre mu et sigma a changer
c        PARAMETER (mu=0,sigma=1)
cc         initialisation de x, y et xavant
c        DOUBLE PRECISION x(n), y(n), xavant(n)
cc         affectation de valeurs a x
c        x(1)=1
c        x(2)=2
c        x(3)=3
```

```
cc        On sauvegarde les valeurs de x dans xavant
c       DO 10, i=1,n
c       xavant(i)=x(i)
c 10    CONTINUE
cc        calcul de pnorm(x,mu,sigma), remplace x par pnorm(x,mu,sigma)
c       CALL pnorm(n,x,mu,sigma)
cc        affecte a y le resultat
c       DO 20, i=1,n
c       y(i)=x(i)
c 20    CONTINUE
cc        remet les bonnes valeurs dans x
c       DO 30, i=1,n
c       x(i)=xavant(i)
c 30    CONTINUE
cc        Affichage des valeurs de y=pnorm(x,mu,sigma)
c       WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de y'
c       WRITE(UNIT=6,FMT=*) y
cc        Affichage des valeurs de x
c       WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de x'
c       WRITE(UNIT=6,FMT=*) x
c       END
c-------------------------------------------
c Instructions de compilation: alias f77='fort77'
c f77 -c nom_du_fichier_contenant_la_fonction_main.f -lnag
c f77 -c pnorm.f -lnag
c f77 nom_du_fichier_contenant_la_fonction_main.o pnorm.o -lnag
c Fonctions exterieures appelees:
c S15ABF NAG Mark16 routine
c Auteur: Pierre Lafaye de Micheaux
c Date: 15/02/2001
c Fin-Commentaires
      SUBROUTINE pnorm (n,x,mu,sigma)
      INTEGER n,i, IFAIL
      DOUBLE PRECISION x(n), mu, sigma, z
      DOUBLE PRECISION  S15ABF
      EXTERNAL S15ABF
      DO 10, i=1,n
         z=(x(i)-mu)/sigma
         IFAIL=0
         x(i)=S15ABF(z,IFAIL)
 10      CONTINUE
      RETURN
      END
```

Programme qnorm.f

```
c Debut-Commentaires
c Nom de la sous-routine: qnorm
c Entrees:
c n (entier), longueur du vecteur x
c x (double), vecteur des points
c auxquels on veut calculer la probabilite
c mu (double), moyenne
c sigma (double), ecart-type
c Sorties: void
c Description:
c Cette fonction ne renvoie rien mais modifie le vecteur x,
c en le remplacant par le vecteur qnorm(x,mu,sigma) des quantiles
c d'une loi Normale(mu,sigma^2) aux points du vecteur x.
c Utilisation dans une fonction main:
c-------------------------------------------
c       PROGRAM main
c       INTEGER n, i
c       DOUBLE PRECISION mu, sigma
cc        longueur du vecteur x, a modifier dans le programme principal
```

```
c       PARAMETER(n=3)
cc        parametre mu et sigma a changer
c       PARAMETER (mu=0,sigma=1)
cc        initialisation de x, y et xavant
c       DOUBLE PRECISION x(n), y(n), xavant(n)
cc        affectation de valeurs a x
c       x(1)=0.1
c       x(2)=0.2
c       x(3)=0.3
cc        On sauvegarde les valeurs de x dans xavant
c       DO 10, i=1,n
c       xavant(i)=x(i)
c 10    CONTINUE
cc        calcul de qnorm(x,mu,sigma), remplace x par qnorm(x,mu,sigma)
c       CALL qnorm(n,x,mu,sigma)
cc        affecte a y le resultat
c       DO 20, i=1,n
c       y(i)=x(i)
c 20    CONTINUE
cc        remet les bonnes valeurs dans x
c       DO 30, i=1,n
c       x(i)=xavant(i)
c 30    CONTINUE
cc        Affichage des valeurs de y=pnorm(x,mu,sigma)
c       WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de y'
c       WRITE(UNIT=6,FMT=*) y
cc        Affichage des valeurs de x
c       WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de x'
c       WRITE(UNIT=6,FMT=*) x
c       END
c-------------------------------------
c Instructions de compilation: alias f77='fort77'
c f77 -c nom_du_fichier_contenant_la_fonction_main.f -lnag
c f77 -c qnorm.f -lnag
c f77 nom_du_fichier_contenant_la_fonction_main.o qnorm.o -lnag
c Fonctions exterieures appelees:
c G01FAF NAG Mark16 routine
c Auteur: Pierre Lafaye de Micheaux
c Date: 15/02/2001
c Fin-Commentaires
      SUBROUTINE qnorm (n,x,mu,sigma)
      INTEGER n,i, IFAIL
      CHARACTER*1  TAIL
      DOUBLE PRECISION x(n), mu, sigma
      DOUBLE PRECISION  G01FAF
      EXTERNAL G01FAF
      DO 10, i=1,n
      IFAIL=0
      TAIL='L'
         x(i)=sigma*G01FAF(TAIL,x(i),IFAIL)+mu
 10      CONTINUE
      RETURN
      END
```

Programme rlap.f

```
c Debut-Commentaires
c Nom de la sous-routine: rlap
c Entrees:
c n (entier), longueur du vecteur x
c x (double), un vecteur de n nombres
c Sorties: void
c Description:
c Cette fonction ne renvoie rien mais modifie le vecteur x,
c en le remplacant par le vecteur rlap(x); c'est a dire
```

```
c simule un echantillon de taille n
c provenant d'une loi de Laplace
c Utilisation dans une fonction main:
c-----------------------------------------
c       PROGRAM main
c       INTEGER n
cc       longueur du vecteur x, a modifier dans le programme principal
c       PARAMETER(n=3)
cc       initialisation de x
c       DOUBLE PRECISION x(n)
c       CALL G05CBF(0)
cc       calcul de rlap(x), remplace x par rlap(x)
c       CALL rlap(n,x)
cc       Affichage des valeurs de rlap(x)
c       WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de rlap(x)'
c       WRITE(UNIT=6,FMT=*) x
c       END
c       INCLUDE 'rlap.f'
c-----------------------------------------
c Instructions de compilation: alias f77='fort77'
c f77 -c nom_du_fichier_contenant_la_fonction_main.f -lnag
c f77 -c rlap.f -lnag
c f77 nom_du_fichier_contenant_la_fonction_main.o rlap.o -lnag
c Fonctions exterieures appelees:
c G05DAF de la librairie NAG Mark16
c Auteur: Pierre Lafaye de Micheaux
c Date: 15/02/2001
c Fin-Commentaires
      SUBROUTINE rlap (n,x)
      INTEGER n,i
      DOUBLE PRECISION x(n), u, G05DAF, a, b
      EXTERNAL G05DAF
      a=DBLE(0.0)
      b=DBLE(1.0)
      DO 10, i=1,n
         u=G05DAF(a,b)
         IF (u .GT. DBLE(0.5)) THEN
            x(i)=-DLOG(2*(1-u))
            ELSE
               x(i)=DLOG(2*u)
         ENDIF
 10      CONTINUE
      RETURN
      END
```

Programme rLC.f

```
c Debut-Commentaires
c Nom de la sous-routine: rLC
c Entrees:
c p (double), parametre de la SC
c m (double), parametre de la SC
c n (entier), longueur du vecteur x
c x (double), un vecteur de n nombres
c Sorties: void
c Description:
c Cette fonction ne renvoie rien mais modifie le vecteur x,
c en le remplacant par le vecteur rLC(x); c'est a dire
c simule un echantillon de taille n
c provenant d'une loi de LC(p,m)
c Utilisation dans une fonction main:
c-----------------------------------------
c       PROGRAM main
c       INTEGER n
cc       longueur du vecteur x, a modifier dans le programme principal
```

```
c       PARAMETER(n=3)
cc       valeurs donnees aux parametres
c       DOUBLE PRECISION p, m
c       PARAMETER(p=0.2, m=3.0)
cc       initialisation de x
c       DOUBLE PRECISION x(n)
c       CALL G05CBF(0)
cc       calcul de rLC(x), remplace x par rLC(x)
c       CALL rLC(p,m,n,x)
cc       Affichage des valeurs de rLC(x)
c       WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de rLC(x)'
c       WRITE(UNIT=6,FMT=*) x
c       END
c       INCLUDE 'rLC.f'
c-------------------------------------------
c Instructions de compilation: alias f77='fort77'
c f77 -c nom_du_fichier_contenant_la_fonction_main.f -lnag
c f77 -c rLC.f -lnag
c f77 nom_du_fichier_contenant_la_fonction_main.o rLC.o -lnag
c Fonctions exterieures appelees:
c G05FAF, G05DDF de la librairie NAG Mark16
c Auteur: Pierre Lafaye de Micheaux
c Date: 15/02/2001
c Fin-Commentaires
      SUBROUTINE rLC (p,m,n,x)
      INTEGER n,i
      DOUBLE PRECISION x(n), p, m, a, b, G05DDF
      EXTERNAL G05DDF
      a=DBLE(0.0)
      b=DBLE(1.0)
      CALL G05FAF(a,b,n,x)
      DO 10, i=1,n
         IF (x(i) .LT. p) THEN
            x(i)=G05DDF(m,b)
         ELSE
            x(i)=G05DDF(a,b)
         ENDIF
 10      CONTINUE
      RETURN
      END
```

Programme rpare.f

```
c Debut-Commentaires
c Nom de la sous-routine: rpare
c Entrees:
c a (double), parametre de la Pareto
c k (double), parametre de la Pareto
c n (entier), longueur du vecteur x
c x (double), un vecteur de n nombres
c Sorties: void
c Description:
c Cette fonction ne renvoie rien mais modifie le vecteur x,
c en le remplacant par le vecteur rpare(x); c'est a dire
c simule un echantillon de taille n
c provenant d'une loi de Pareto(a,k)
c Utilisation dans une fonction main:
c-------------------------------------------
c       PROGRAM main
c       INTEGER n
cc       longueur du vecteur x, a modifier dans le programme principal
c       PARAMETER(n=3)
cc       valeurs donnees aux parametres
c       DOUBLE PRECISION a, k
c       PARAMETER(a=1.8, k=1.2)
```

```
cc        initialisation de x
c        DOUBLE PRECISION x(n)
c        CALL G05CBF(0)
cc        calcul de rpare(x), remplace x par rpare(x)
c        CALL rpare(a,k,n,x)
cc        Affichage des valeurs de rpare(x)
c        WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de rpare(x)'
c        WRITE(UNIT=6,FMT=*) x
c        END
c-------------------------------------------
c Instructions de compilation: alias f77='fort77'
c f77 -c nom_du_fichier_contenant_la_fonction_main.f -lnag
c f77 -c rlap.f -lnag
c f77 nom_du_fichier_contenant_la_fonction_main.o rpare.o -lnag
c Fonctions exterieures appelees:
c G05FAF de la librairie NAG Mark16
c Auteur: Pierre Lafaye de Micheaux
c Date: 15/02/2001
c Fin-Commentaires
      SUBROUTINE rpare (a,k,n,x)
      INTEGER n,i
      DOUBLE PRECISION x(n), a, k, a1, b1
      a1=DBLE(0.0)
      b1=DBLE(1.0)
      CALL G05FAF(a1,b1,n,x)
      DO 10, i=1,n
         x(i)=k/((1-x(i))**(DBLE(1.0)/a))
 10      CONTINUE
      RETURN
      END
```

Programme rSB.f

```
c Debut-Commentaires
c Nom de la sous-routine: rSB
c Entrees:
c g (double), parametre de rSB
c d (double), parametre de rSB
c n (entier), longueur du vecteur x
c x (double), un vecteur de n nombres
c Sorties: void
c Description:
c Cette fonction ne renvoie rien mais modifie le vecteur x,
c en le remplacant par le vecteur rSB(x); c'est a dire
c simule un echantillon de taille n
c provenant d'une loi de SB(g,d)
c Utilisation dans une fonction main:
c-------------------------------------------
c        PROGRAM main
c        INTEGER n
cc        longueur du vecteur x, a modifier dans le programme principal
c        PARAMETER(n=3)
cc        Parametres de rSB
c        DOUBLE PRECISION g, d
c        PARAMETER(g=1.0,d=1.0)
cc        initialisation de x
c        DOUBLE PRECISION x(n)
c        CALL G05CBF(0)
cc        calcul de rSB(x), remplace x par rSB(x)
c        CALL rSB(g,d,n,x)
cc        Affichage des valeurs de rSB(x)
c        WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de rSB(x)'
c        WRITE(UNIT=6,FMT=*) x
c        END
c-------------------------------------------
```

```
c Instructions de compilation: alias f77='fort77'
c f77 -c nom_du_fichier_contenant_la_fonction_main.f -lnag
c f77 -c rSB.f -lnag
c f77 nom_du_fichier_contenant_la_fonction_main.o rSB.o -lnag
c Fonctions exterieures appelees:
c G05FDF de la librairie NAG Mark16
c Auteur: Pierre Lafaye de Micheaux
c Date: 15/02/2001
c Fin-Commentaires
      SUBROUTINE rSB (g,d,n,x)
      INTEGER n,i
      DOUBLE PRECISION x(n), a, b, g, d
      a=DBLE(0.0)
      b=DBLE(1.0)
      CALL G05FDF(a,b,n,x)
      DO 10, i=1,n
         x(i)=DBLE(1.0)/(1+dexp((x(i)-g)/d))
 10      CONTINUE
      RETURN
      END
```

Programme rSC.f

```
c Debut-Commentaires
c Nom de la sous-routine: rSC
c Entrees:
c p (double), parametre de la SC
c d (double), parametre de la SC
c n (entier), longueur du vecteur x
c x (double), un vecteur de n nombres
c Sorties: void
c Description:
c Cette fonction ne renvoie rien mais modifie le vecteur x,
c en le remplacant par le vecteur rSC(x); c'est a dire
c simule un echantillon de taille n
c provenant d'une loi de SC(p,d)
c Utilisation dans une fonction main:
c-----------------------------------------
c       PROGRAM main
c       INTEGER n
cc       longueur du vecteur x, a modifier dans le programme principal
c       PARAMETER(n=3)
cc       valeurs donnees aux parametres
c       DOUBLE PRECISION p, d
c       PARAMETER(p=0.2, d=5.0)
cc       initialisation de x
c       DOUBLE PRECISION x(n)
c       CALL G05CBF(0)
cc       calcul de rSC(x), remplace x par rSC(x)
c       CALL rSC(p,d,n,x)
cc       Affichage des valeurs de rSC(x)
c       WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de rSC(x)'
c       WRITE(UNIT=6,FMT=*) x
c       END
c-----------------------------------------
c Instructions de compilation: alias f77='fort77'
c f77 -c nom_du_fichier_contenant_la_fonction_main.f -lnag
c f77 -c rSC.f -lnag
c f77 nom_du_fichier_contenant_la_fonction_main.o rSC.o -lnag
c Fonctions exterieures appelees:
c G05FAF, G05DDF de la librairie NAG Mark16
c Auteur: Pierre Lafaye de Micheaux
c Date: 15/02/2001
c Fin-Commentaires
      SUBROUTINE rSC (p,d,n,x)
```

```
      INTEGER n,i
      DOUBLE PRECISION x(n), p, d, a, b, G05DDF
      EXTERNAL G05DDF
      a=DBLE(0.0)
      b=DBLE(1.0)
      CALL G05FAF(a,b,n,x)
      DO 10, i=1,n
         IF (x(i) .LT. p) THEN
            x(i)=G05DDF(a,d)
         ELSE
            x(i)=G05DDF(a,b)
         ENDIF
10       CONTINUE
      RETURN
      END
```

Programme rS.f

```
c Debut-Commentaires
c Nom de la fonction: rS
c Entrees:
c a (double), parametre de rS
c b (double), parametre de rS
c Sorties: void
c Description:
c Cette fonction renvoie simule une valeur d'une loi S(a,b)
c Utilisation dans une fonction main:
c-----------------------------------------
c       PROGRAM main
c       INTEGER i
c       DOUBLE PRECISION rS, res
cc       Parametres de rS
c       DOUBLE PRECISION a, b
c       PARAMETER(a=2.0,b=0.5)
c       CALL G05CBF(0)
c       DO 10, i=1,10
cc       Simulation d'une valeur de S(a,b)
c       res=rS(a,b)
cc       Affichage des valeurs de rS(x)
c       WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de rS(x)'
c       WRITE(UNIT=6,FMT=*) res
c 10    CONTINUE
c       END
c-----------------------------------------
c Instructions de compilation: alias f77='fort77'
c f77 -c nom_du_fichier_contenant_la_fonction_main.f -lnag
c f77 -c rS.f -lnag
c f77 nom_du_fichier_contenant_la_fonction_main.o rS.o -lnag
c Fonctions exterieures appelees:
c G05FAF de la librairie NAG Mark16
c Auteur: Pierre Lafaye de Micheaux
c Date: 15/02/2001
c Fin-Commentaires
      DOUBLE PRECISION FUNCTION rS(a,b)
      DOUBLE PRECISION x(2), x1, x2, x3, a1, b1, a, b, PI
      PARAMETER (PI = 3.14159265358979)
      a1=DBLE(0.0)
      b1=DBLE(1.0)
      CALL G05FAF(a1,b1,2,x)
      x1=dsin(a*(x(1)*PI-0.5*PI)+0.5*PI*b*(2-a))
      x2=(dcos(x(1)*PI-0.5*PI))**(DBLE(1.0)/a)
      x3=((dcos((x(1)*PI-0.5*PI)*(1-a)-0.5*PI*b*(2-a)))/(-dlog(x(2))
     +     ))**((1-a)/a)
      rS=(x1/x2)*x3
      RETURN
```

```
      END
```

Programme rskew.f

```
c Debut-Commentaires
c Nom de la sous-routine: rskew
c Entrees:
c n (entier), longueur du vecteur x
c x (double), un vecteur de n nombres
c lambda (double)
c Sorties: void
c Description:
c Cette fonction ne renvoie rien mais modifie le vecteur x,
c en le remplacant par le vecteur rskew(x,lambda); c'est a dire
c simule un echantillon de taille n
c provenant d'une loi skew-normale(lambda)
c Utilisation dans une fonction main:
c-------------------------------------------------
c       PROGRAM main
c       INTEGER n
cc       longueur du vecteur x, a modifier dans le programme principal
c       PARAMETER(n=3)
cc       initialisation de x
c       DOUBLE PRECISION x(n), lambda
c       PARAMETER(lambda=1.0)
c       CALL G05CBF(0)
cc       calcul de rskew(x,lambda), remplace x par rskew(x,lambda)
c       CALL rskew(n,x,lambda)
cc       Affichage des valeurs de rskew(x,lambda)
c       WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de rskew(x,lambda)'
c       WRITE(UNIT=6,FMT=*) x
c       END
c-------------------------------------------------
c Instructions de compilation: alias f77='fort77'
c f77 -c nom_du_fichier_contenant_la_fonction_main.f -lnag
c f77 -c rskew.f -lnag
c f77 nom_du_fichier_contenant_la_fonction_main.o rskew.o -lnag
c Fonctions exterieures appelees:
c G05DDF de la librairie NAG Mark16
c Auteur: Pierre Lafaye de Micheaux
c Date: 15/02/2001
c Fin-Commentaires
      SUBROUTINE rskew (n,x,lambda)
      INTEGER n,i
      DOUBLE PRECISION x(n), u, v, a, b, G05DDF, lambda
      EXTERNAL G05DDF
      a=DBLE(0.0)
      b=DBLE(1.0)
      DO 10, i=1,n
         u=G05DDF(a,b)
         v=G05DDF(a,b)
         IF ((lambda*u) .GT. v) THEN
            x(i)=u
            ELSE
               x(i)=-u
         ENDIF
 10      CONTINUE
      RETURN
      END
```

Programme rspare.f

```
c Debut-Commentaires
c Nom de la sous-routine: rspare
c Entrees:
c n (entier), longueur du vecteur x
```

```
c x (double), un vecteur de n nombres
c Sorties: void
c Description:
c Cette fonction ne renvoie rien mais modifie le vecteur x,
c en le remplacant par le vecteur rspare(x); c'est a dire
c simule un echantillon de taille n
c provenant d'une loi de Shifted Pareto
c Utilisation dans une fonction main:
c-------------------------------------------
c       PROGRAM main
c       INTEGER n
cc       longueur du vecteur x, a modifier dans le programme principal
c       PARAMETER(n=3)
cc       initialisation de x
c       DOUBLE PRECISION x(n)
c       CALL G05CBF(0)
cc       calcul de rspare(x), remplace x par rspare(x)
c       CALL rspare(n,x)
cc       Affichage des valeurs de rspare(x)
c       WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de rspare(x)'
c       WRITE(UNIT=6,FMT=*) x
c       END
c-------------------------------------------
c Instructions de compilation: alias f77='fort77'
c f77 -c nom_du_fichier_contenant_la_fonction_main.f -lnag
c f77 -c rlap.f -lnag
c f77 nom_du_fichier_contenant_la_fonction_main.o rspare.o -lnag
c Fonctions exterieures appelees:
c G05FAF de la librairie NAG Mark16
c Auteur: Pierre Lafaye de Micheaux
c Date: 15/02/2001
c Fin-Commentaires
      SUBROUTINE rspare (n,x)
      INTEGER n,i
      DOUBLE PRECISION x(n), a, b
      a=DBLE(0.0)
      b=DBLE(1.0)
      CALL G05FAF(a,b,n,x)
      DO 10, i=1,n
         x(i)=DBLE(1.0)/((1-x(i))**0.5)-1
 10      CONTINUE
      RETURN
      END
```

Programme rSU.f

```
c Debut-Commentaires
c Nom de la sous-routine: rSU
c Entrees:
c g (double), parametre de rSU
c d (double), parametre de rSU
c n (entier), longueur du vecteur x
c x (double), un vecteur de n nombres
c Sorties: void
c Description:
c Cette fonction ne renvoie rien mais modifie le vecteur x,
c en le remplacant par le vecteur rSU(x); c'est a dire
c simule un echantillon de taille n
c provenant d'une loi de SU(g,d)
c Utilisation dans une fonction main:
c-------------------------------------------
c       PROGRAM main
c       INTEGER n
cc       longueur du vecteur x, a modifier dans le programme principal
c       PARAMETER(n=3)
```

```
cc        Parametres de rSU
c        DOUBLE PRECISION g, d
c        PARAMETER(g=0.0,d=1.0)
cc        initialisation de x
c        DOUBLE PRECISION x(n)
c        CALL G05CBF(0)
cc        calcul de rSU(x), remplace x par rSU(x)
c        CALL rSU(g,d,n,x)
cc        Affichage des valeurs de rSU(x)
c        WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de rSU(x)'
c        WRITE(UNIT=6,FMT=*) x
c        END
c-------------------------------------------
c Instructions de compilation: alias f77='fort77'
c f77 -c nom_du_fichier_contenant_la_fonction_main.f -lnag
c f77 -c rlap.f -lnag
c f77 nom_du_fichier_contenant_la_fonction_main.o rSU.o -lnag
c Fonctions exterieures appelees:
c G05FDF de la librairie NAG Mark16
c Auteur: Pierre Lafaye de Micheaux
c Date: 15/02/2001
c Fin-Commentaires
      SUBROUTINE rSU (g,d,n,x)
      INTEGER n,i
      DOUBLE PRECISION x(n), a, b, g, d
      a=DBLE(0.0)
      b=DBLE(1.0)
      CALL G05FDF(a,b,n,x)
      DO 10, i=1,n
         x(i)=dsinh((x(i)-g)/d)
 10      CONTINUE
      RETURN
      END
```

Programme rTU.f

```
c Debut-Commentaires
c Nom de la sous-routine: rTU
c Entrees:
c l (double), parametre de rTU
c n (entier), longueur du vecteur x
c x (double), un vecteur de n nombres
c Sorties: void
c Description:
c Cette fonction ne renvoie rien mais modifie le vecteur x,
c en le remplacant par le vecteur rTU(x); c'est a dire
c simule un echantillon de taille n
c provenant d'une loi de TU(l)
c Utilisation dans une fonction main:
c-------------------------------------------
c        PROGRAM main
c        INTEGER n
cc        longueur du vecteur x, a modifier dans le programme principal
c        PARAMETER(n=3)
cc        Parametre de rTU
c        DOUBLE PRECISION l
c        PARAMETER(l=1.5)
cc        initialisation de x
c        DOUBLE PRECISION x(n)
c        CALL G05CBF(0)
cc        calcul de rTU(x), remplace x par rTU(x)
c        CALL rTU(l,n,x)
cc        Affichage des valeurs de rTU(x)
c        WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de rTU(x)'
c        WRITE(UNIT=6,FMT=*) x
```

```
c       END
c-------------------------------------------
c Instructions de compilation: alias f77='fort77'
c f77 -c nom_du_fichier_contenant_la_fonction_main.f -lnag
c f77 -c rlap.f -lnag
c f77 nom_du_fichier_contenant_la_fonction_main.o rTU.o -lnag
c Fonctions exterieures appelees:
c G05FAF de la librairie NAG Mark16
c Auteur: Pierre Lafaye de Micheaux
c Date: 15/02/2001
c Fin-Commentaires
      SUBROUTINE rTU (l,n,x)
      INTEGER n,i
      DOUBLE PRECISION x(n), a, b, l, y, z
      a=DBLE(0.0)
      b=DBLE(1.0)
      CALL G05FAF(a,b,n,x)
      DO 10, i=1,n
         y=x(i)**l
         z=(1-x(i))**l
         x(i)=y-z
 10      CONTINUE
      RETURN
      END
```

A.11. LES PROGRAMMES POUR LES DIFFÉRENTS POLYNÔMES DE LEGENDRE

Programme H1.f

```
c Debut-Commentaires
c Nom de la sous-routine: H1
c Entrees:
c n (entier), longueur du vecteur x
c x (double), un vecteur de n nombres
c Sorties: void
c Description:
c Calcul de H1(x), H1 etant le premier polynome de Legendre
c Modifie le vecteur x dans le programme principal, en le remplacant par H1(x)
c Utilisation dans une fonction main:
c-------------------------------------------
c       PROGRAM main
c       INTEGER n, i
cc       longueur du vecteur x, a modifier dans le programme principal
c       PARAMETER(n=3)
cc       initialisation de x, y et xavant
c       DOUBLE PRECISION x(n), y(n), xavant(n)
cc       affectation de valeurs a x
c       x(1)=1
c       x(2)=2
c       x(3)=3
cc       On sauvegarde les valeurs de x dans xavant
c       DO 10, i=1,n
c       xavant(i)=x(i)
c 10    CONTINUE
cc       calcul de H1(x), remplace x par H1(x)
c       CALL H1(n,x)
cc       affecte a y le resultat
c       DO 20, i=1,n
c       y(i)=x(i)
c 20    CONTINUE
cc       remet les bonnes valeurs dans x
cc       affecte a y le resultat
c       DO 30, i=1,n
```

```
c       x(i)=xavant(i)
c 30    CONTINUE
cc       Affichage des valeurs de y=H1(x)
c       WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de y'
c       WRITE(UNIT=6,FMT=*) y
cc       Affichage des valeurs de x
c       WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de x'
c       WRITE(UNIT=6,FMT=*) x
c       END
c-------------------------------------------
c Instructions de compilation: alias f77='fort77'
c f77 -c nom_du_fichier_contenant_la_fonction_main.f
c f77 -c H1.f
c f77 nom_du_fichier_contenant_la_fonction_main.o H1.o
c Fonctions exterieures appelees:
c Auteur: Pierre Lafaye de Micheaux
c Date: 15/02/2001
c Fin-Commentaires
      SUBROUTINE H1 (n,x)
      INTEGER n,i
      DOUBLE PRECISION x(n)
      DO 10, i=1,n
         x(i)=DSQRT(DBLE(3.0))*x(i)
 10      CONTINUE
      RETURN
      END
```

Programme H2.f

```
c Debut-Commentaires
c Nom de la sous-routine: H2
c Entrees:
c n (entier), longueur du vecteur x
c x (double), un vecteur de n nombres
c Sorties: void
c Description:
c Calcul de H2(x), H2 etant le deuxieme polynome de Legendre
c Modifie le vecteur x dans le programme principal, en le remplacant par H2(x)
c Utilisation dans une fonction main:
c-------------------------------------------
c       PROGRAM main
c       INTEGER n, i
cc       longueur du vecteur x, a modifier dans le programme principal
c       PARAMETER(n=3)
cc       initialisation de x, y et xavant
c       DOUBLE PRECISION x(n), y(n), xavant(n)
cc       affectation de valeurs a x
c       x(1)=1
c       x(2)=2
c       x(3)=3
cc       On sauvegarde les valeurs de x dans xavant
c       DO 10, i=1,n
c       xavant(i)=x(i)
c 10    CONTINUE
cc       calcul de H2(x), remplace x par H2(x)
c       CALL H2(n,x)
cc       affecte a y le resultat
c       DO 20, i=1,n
c       y(i)=x(i)
c 20    CONTINUE
cc       remet les bonnes valeurs dans x
cc       affecte a y le resultat
c       DO 30, i=1,n
c       x(i)=xavant(i)
c 30    CONTINUE
```

```
cc        Affichage des valeurs de y=H2(x)
c        WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de y'
c        WRITE(UNIT=6,FMT=*) y
cc        Affichage des valeurs de x
c        WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de x'
c        WRITE(UNIT=6,FMT=*) x
c        END
c-------------------------------------------
c Instructions de compilation: alias f77='fort77'
c f77 -c nom_du_fichier_contenant_la_fonction_main.f
c f77 -c H2.f
c f77 nom_du_fichier_contenant_la_fonction_main.o H2.o
c Fonctions exterieures appelees:
c Auteur: Pierre Lafaye de Micheaux
c Date: 15/02/2001
c Fin-Commentaires
      SUBROUTINE H2 (n,x)
      INTEGER n,i
      DOUBLE PRECISION x(n)
      DO 10, i=1,n
         x(i)=DSQRT(DBLE(5.0))*(3*(x(i)**2)-1)/2
 10      CONTINUE
      RETURN
      END
```

Programme H3.f

```
c Debut-Commentaires
c Nom de la sous-routine: H3
c Entrees:
c n (entier), longueur du vecteur x
c x (double), un vecteur de n nombres
c Sorties: void
c Description:
c Calcul de H3(x), H3 etant le troisieme polynome de Legendre
c Modifie le vecteur x dans le programme principal, en le remplacant par H3(x)
c Utilisation dans une fonction main:
c-------------------------------------
c       PROGRAM main
c       INTEGER n, i
cc       longueur du vecteur x, a modifier dans le programme principal
c       PARAMETER(n=3)
cc       initialisation de x, y et xavant
c       DOUBLE PRECISION x(n), y(n), xavant(n)
cc       affectation de valeurs a x
c       x(1)=1
c       x(2)=2
c       x(3)=3
cc       On sauvegarde les valeurs de x dans xavant
c       DO 10, i=1,n
c       xavant(i)=x(i)
c 10    CONTINUE
cc       calcul de H3(x), remplace x par H3(x)
c       CALL H3(n,x)
cc       affecte a y le resultat
c       DO 20, i=1,n
c       y(i)=x(i)
c 20    CONTINUE
cc       remet les bonnes valeurs dans x
cc       affecte a y le resultat
c       DO 30, i=1,n
c       x(i)=xavant(i)
c 30    CONTINUE
cc       Affichage des valeurs de y=H3(x)
c       WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de y'
```

```
c        WRITE(UNIT=6,FMT=*) y
cc        Affichage des valeurs de x
c        WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de x'
c        WRITE(UNIT=6,FMT=*) x
c        END
c-------------------------------------------
c Instructions de compilation: alias f77='fort77'
c f77 -c nom_du_fichier_contenant_la_fonction_main.f
c f77 -c H3.f
c f77 nom_du_fichier_contenant_la_fonction_main.o H3.o
c Fonctions exterieures appelees:
c Auteur: Pierre Lafaye de Micheaux
c Date: 15/02/2001
c Fin-Commentaires
      SUBROUTINE H3 (n,x)
      INTEGER n,i
      DOUBLE PRECISION x(n)
      DO 10, i=1,n
         x(i)=DSQRT(DBLE(7.0))*(5*(x(i)**3)-3*x(i))/2
 10      CONTINUE
      RETURN
      END
```

Programme H4.f

```
c Debut-Commentaires
c Nom de la sous-routine: H4
c Entrees:
c n (entier), longueur du vecteur x
c x (double), un vecteur de n nombres
c Sorties: void
c Description:
c Calcul de H4(x), H4 etant le troisieme polynome de Legendre
c Modifie le vecteur x dans le programme principal, en le remplacant par H4(x)
c Utilisation dans une fonction main:
c-------------------------------------------
c        PROGRAM main
c        INTEGER n, i
cc        longueur du vecteur x, a modifier dans le programme principal
c        PARAMETER(n=3)
cc        initialisation de x, y et xavant
c        DOUBLE PRECISION x(n), y(n), xavant(n)
cc        affectation de valeurs a x
c        x(1)=1
c        x(2)=2
c        x(3)=3
cc        On sauvegarde les valeurs de x dans xavant
c        DO 10, i=1,n
c        xavant(i)=x(i)
c 10     CONTINUE
cc        calcul de H4(x), remplace x par H4(x)
c        CALL H4(n,x)
cc        affecte a y le resultat
c        DO 20, i=1,n
c        y(i)=x(i)
c 20     CONTINUE
cc        remet les bonnes valeurs dans x
cc        affecte a y le resultat
c        DO 30, i=1,n
c        x(i)=xavant(i)
c 30     CONTINUE
cc        Affichage des valeurs de y=H4(x)
c        WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de y'
c        WRITE(UNIT=6,FMT=*) y
cc        Affichage des valeurs de x
```

```
c       WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de x'
c       WRITE(UNIT=6,FMT=*) x
c       END
c-------------------------------------------
c Instructions de compilation: alias f77='fort77'
c f77 -c nom_du_fichier_contenant_la_fonction_main.f
c f77 -c H4.f
c f77 nom_du_fichier_contenant_la_fonction_main.o H4.o
c Fonctions exterieures appelees:
c Auteur: Pierre Lafaye de Micheaux
c Date: 15/02/2001
c Fin-Commentaires
      SUBROUTINE H4 (n,x)
      INTEGER n,i
      DOUBLE PRECISION x(n)
      DO 10, i=1,n
         x(i)=3*(35*(x(i)**4)-30*(x(i)**2)+3)/8
 10      CONTINUE
      RETURN
      END
```

Programme H5.f

```
c Debut-Commentaires
c Nom de la sous-routine: H5
c Entrees:
c n (entier), longueur du vecteur x
c x (double), un vecteur de n nombres
c Sorties: void
c Description:
c Calcul de H5(x), H5 etant le cinquieme polynome de Legendre
c Modifie le vecteur x dans le programme principal, en le remplacant par H5(x)
c Utilisation dans une fonction main:
c-------------------------------------------
c       PROGRAM main
c       INTEGER n, i
cc       longueur du vecteur x, a modifier dans le programme principal
c       PARAMETER(n=3)
cc       initialisation de x, y et xavant
c       DOUBLE PRECISION x(n), y(n), xavant(n)
cc       affectation de valeurs a x
c       x(1)=1
c       x(2)=2
c       x(3)=3
cc       On sauvegarde les valeurs de x dans xavant
c       DO 10, i=1,n
c       xavant(i)=x(i)
c 10    CONTINUE
cc       calcul de H5(x), remplace x par H5(x)
c       CALL H5(n,x)
cc       affecte a y le resultat
c       DO 20, i=1,n
c       y(i)=x(i)
c 20    CONTINUE
cc       remet les bonnes valeurs dans x
cc       affecte a y le resultat
c       DO 30, i=1,n
c       x(i)=xavant(i)
c 30    CONTINUE
cc       Affichage des valeurs de y=H5(x)
c       WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de y'
c       WRITE(UNIT=6,FMT=*) y
cc       Affichage des valeurs de x
c       WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de x'
c       WRITE(UNIT=6,FMT=*) x
```

```
c       END
c-----------------------------------------------
c Instructions de compilation: alias f77='fort77'
c f77 -c nom_du_fichier_contenant_la_fonction_main.f
c f77 -c H5.f
c f77 nom_du_fichier_contenant_la_fonction_main.o H5.o
c Fonctions exterieures appelees:
c Auteur: Pierre Lafaye de Micheaux
c Date: 15/02/2001
c Fin-Commentaires
      SUBROUTINE H5 (n,x)
      INTEGER n,i
      DOUBLE PRECISION x(n)
      DO 10, i=1,n
         x(i)=DSQRT(DBLE(11.0))*(63*(x(i)**5)-70*(x(i)**3)+15*x(i))/8
 10      CONTINUE
      RETURN
      END
```

Programme H6.f

```
c Debut-Commentaires
c Nom de la sous-routine: H6
c Entrees:
c n (entier), longueur du vecteur x
c x (double), un vecteur de n nombres
c Sorties: void
c Description:
c Calcul de H6(x), H6 etant le sixieme polynome de Legendre
c Modifie le vecteur x dans le programme principal, en le remplacant par H6(x)
c Utilisation dans une fonction main:
c-----------------------------------------------
c       PROGRAM main
c       INTEGER n, i
cc       longueur du vecteur x, a modifier dans le programme principal
c       PARAMETER(n=3)
cc       initialisation de x, y et xavant
c       DOUBLE PRECISION x(n), y(n), xavant(n)
cc       affectation de valeurs a x
c       x(1)=1
c       x(2)=2
c       x(3)=3
cc       On sauvegarde les valeurs de x dans xavant
c       DO 10, i=1,n
c       xavant(i)=x(i)
c 10    CONTINUE
cc       calcul de H6(x), remplace x par H6(x)
c       CALL H6(n,x)
cc       affecte a y le resultat
c       DO 20, i=1,n
c       y(i)=x(i)
c 20    CONTINUE
cc       remet les bonnes valeurs dans x
cc       affecte a y le resultat
c       DO 30, i=1,n
c       x(i)=xavant(i)
c 30    CONTINUE
cc       Affichage des valeurs de y=H6(x)
c       WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de y'
c       WRITE(UNIT=6,FMT=*) y
cc       Affichage des valeurs de x
c       WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de x'
c       WRITE(UNIT=6,FMT=*) x
c       END
c-----------------------------------------------
```

```
c Instructions de compilation: alias f77='fort77'
c f77 -c nom_du_fichier_contenant_la_fonction_main.f
c f77 -c H6.f
c f77 nom_du_fichier_contenant_la_fonction_main.o H6.o
c Fonctions exterieures appelees:
c Auteur: Pierre Lafaye de Micheaux
c Date: 15/02/2001
c Fin-Commentaires
      SUBROUTINE H6 (n,x)
      INTEGER n,i
      DOUBLE PRECISION x(n)
      DO 10, i=1,n
         x(i)=DSQRT(DBLE(13.0))*(231*(x(i)**6)-315*(x(i)**4)+
     $        105*(x(i)**2)-5)/16
 10      CONTINUE
      RETURN
      END
```

Programme H7.f

```
c Debut-Commentaires
c Nom de la sous-routine: H7
c Entrees:
c n (entier), longueur du vecteur x
c x (double), un vecteur de n nombres
c Sorties: void
c Description:
c Calcul de H7(x), H7 etant le septieme polynome de Legendre
c Modifie le vecteur x dans le programme principal, en le remplacant par H7(x)
c Utilisation dans une fonction main:
c-------------------------------------------
c       PROGRAM main
c       INTEGER n, i
cc       longueur du vecteur x, a modifier dans le programme principal
c       PARAMETER(n=3)
cc       initialisation de x, y et xavant
c       DOUBLE PRECISION x(n), y(n), xavant(n)
cc       affectation de valeurs a x
c       x(1)=1
c       x(2)=2
c       x(3)=3
cc       On sauvegarde les valeurs de x dans xavant
c       DO 10, i=1,n
c       xavant(i)=x(i)
c 10    CONTINUE
cc       calcul de H7(x), remplace x par H7(x)
c       CALL H7(n,x)
cc       affecte a y le resultat
c       DO 20, i=1,n
c       y(i)=x(i)
c 20    CONTINUE
cc       remet les bonnes valeurs dans x
cc       affecte a y le resultat
c       DO 30, i=1,n
c       x(i)=xavant(i)
c 30    CONTINUE
cc       Affichage des valeurs de y=H7(x)
c       WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de y'
c       WRITE(UNIT=6,FMT=*) y
cc       Affichage des valeurs de x
c       WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de x'
c       WRITE(UNIT=6,FMT=*) x
c       END
c-------------------------------------------
c Instructions de compilation: alias f77='fort77'
```

```
c f77 -c nom_du_fichier_contenant_la_fonction_main.f
c f77 -c H7.f
c f77 nom_du_fichier_contenant_la_fonction_main.o H7.o
c Fonctions exterieures appelees:
c Auteur: Pierre Lafaye de Micheaux
c Date: 15/02/2001
c Fin-Commentaires
      SUBROUTINE H7 (n,x)
      INTEGER n,i
      DOUBLE PRECISION x(n)
      DO 10, i=1,n
         x(i)=DSQRT(DBLE(15.0))*(429*(x(i)**7)-693*(x(i)**5)+
     $        315*(x(i)**3)-35*x(i))/16
 10      CONTINUE
      RETURN
      END
```

Programme H8.f

```
c Debut-Commentaires
c Nom de la sous-routine: H8
c Entrees:
c n (entier), longueur du vecteur x
c x (double), un vecteur de n nombres
c Sorties: void
c Description:
c Calcul de H8(x), H8 etant le huitieme polynome de Legendre
c Modifie le vecteur x dans le programme principal, en le remplacant par H8(x)
c Utilisation dans une fonction main:
c-------------------------------------
c       PROGRAM main
c       INTEGER n, i
cc       longueur du vecteur x, a modifier dans le programme principal
c       PARAMETER(n=3)
cc       initialisation de x, y et xavant
c       DOUBLE PRECISION x(n), y(n), xavant(n)
cc       affectation de valeurs a x
c       x(1)=1
c       x(2)=2
c       x(3)=3
cc       On sauvegarde les valeurs de x dans xavant
c       DO 10, i=1,n
c         xavant(i)=x(i)
c       10 CONTINUE
cc       calcul de H8(x), remplace x par H8(x)
c       CALL H8(n,x)
cc       affecte a y le resultat
c       DO 20, i=1,n
c         y(i)=x(i)
c 20    CONTINUE
cc       remet les bonnes valeurs dans x
cc       affecte a y le resultat
c       DO 30, i=1,n
c         x(i)=xavant(i)
c 30    CONTINUE
cc       Affichage des valeurs de y=H8(x)
c       WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de y'
c       WRITE(UNIT=6,FMT=*) y
cc       Affichage des valeurs de x
c       WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de x'
c       WRITE(UNIT=6,FMT=*) x
c       END
c-------------------------------------
c Instructions de compilation: alias f77='fort77'
c f77 -c nom_du_fichier_contenant_la_fonction_main.f
```

```
c f77 -c H8.f
c f77 nom_du_fichier_contenant_la_fonction_main.o H8.o
c Fonctions exterieures appelees:
c Auteur: Pierre Lafaye de Micheaux
c Date: 15/02/2001
c Fin-Commentaires
      SUBROUTINE H8 (n,x)
      INTEGER n,i
      DOUBLE PRECISION x(n)
      DO 10, i=1,n
         x(i)=DSQRT(DBLE(17.0))*(6435*(x(i)**8)-12012*(x(i)**6)+
     $        6930*(x(i)**4)-1260*(x(i)**2)+35)/128
 10      CONTINUE
      RETURN
      END
```

Programme H9.f

```
c Debut-Commentaires
c Nom de la sous-routine: H9
c Entrees:
c n (entier), longueur du vecteur x
c x (double), un vecteur de n nombres
c Sorties: void
c Description:
c Calcul de H9(x), H9 etant le neuvieme polynome de Legendre
c Modifie le vecteur x dans le programme principal, en le remplacant par H9(x)
c Utilisation dans une fonction main:
c-------------------------------------
c       PROGRAM main
c       INTEGER n, i
cc       longueur du vecteur x, a modifier dans le programme principal
c       PARAMETER(n=3)
cc       initialisation de x, y et xavant
c       DOUBLE PRECISION x(n), y(n), xavant(n)
cc       affectation de valeurs a x
c       x(1)=1
c       x(2)=2
c       x(3)=3
cc       On sauvegarde les valeurs de x dans xavant
c       DO 10, i=1,n
c       xavant(i)=x(i)
c 10    CONTINUE
cc       calcul de H9(x), remplace x par H9(x)
c       CALL H9(n,x)
cc       affecte a y le resultat
c       DO 20, i=1,n
c       y(i)=x(i)
c 20    CONTINUE
cc       remet les bonnes valeurs dans x
cc       affecte a y le resultat
c       DO 30, i=1,n
c       x(i)=xavant(i)
c 30    CONTINUE
cc       Affichage des valeurs de y=H9(x)
c       WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de y'
c       WRITE(UNIT=6,FMT=*) y
cc       Affichage des valeurs de x
c       WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de x'
c       WRITE(UNIT=6,FMT=*) x
c       END
c-------------------------------------
c Instructions de compilation: alias f77='fort77'
c f77 -c nom_du_fichier_contenant_la_fonction_main.f
c f77 -c H9.f
```

```
c f77 nom_du_fichier_contenant_la_fonction_main.o H9.o
c Fonctions exterieures appelees:
c Auteur: Pierre Lafaye de Micheaux
c Date: 15/02/2001
c Fin-Commentaires
      SUBROUTINE H9 (n,x)
      INTEGER n,i
      DOUBLE PRECISION x(n)
      DO 10, i=1,n
         x(i)=DSQRT(DBLE(19.0))*(12155*(x(i)**9)-25740*(x(i)**7)+
     $        18018*(x(i)**5)-4620*(x(i)**3)+315*x(i))/128
 10      CONTINUE
      RETURN
      END
```

Programme H10.f

```
c Debut-Commentaires
c Nom de la sous-routine: H10
c Entrees:
c n (entier), longueur du vecteur x
c x (double), un vecteur de n nombres
c Sorties: void
c Description:
c Calcul de H10(x), H10 etant le dixieme polynome de Legendre
c Modifie le vecteur x dans le programme principal, en le remplacant par H10(x)
c Utilisation dans une fonction main:
c-------------------------------------------
c       PROGRAM main
c       INTEGER n, i
cc       longueur du vecteur x, a modifier dans le programme principal
c       PARAMETER(n=3)
cc       initialisation de x, y et xavant
c       DOUBLE PRECISION x(n), y(n), xavant(n)
cc       affectation de valeurs a x
c       x(1)=1
c       x(2)=2
c       x(3)=3
cc       On sauvegarde les valeurs de x dans xavant
c       DO 10, i=1,n
c       xavant(i)=x(i)
c 10    CONTINUE
cc       calcul de H10(x), remplace x par H10(x)
c       CALL H10(n,x)
cc       affecte a y le resultat
c       DO 20, i=1,n
c       y(i)=x(i)
c 20    CONTINUE
cc       remet les bonnes valeurs dans x
cc       affecte a y le resultat
c       DO 30, i=1,n
c       x(i)=xavant(i)
c 30    CONTINUE
cc       Affichage des valeurs de y=H10(x)
c       WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de y'
c       WRITE(UNIT=6,FMT=*) y
cc       Affichage des valeurs de x
c       WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de x'
c       WRITE(UNIT=6,FMT=*) x
c       END
c-------------------------------------------
c Instructions de compilation: alias f77='fort77'
c f77 -c nom_du_fichier_contenant_la_fonction_main.f
c f77 -c H10.f
c f77 nom_du_fichier_contenant_la_fonction_main.o H10.o
```

```
c Fonctions exterieures appelees:
c Auteur: Pierre Lafaye de Micheaux
c Date: 15/02/2001
c Fin-Commentaires
      SUBROUTINE H10 (n,x)
      INTEGER n,i
      DOUBLE PRECISION x(n)
      DO 10, i=1,n
         x(i)=DSQRT(DBLE(21.0))*(46189*(x(i)**10)-109395*(x(i)**8)+
     $        90090*(x(i)**6)-30030*(x(i)**4)+3465*(x(i)**2)-63)/256
 10      CONTINUE
      RETURN
      END
```

Programme H1isa.f

```
c Debut-Commentaires
c Nom de la sous-routine: H1isa
c Entrees: n entier, longueur du vecteur x ; x, un vecteur de n nombres de type double
c Sorties: void
c Description:
c Calcul de H1isa(x), H1isa etant le premier polynome de Legendre modifie d'isabelle
c Modifie le vecteur x dans le programme principal, en le remplacant par H1isa(x)
c Utilisation dans une fonction main:
c-----------------------------------------
c       PROGRAM main
c       INTEGER n, i
cc       longueur du vecteur x, a modifier dans le programme principal
c       PARAMETER(n=3)
cc       initialisation de x, y et xavant
c       DOUBLE PRECISION x(n), y(n), xavant(n)
cc       affectation de valeurs a x
c       x(1)=1
c       x(2)=2
c       x(3)=3
cc       On sauvegarde les valeurs de x dans xavant
c       DO 10, i=1,n
c       xavant(i)=x(i)
c 10    CONTINUE
cc       calcul de H1isa(x), remplace x par H1isa(x)
c       CALL H1isa(n,x)
cc       affecte a y le resultat
c       DO 20, i=1,n
c       y(i)=x(i)
c 20    CONTINUE
cc       remet les bonnes valeurs dans x
cc       affecte a y le resultat
c       DO 30, i=1,n
c       x(i)=xavant(i)
c 30    CONTINUE
cc       Affichage des valeurs de y=H1isa(x)
c       WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de y'
c       WRITE(UNIT=6,FMT=*) y
cc       Affichage des valeurs de x
c       WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de x'
c       WRITE(UNIT=6,FMT=*) x
c       END
c-----------------------------------------
c Instructions de compilation: alias f77='fort77'
c f77 -c nom_du_fichier_contenant_la_fonction_main.f
c f77 -c H1isa.f
c f77 nom_du_fichier_contenant_la_fonction_main.o H1isa.o
c Fonctions exterieures appelees:
c Auteur: Pierre Lafaye de Micheaux
c Date: 15/02/2001
```

```
c Fin-Commentaires
      SUBROUTINE H1isa (n,x)
      INTEGER n,i
      DOUBLE PRECISION x(n)
      DO 10, i=1,n
         x(i)=8.158591774*x(i)
 10      CONTINUE
      RETURN
      END
```

Programme H2isa.f

```
c Debut-Commentaires
c Nom de la sous-routine: H2isa
c Entrees: n entier, longueur du vecteur x ; x, un vecteur de n nombres de type double
c Sorties: void
c Description:
c Calcul de H2isa(x), H2isa etant le deuxieme polynome de Legendre modifie d'isabelle
c Modifie le vecteur x dans le programme principal, en le remplacant par H2isa(x)
c Utilisation dans une fonction main:
c-------------------------------------------
c       PROGRAM main
c       INTEGER n, i
cc       longueur du vecteur x, a modifier dans le programme principal
c       PARAMETER(n=3)
cc       initialisation de x, y et xavant
c       DOUBLE PRECISION x(n), y(n), xavant(n)
cc       affectation de valeurs a x
c       x(1)=1
c       x(2)=2
c       x(3)=3
cc       On sauvegarde les valeurs de x dans xavant
c       DO 10, i=1,n
c       xavant(i)=x(i)
c 10    CONTINUE
cc       calcul de H2isa(x), remplace x par H2isa(x)
c       CALL H2isa(n,x)
cc       affecte a y le resultat
c       DO 20, i=1,n
c       y(i)=x(i)
c 20    CONTINUE
cc       remet les bonnes valeurs dans x
cc       affecte a y le resultat
c       DO 30, i=1,n
c       x(i)=xavant(i)
c 30    CONTINUE
cc       Affichage des valeurs de y=H2isa(x)
c       WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de y'
c       WRITE(UNIT=6,FMT=*) y
cc       Affichage des valeurs de x
c       WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de x'
c       WRITE(UNIT=6,FMT=*) x
c       END
c-------------------------------------------
c Instructions de compilation: alias f77='fort77'
c f77 -c nom_du_fichier_contenant_la_fonction_main.f
c f77 -c H2isa.f
c f77 nom_du_fichier_contenant_la_fonction_main.o H2isa.o
c Fonctions exterieures appelees:
c Auteur: Pierre Lafaye de Micheaux
c Date: 15/02/2001
c Fin-Commentaires
      SUBROUTINE H2isa (n,x)
      INTEGER n,i
      DOUBLE PRECISION x(n)
```

```
      DO 10, i=1,n
         x(i)=-2.2817442+6.845232601*(x(i)**2)
 10      CONTINUE
      RETURN
      END
```

Programme H3isa.f

```
c Debut-Commentaires
c Nom de la sous-routine: H3isa
c Entrees: n entier, longueur du vecteur x ; x, un vecteur de n nombres de type double
c Sorties: void
c Description:
c Calcul de H3isa(x), H3isa etant le troisieme polynome de Legendre modifie d'isabelle
c Modifie le vecteur x dans le programme principal, en le remplacant par H3isa(x)
c Utilisation dans une fonction main:
c-----------------------------------------
c       PROGRAM main
c       INTEGER n, i
cc       longueur du vecteur x, a modifier dans le programme principal
c       PARAMETER(n=3)
cc       initialisation de x, y et xavant
c       DOUBLE PRECISION x(n), y(n), xavant(n)
cc       affectation de valeurs a x
c       x(1)=1
c       x(2)=2
c       x(3)=3
cc       On sauvegarde les valeurs de x dans xavant
c       DO 10, i=1,n
c       xavant(i)=x(i)
c 10    CONTINUE
cc       calcul de H3isa(x), remplace x par H3isa(x)
c       CALL H3isa(n,x)
cc       affecte a y le resultat
c       DO 20, i=1,n
c       y(i)=x(i)
c 20    CONTINUE
cc       remet les bonnes valeurs dans x
cc       affecte a y le resultat
c       DO 30, i=1,n
c       x(i)=xavant(i)
c 30    CONTINUE
cc       Affichage des valeurs de y=H3isa(x)
c       WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de y'
c       WRITE(UNIT=6,FMT=*) y
cc       Affichage des valeurs de x
c       WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de x'
c       WRITE(UNIT=6,FMT=*) x
c       END
c-----------------------------------------
c Instructions de compilation: alias f77='fort77'
c f77 -c nom_du_fichier_contenant_la_fonction_main.f
c f77 -c H3isa.f
c f77 nom_du_fichier_contenant_la_fonction_main.o H3isa.o
c Fonctions exterieures appelees:
c Auteur: Pierre Lafaye de Micheaux
c Date: 15/02/2001
c Fin-Commentaires
      SUBROUTINE H3isa (n,x)
      INTEGER n,i
      DOUBLE PRECISION x(n)
      DO 10, i=1,n
         x(i)=5.729636522*x(i)+13.050032*(x(i)**3)
 10      CONTINUE
      RETURN
```

```
      END
```

Programme H4isa.f

```
c Debut-Commentaires
c Nom de la sous-routine: H4isa
c Entrees: n entier, longueur du vecteur x ; x, un vecteur de n nombres de type double
c Sorties: void
c Description:
c Calcul de H4isa(x), H4isa etant le quatrieme polynome de Legendre modifie d'isabelle
c Modifie le vecteur x dans le programme principal, en le remplacant par H4isa(x)
c Utilisation dans une fonction main:
c-----------------------------------------
c       PROGRAM main
c       INTEGER n, i
cc       longueur du vecteur x, a modifier dans le programme principal
c       PARAMETER(n=3)
cc       initialisation de x, y et xavant
c       DOUBLE PRECISION x(n), y(n), xavant(n)
cc       affectation de valeurs a x
c       x(1)=1
c       x(2)=2
c       x(3)=3
cc       On sauvegarde les valeurs de x dans xavant
c       DO 10, i=1,n
c       xavant(i)=x(i)
c 10    CONTINUE
cc       calcul de H4isa(x), remplace x par H4isa(x)
c       CALL H4isa(n,x)
cc       affecte a y le resultat
c       DO 20, i=1,n
c       y(i)=x(i)
c 20    CONTINUE
cc       remet les bonnes valeurs dans x
cc       affecte a y le resultat
c       DO 30, i=1,n
c       x(i)=xavant(i)
c 30    CONTINUE
cc       Affichage des valeurs de y=H4isa(x)
c       WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de y'
c       WRITE(UNIT=6,FMT=*) y
cc       Affichage des valeurs de x
c       WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de x'
c       WRITE(UNIT=6,FMT=*) x
c       END
c-----------------------------------------
c Instructions de compilation: alias f77='fort77'
c f77 -c nom_du_fichier_contenant_la_fonction_main.f
c f77 -c H4isa.f
c f77 nom_du_fichier_contenant_la_fonction_main.o H4isa.o
c Fonctions exterieures appelees:
c Auteur: Pierre Lafaye de Micheaux
c Date: 15/02/2001
c Fin-Commentaires
      SUBROUTINE H4isa (n,x)
      INTEGER n,i
      DOUBLE PRECISION x(n)
      DO 10, i=1,n
         x(i)=-0.562632699-10.25980936*(x(i)**2)+19.91284577*(x(i)**4)
 10      CONTINUE
      RETURN
      END
```

Programme H5isa.f

```
c Debut-Commentaires
```

```
c Nom de la sous-routine: H5isa
c Entrees: n entier, longueur du vecteur x ; x, un vecteur de n nombres de type double
c Sorties: void
c Description:
c Calcul de H5isa(x), H5isa etant le cinquieme polynome de Legendre modifie d'isabelle
c Modifie le vecteur x dans le programme principal, en le remplacant par H5isa(x)
c Utilisation dans une fonction main:
c-------------------------------------------
c       PROGRAM main
c       INTEGER n, i
cc       longueur du vecteur x, a modifier dans le programme principal
c       PARAMETER(n=3)
cc       initialisation de x, y et xavant
c       DOUBLE PRECISION x(n), y(n), xavant(n)
cc       affectation de valeurs a x
c       x(1)=1
c       x(2)=2
c       x(3)=3
cc       On sauvegarde les valeurs de x dans xavant
c       DO 10, i=1,n
c       xavant(i)=x(i)
c 10    CONTINUE
cc       calcul de H5isa(x), remplace x par H5isa(x)
c       CALL H5isa(n,x)
cc       affecte a y le resultat
c       DO 20, i=1,n
c       y(i)=x(i)
c 20    CONTINUE
cc       remet les bonnes valeurs dans x
cc       affecte a y le resultat
c       DO 30, i=1,n
c       x(i)=xavant(i)
c 30    CONTINUE
cc       Affichage des valeurs de y=H5isa(x)
c       WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de y'
c       WRITE(UNIT=6,FMT=*) y
cc       Affichage des valeurs de x
c       WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de x'
c       WRITE(UNIT=6,FMT=*) x
c       END
c-------------------------------------------
c Instructions de compilation: alias f77='fort77'
c f77 -c nom_du_fichier_contenant_la_fonction_main.f
c f77 -c H5isa.f
c f77 nom_du_fichier_contenant_la_fonction_main.o H5isa.o
c Fonctions exterieures appelees:
c Auteur: Pierre Lafaye de Micheaux
c Date: 15/02/2001
c Fin-Commentaires
      SUBROUTINE H5isa (n,x)
      INTEGER n,i
      DOUBLE PRECISION x(n)
      DO 10, i=1,n
         x(i)=20.03273031*x(i)-31.46840914*(x(i)**3)
     $     +40.13397202*(x(i)**5)
 10      CONTINUE
      RETURN
      END
```

Programme H6isa.f

```
c Debut-Commentaires
c Nom de la sous-routine: H6isa
c Entrees: n entier, longueur du vecteur x ; x, un vecteur de n nombres de type double
c Sorties: void
```

```
c Description:
c Calcul de H6isa(x), H6isa etant le sixieme polynome de Legendre modifie d'isabelle
c Modifie le vecteur x dans le programme principal, en le remplacant par H6isa(x)
c Utilisation dans une fonction main:
c-------------------------------------------
c       PROGRAM main
c       INTEGER n, i
cc       longueur du vecteur x, a modifier dans le programme principal
c       PARAMETER(n=3)
cc       initialisation de x, y et xavant
c       DOUBLE PRECISION x(n), y(n), xavant(n)
cc       affectation de valeurs a x
c       x(1)=1
c       x(2)=2
c       x(3)=3
cc       On sauvegarde les valeurs de x dans xavant
c       DO 10, i=1,n
c       xavant(i)=x(i)
c 10    CONTINUE
cc       calcul de H6isa(x), remplace x par H6isa(x)
c       CALL H6isa(n,x)
cc       affecte a y le resultat
c       DO 20, i=1,n
c       y(i)=x(i)
c 20    CONTINUE
cc       remet les bonnes valeurs dans x
cc       affecte a y le resultat
c       DO 30, i=1,n
c       x(i)=xavant(i)
c 30    CONTINUE
cc       Affichage des valeurs de y=H6isa(x)
c       WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de y'
c       WRITE(UNIT=6,FMT=*) y
cc       Affichage des valeurs de x
c       WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de x'
c       WRITE(UNIT=6,FMT=*) x
c       END
c-------------------------------------------
c Instructions de compilation: alias f77='fort77'
c f77 -c nom_du_fichier_contenant_la_fonction_main.f
c f77 -c H6isa.f
c f77 nom_du_fichier_contenant_la_fonction_main.o H6isa.o
c Fonctions exterieures appelees:
c Auteur: Pierre Lafaye de Micheaux
c Date: 15/02/2001
c Fin-Commentaires
      SUBROUTINE H6isa (n,x)
      INTEGER n,i
      DOUBLE PRECISION x(n)
      DO 10, i=1,n
         x(i)=-3.06324985+28.36318105*(x(i)**2)-81.84483538*(x(i)**4)
     $              +69.84318863*(x(i)**6)
 10      CONTINUE
      RETURN
      END
```

Programme H7isa.f

```
c Debut-Commentaires
c Nom de la sous-routine: H7isa
c Entrees: n entier, longueur du vecteur x ; x, un vecteur de n nombres de type double
c Sorties: void
c Description:
c Calcul de H7isa(x), H7isa etant le septieme polynome de Legendre modifie d'isabelle
c Modifie le vecteur x dans le programme principal, en le remplacant par H7isa(x)
```

```
c Utilisation dans une fonction main:
c-------------------------------------------
c       PROGRAM main
c       INTEGER n, i
cc       longueur du vecteur x, a modifier dans le programme principal
c       PARAMETER(n=3)
cc       initialisation de x, y et xavant
c       DOUBLE PRECISION x(n), y(n), xavant(n)
cc       affectation de valeurs a x
c       x(1)=1
c       x(2)=2
c       x(3)=3
cc       On sauvegarde les valeurs de x dans xavant
c       DO 10, i=1,n
c       xavant(i)=x(i)
c 10    CONTINUE
cc       calcul de H7isa(x), remplace x par H7isa(x)
c       CALL H7isa(n,x)
cc       affecte a y le resultat
c       DO 20, i=1,n
c       y(i)=x(i)
c 20    CONTINUE
cc       remet les bonnes valeurs dans x
cc       affecte a y le resultat
c       DO 30, i=1,n
c       x(i)=xavant(i)
c 30    CONTINUE
cc       Affichage des valeurs de y=H7isa(x)
c       WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de y'
c       WRITE(UNIT=6,FMT=*) y
cc       Affichage des valeurs de x
c       WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de x'
c       WRITE(UNIT=6,FMT=*) x
c       END
c-------------------------------------------
c Instructions de compilation: alias f77='fort77'
c f77 -c nom_du_fichier_contenant_la_fonction_main.f
c f77 -c H7isa.f
c f77 nom_du_fichier_contenant_la_fonction_main.o H7isa.o
c Fonctions exterieures appelees:
c Auteur: Pierre Lafaye de Micheaux
c Date: 15/02/2001
c Fin-Commentaires
      SUBROUTINE H7isa (n,x)
      INTEGER n,i
      DOUBLE PRECISION x(n)
      DO 10, i=1,n
         x(i)=8.033523458*(x(i))+88.76807724*(x(i)**3)
     $             -200.8654452*(x(i)**5)+141.9167731*(x(i)**7)
 10      CONTINUE
      RETURN
      END
```

Programme H8isa.f

```
c Debut-Commentaires
c Nom de la sous-routine: H8isa
c Entrees: n entier, longueur du vecteur x ; x, un vecteur de n nombres de type double
c Sorties: void
c Description:
c Calcul de H8isa(x), H8isa etant le huitieme polynome de Legendre modifie d'isabelle
c Modifie le vecteur x dans le programme principal, en le remplacant par H8isa(x)
c Utilisation dans une fonction main:
c-------------------------------------------
c       PROGRAM main
```

```
c       INTEGER n, i
cc       longueur du vecteur x, a modifier dans le programme principal
c       PARAMETER(n=3)
cc       initialisation de x, y et xavant
c       DOUBLE PRECISION x(n), y(n), xavant(n)
cc       affectation de valeurs a x
c       x(1)=1
c       x(2)=2
c       x(3)=3
cc       On sauvegarde les valeurs de x dans xavant
c       DO 10, i=1,n
c       xavant(i)=x(i)
c 10    CONTINUE
cc       calcul de H8isa(x), remplace x par H8isa(x)
c       CALL H8isa(n,x)
cc       affecte a y le resultat
c       DO 20, i=1,n
c       y(i)=x(i)
c 20    CONTINUE
cc       remet les bonnes valeurs dans x
cc       affecte a y le resultat
c       DO 30, i=1,n
c       x(i)=xavant(i)
c 30    CONTINUE
cc       Affichage des valeurs de y=H8isa(x)
c       WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de y'
c       WRITE(UNIT=6,FMT=*) y
cc       Affichage des valeurs de x
c       WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de x'
c       WRITE(UNIT=6,FMT=*) x
c       END
c-------------------------------------------
c Instructions de compilation: alias f77='fort77'
c f77 -c nom_du_fichier_contenant_la_fonction_main.f
c f77 -c H8isa.f
c f77 nom_du_fichier_contenant_la_fonction_main.o H8isa.o
c Fonctions exterieures appelees:
c Auteur: Pierre Lafaye de Micheaux
c Date: 15/02/2001
c Fin-Commentaires
      SUBROUTINE H8isa (n,x)
      INTEGER n,i
      DOUBLE PRECISION x(n)
      DO 10, i=1,n
         x(i)=-1.112255254-38.72988351*(x(i)**2)+247.1833561*(x(i)**4)
     $             -450.1763319*(x(i)**6)+260.0680479*(x(i)**8)
 10      CONTINUE
      RETURN
      END
```

Programme H9isa.f

```
c Debut-Commentaires
c Nom de la sous-routine: H9isa
c Entrees: n entier, longueur du vecteur x ; x, un vecteur de n nombres de type double
c Sorties: void
c Description:
c Calcul de H9isa(x), H9isa etant le neuvieme polynome de Legendre modifie d'isabelle
c Modifie le vecteur x dans le programme principal, en le remplacant par H9isa(x)
c Utilisation dans une fonction main:
c-------------------------------------------
c       PROGRAM main
c       INTEGER n, i
cc       longueur du vecteur x, a modifier dans le programme principal
c       PARAMETER(n=3)
```

```
cc       initialisation de x, y et xavant
c       DOUBLE PRECISION x(n), y(n), xavant(n)
cc       affectation de valeurs a x
c       x(1)=1
c       x(2)=2
c       x(3)=3
cc       On sauvegarde les valeurs de x dans xavant
c       DO 10, i=1,n
c       xavant(i)=x(i)
c 10    CONTINUE
cc       calcul de H9isa(x), remplace x par H9isa(x)
c       CALL H9isa(n,x)
cc       affecte a y le resultat
c       DO 20, i=1,n
c       y(i)=x(i)
c 20    CONTINUE
cc       remet les bonnes valeurs dans x
cc       affecte a y le resultat
c       DO 30, i=1,n
c       x(i)=xavant(i)
c 30    CONTINUE
cc       Affichage des valeurs de y=H9isa(x)
c       WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de y'
c       WRITE(UNIT=6,FMT=*) y
cc       Affichage des valeurs de x
c       WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de x'
c       WRITE(UNIT=6,FMT=*) x
c       END
c-------------------------------------------------
c Instructions de compilation: alias f77='fort77'
c f77 -c nom_du_fichier_contenant_la_fonction_main.f
c f77 -c H9isa.f
c f77 nom_du_fichier_contenant_la_fonction_main.o H9isa.o
c Fonctions exterieures appelees:
c Auteur: Pierre Lafaye de Micheaux
c Date: 15/02/2001
c Fin-Commentaires
      SUBROUTINE H9isa (n,x)
      INTEGER n,i
      DOUBLE PRECISION x(n)
      DO 10, i=1,n
         x(i)=30.29753057*x(i)-162.5156573*(x(i)**3)+
     $          692.6715145*(x(i)**5)-1042.779433*(x(i)**7)
     $         +528.7259394*(x(i)**9)
 10      CONTINUE
      RETURN
      END
```

Programme H10isa.f

```
c Debut-Commentaires
c Nom de la sous-routine: H10isa
c Entrees: n entier, longueur du vecteur x ; x, un vecteur de n nombres de type double
c Sorties: void
c Description:
c Calcul de H10isa(x), H10isa etant le dixieme polynome de Legendre modifie d'isabelle
c Modifie le vecteur x dans le programme principal, en le remplacant par H10isa(x)
c Utilisation dans une fonction main:
c-------------------------------------------------
c       PROGRAM main
c       INTEGER n, i
cc       longueur du vecteur x, a modifier dans le programme principal
c       PARAMETER(n=3)
cc       initialisation de x, y et xavant
c       DOUBLE PRECISION x(n), y(n), xavant(n)
```

```
cc        affectation de valeurs a x
c        x(1)=1
c        x(2)=2
c        x(3)=3
cc        On sauvegarde les valeurs de x dans xavant
c        DO 10, i=1,n
c        xavant(i)=x(i)
c 10     CONTINUE
cc        calcul de H10isa(x), remplace x par H10isa(x)
c        CALL H10isa(n,x)
cc        affecte a y le resultat
c        DO 20, i=1,n
c        y(i)=x(i)
c 20     CONTINUE
cc        remet les bonnes valeurs dans x
cc        affecte a y le resultat
c        DO 30, i=1,n
c        x(i)=xavant(i)
c 30     CONTINUE
cc        Affichage des valeurs de y=H10isa(x)
c        WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de y'
c        WRITE(UNIT=6,FMT=*) y
cc        Affichage des valeurs de x
c        WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de x'
c        WRITE(UNIT=6,FMT=*) x
c        END
c-------------------------------------------
c Instructions de compilation: alias f77='fort77'
c f77 -c nom_du_fichier_contenant_la_fonction_main.f
c f77 -c H10isa.f
c f77 nom_du_fichier_contenant_la_fonction_main.o H10isa.o
c Fonctions exterieures appelees:
c Auteur: Pierre Lafaye de Micheaux
c Date: 15/02/2001
c Fin-Commentaires
      SUBROUTINE H10isa (n,x)
      INTEGER n,i
      DOUBLE PRECISION x(n)
      DO 10, i=1,n
         x(i)=-3.58188372+67.54323446*(x(i)**2)-569.921170098*(x(i)**4)
     $           +1782.81828*(x(i)**6)-2250.408033*(x(i)**8)
     $           +994.4970803*(x(i)**10)
 10      CONTINUE
      RETURN
      END
```

Programme H1etoile.f

```
c Debut-Commentaires
c Nom de la sous-routine: H1etoi
c Entrees:
c n (entier), longueur du vecteur x
c x (double), un vecteur de n nombres
c Sorties: void
c Description:
c Calcul de H1etoile(x), H1etoile etant le premier polynome de Legendre modifie
c Modifie le vecteur x dans le programme principal, en le remplacant par H1etoile(x)
c Utilisation dans une fonction main:
c-------------------------------------------
c        PROGRAM main
c        INTEGER n, i
cc        longueur du vecteur x, a modifier dans le programme principal
c        PARAMETER(n=3)
cc        initialisation de x, y et xavant
c        DOUBLE PRECISION x(n), y(n), xavant(n)
```

```
cc        affectation de valeurs a x
c        x(1)=1
c        x(2)=2
c        x(3)=3
cc        On sauvegarde les valeurs de x dans xavant
c        DO 10, i=1,n
c        xavant(i)=x(i)
c 10     CONTINUE
cc        calcul de H1etoile(x), remplace x par H1etoile(x)
c        CALL H1etoi(n,x)
cc        affecte a y le resultat
c        DO 20, i=1,n
c        y(i)=x(i)
c 20     CONTINUE
cc        remet les bonnes valeurs dans x
cc        affecte a y le resultat
c        DO 30, i=1,n
c        x(i)=xavant(i)
c 30     CONTINUE
cc        Affichage des valeurs de y=H1etoile(x)
c        WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de y'
c        WRITE(UNIT=6,FMT=*) y
cc        Affichage des valeurs de x
c        WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de x'
c        WRITE(UNIT=6,FMT=*) x
c        END
c-------------------------------------------
c Instructions de compilation: alias f77='fort77'
c f77 -c nom_du_fichier_contenant_la_fonction_main.f
c f77 -c H1etoile.f
c f77 nom_du_fichier_contenant_la_fonction_main.o H1etoile.o
c Fonctions exterieures appelees:
c Auteur: Pierre Lafaye de Micheaux
c Date: 15/02/2001
c Fin-Commentaires
      SUBROUTINE H1etoi (n,x)
      INTEGER n,i
      DOUBLE PRECISION x(n)
      DO 10, i=1,n
         x(i)=1.73205*x(i)
 10      CONTINUE
      RETURN
      END
```

Programme H2etoile.f

```
c Debut-Commentaires
c Nom de la sous-routine: H2etoi
c Entrees:
c n (entier), longueur du vecteur x
c x (double), un vecteur de n nombres
c Sorties: void
c Description:
c Calcul de H2etoile(x), H2etoile etant le deuxieme polynome de Legendre modifie
c Modifie le vecteur x dans le programme principal, en le remplacant par H2etoile(x)
c Utilisation dans une fonction main:
c-------------------------------------------
c        PROGRAM main
c        INTEGER n, i
cc        longueur du vecteur x, a modifier dans le programme principal
c        PARAMETER(n=3)
cc        initialisation de x, y et xavant
c        DOUBLE PRECISION x(n), y(n), xavant(n)
cc        affectation de valeurs a x
c        x(1)=1
```

```
c       x(2)=2
c       x(3)=3
cc       On sauvegarde les valeurs de x dans xavant
c       DO 10, i=1,n
c       xavant(i)=x(i)
c 10    CONTINUE
cc       calcul de H2etoile(x), remplace x par H2etoile(x)
c       CALL H2etoi(n,x)
cc       affecte a y le resultat
c       DO 20, i=1,n
c       y(i)=x(i)
c 20    CONTINUE
cc       remet les bonnes valeurs dans x
cc       affecte a y le resultat
c       DO 30, i=1,n
c       x(i)=xavant(i)
c 30    CONTINUE
cc       Affichage des valeurs de y=H2etoile(x)
c       WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de y'
c       WRITE(UNIT=6,FMT=*) y
cc       Affichage des valeurs de x
c       WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de x'
c       WRITE(UNIT=6,FMT=*) x
c       END
c-------------------------------------------
c Instructions de compilation: alias f77='fort77'
c f77 -c nom_du_fichier_contenant_la_fonction_main.f
c f77 -c H2etoile.f
c f77 nom_du_fichier_contenant_la_fonction_main.o H2etoile.o
c Fonctions exterieures appelees:
c Auteur: Pierre Lafaye de Micheaux
c Date: 15/02/2001
c Fin-Commentaires
      SUBROUTINE H2etoi (n,x)
      INTEGER n,i
      DOUBLE PRECISION x(n)
      DO 10, i=1,n
         x(i)=6.84525*(x(i)**2)-2.28175
 10      CONTINUE
      RETURN
      END
```

Programme H3etoile.f

```
c Debut-Commentaires
c Nom de la sous-routine: H3etoi
c Entrees:
c n (entier), longueur du vecteur x
c x (double), un vecteur de n nombres
c Sorties: void
c Description:
c Calcul de H3etoile(x), H3etoile etant le troisieme polynome de Legendre modifie
c Modifie le vecteur x dans le programme principal, en le remplacant par H3etoile(x)
c Utilisation dans une fonction main:
c-------------------------------------------
c       PROGRAM main
c       INTEGER n, i
cc       longueur du vecteur x, a modifier dans le programme principal
c       PARAMETER(n=3)
cc       initialisation de x, y et xavant
c       DOUBLE PRECISION x(n), y(n), xavant(n)
cc       affectation de valeurs a x
c       x(1)=1
c       x(2)=2
c       x(3)=3
```

```
cc       On sauvegarde les valeurs de x dans xavant
c       DO 10, i=1,n
c       xavant(i)=x(i)
c 10    CONTINUE
cc       calcul de H3etoile(x), remplace x par H3etoile(x)
c       CALL H3etoi(n,x)
cc       affecte a y le resultat
c       DO 20, i=1,n
c       y(i)=x(i)
c 20    CONTINUE
cc       remet les bonnes valeurs dans x
cc       affecte a y le resultat
c       DO 30, i=1,n
c       x(i)=xavant(i)
c 30    CONTINUE
cc       Affichage des valeurs de y=H3etoile(x)
c       WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de y'
c       WRITE(UNIT=6,FMT=*) y
cc       Affichage des valeurs de x
c       WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de x'
c       WRITE(UNIT=6,FMT=*) x
c       END
c-------------------------------------
c Instructions de compilation: alias f77='fort77'
c f77 -c nom_du_fichier_contenant_la_fonction_main.f
c f77 -c H3etoile.f
c f77 nom_du_fichier_contenant_la_fonction_main.o H3etoile.o
c Fonctions exterieures appelees:
c Auteur: Pierre Lafaye de Micheaux
c Date: 15/02/2001
c Fin-Commentaires
      SUBROUTINE H3etoi (n,x)
      INTEGER n,i
      DOUBLE PRECISION x(n)
      DO 10, i=1,n
         x(i)=6.61438*(x(i)**3)-3.96863*x(i)
 10      CONTINUE
      RETURN
      END
```

Programme H4etoile.f

```
c Debut-Commentaires
c Nom de la sous-routine: H4etoi
c Entrees:
c n (entier), longueur du vecteur x
c x (double), un vecteur de n nombres
c Sorties: void
c Description:
c Calcul de H4etoile(x), H4etoile etant le quatrieme polynome de Legendre modifie
c Modifie le vecteur x dans le programme principal, en le remplacant par H4etoile(x)
c Utilisation dans une fonction main:
c-------------------------------------
c       PROGRAM main
c       INTEGER n, i
cc       longueur du vecteur x, a modifier dans le programme principal
c       PARAMETER(n=3)
cc       initialisation de x, y et xavant
c       DOUBLE PRECISION x(n), y(n), xavant(n)
cc       affectation de valeurs a x
c       x(1)=1
c       x(2)=2
c       x(3)=3
cc       On sauvegarde les valeurs de x dans xavant
c       DO 10, i=1,n
```

```
c        xavant(i)=x(i)
c 10     CONTINUE
cc        calcul de H4etoile(x), remplace x par H4etoile(x)
c        CALL H4etoi(n,x)
cc        affecte a y le resultat
c        DO 20, i=1,n
c        y(i)=x(i)
c 20     CONTINUE
cc        remet les bonnes valeurs dans x
cc        affecte a y le resultat
c        DO 30, i=1,n
c        x(i)=xavant(i)
c 30     CONTINUE
cc        Affichage des valeurs de y=H4etoile(x)
c        WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de y'
c        WRITE(UNIT=6,FMT=*) y
cc        Affichage des valeurs de x
c        WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de x'
c        WRITE(UNIT=6,FMT=*) x
c        END
c-------------------------------------------
c Instructions de compilation: alias f77='fort77'
c f77 -c nom_du_fichier_contenant_la_fonction_main.f
c f77 -c H4etoile.f
c f77 nom_du_fichier_contenant_la_fonction_main.o H4etoile.o
c Fonctions exterieures appelees:
c Auteur: Pierre Lafaye de Micheaux
c Date: 15/02/2001
c Fin-Commentaires
      SUBROUTINE H4etoi (n,x)
      INTEGER n,i
      DOUBLE PRECISION x(n)
      DO 10, i=1,n
         x(i)=19.9129*(x(i)**4)-10.2598*(x(i)**2)-0.562652
 10      CONTINUE
      RETURN
      END
```

Programme H5etoile.f

```
c Debut-Commentaires
c Nom de la sous-routine: H5etoi
c Entrees:
c n (entier), longueur du vecteur x
c x (double), un vecteur de n nombres
c Sorties: void
c Description:
c Calcul de H5etoile(x), H5etoile etant le cinquieme polynome de Legendre modifie
c Modifie le vecteur x dans le programme principal, en le remplacant par H5etoile(x)
c Utilisation dans une fonction main:
c-------------------------------------------
c        PROGRAM main
c        INTEGER n, i
cc        longueur du vecteur x, a modifier dans le programme principal
c        PARAMETER(n=3)
cc        initialisation de x, y et xavant
c        DOUBLE PRECISION x(n), y(n), xavant(n)
cc        affectation de valeurs a x
c        x(1)=1
c        x(2)=2
c        x(3)=3
cc        On sauvegarde les valeurs de x dans xavant
c        DO 10, i=1,n
c        xavant(i)=x(i)
c 10     CONTINUE
```

```
cc       calcul de H5etoile(x), remplace x par H5etoile(x)
c       CALL H5etoi(n,x)
cc       affecte a y le resultat
c       DO 20, i=1,n
c       y(i)=x(i)
c 20    CONTINUE
cc       remet les bonnes valeurs dans x
cc       affecte a y le resultat
c       DO 30, i=1,n
c       x(i)=xavant(i)
c 30    CONTINUE
cc       Affichage des valeurs de y=H5etoile(x)
c       WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de y'
c       WRITE(UNIT=6,FMT=*) y
cc       Affichage des valeurs de x
c       WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de x'
c       WRITE(UNIT=6,FMT=*) x
c       END
c-------------------------------------------
c Instructions de compilation: alias f77='fort77'
c f77 -c nom_du_fichier_contenant_la_fonction_main.f
c f77 -c H5etoile.f
c f77 nom_du_fichier_contenant_la_fonction_main.o H5etoile.o
c Fonctions exterieures appelees:
c Auteur: Pierre Lafaye de Micheaux
c Date: 15/02/2001
c Fin-Commentaires
      SUBROUTINE H5etoi (n,x)
      INTEGER n,i
      DOUBLE PRECISION x(n)
      DO 10, i=1,n
         x(i)=26.1184*(x(i)**5)-29.0205*(x(i)**3)+6.21867*x(i)
 10      CONTINUE
      RETURN
      END
```

Programme H6etoile.f

```
c Debut-Commentaires
c Nom de la sous-routine: H6etoi
c Entrees:
c n (entier), longueur du vecteur x
c x (double), un vecteur de n nombres
c Sorties: void
c Description:
c Calcul de H6etoile(x), H6etoile etant le sixieme polynome de Legendre modifie
c Modifie le vecteur x dans le programme principal, en le remplacant par H6etoile(x)
c Utilisation dans une fonction main:
c-------------------------------------------
c       PROGRAM main
c       INTEGER n, i
cc       longueur du vecteur x, a modifier dans le programme principal
c       PARAMETER(n=3)
cc       initialisation de x, y et xavant
c       DOUBLE PRECISION x(n), y(n), xavant(n)
cc       affectation de valeurs a x
c       x(1)=1
c       x(2)=2
c       x(3)=3
cc       On sauvegarde les valeurs de x dans xavant
c       DO 10, i=1,n
c       xavant(i)=x(i)
c 10    CONTINUE
cc       calcul de H6etoile(x), remplace x par H6etoile(x)
c       CALL H6etoi(n,x)
```

```
cc        affecte a y le resultat
c        DO 20, i=1,n
c        y(i)=x(i)
c 20     CONTINUE
cc        remet les bonnes valeurs dans x
cc        affecte a y le resultat
c        DO 30, i=1,n
c        x(i)=xavant(i)
c 30     CONTINUE
cc        Affichage des valeurs de y=H6etoile(x)
c        WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de y'
c        WRITE(UNIT=6,FMT=*) y
cc        Affichage des valeurs de x
c        WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de x'
c        WRITE(UNIT=6,FMT=*) x
c        END
c-------------------------------------------
c Instructions de compilation: alias f77='fort77'
c f77 -c nom_du_fichier_contenant_la_fonction_main.f
c f77 -c H6etoile.f
c f77 nom_du_fichier_contenant_la_fonction_main.o H6etoile.o
c Fonctions exterieures appelees:
c Auteur: Pierre Lafaye de Micheaux
c Date: 15/02/2001
c Fin-Commentaires
      SUBROUTINE H6etoi (n,x)
      INTEGER n,i
      DOUBLE PRECISION x(n)
      DO 10, i=1,n
      x(i)=69.8435*(x(i)**6)-81.845*(x(i)**4)+28.3633*(x(i)**2)-3.06303
 10      CONTINUE
      RETURN
      END
```

Programme H7etoile.f

```
c Debut-Commentaires
c Nom de la sous-routine: H7etoi
c Entrees:
c n (entier), longueur du vecteur x
c x (double), un vecteur de n nombres
c Sorties: void
c Description:
c Calcul de H7etoile(x), H7etoile etant le septieme polynome de Legendre modifie
c Modifie le vecteur x dans le programme principal, en le remplacant par H7etoile(x)
c Utilisation dans une fonction main:
c-------------------------------------------
c        PROGRAM main
c        INTEGER n, i
cc        longueur du vecteur x, a modifier dans le programme principal
c        PARAMETER(n=3)
cc        initialisation de x, y et xavant
c        DOUBLE PRECISION x(n), y(n), xavant(n)
cc        affectation de valeurs a x
c        x(1)=1
c        x(2)=2
c        x(3)=3
cc        On sauvegarde les valeurs de x dans xavant
c        DO 10, i=1,n
c        xavant(i)=x(i)
c 10     CONTINUE
cc        calcul de H7etoile(x), remplace x par H7etoile(x)
c        CALL H7etoi(n,x)
cc        affecte a y le resultat
c        DO 20, i=1,n
```

```
c       y(i)=x(i)
c 20    CONTINUE
cc       remet les bonnes valeurs dans x
cc       affecte a y le resultat
c       DO 30, i=1,n
c       x(i)=xavant(i)
c 30    CONTINUE
cc       Affichage des valeurs de y=H7etoile(x)
c       WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de y'
c       WRITE(UNIT=6,FMT=*) y
cc       Affichage des valeurs de x
c       WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de x'
c       WRITE(UNIT=6,FMT=*) x
c       END
c-------------------------------------------
c Instructions de compilation: alias f77='fort77'
c f77 -c nom_du_fichier_contenant_la_fonction_main.f
c f77 -c H7etoile.f
c f77 nom_du_fichier_contenant_la_fonction_main.o H7etoile.o
c Fonctions exterieures appelees:
c Auteur: Pierre Lafaye de Micheaux
c Date: 15/02/2001
c Fin-Commentaires
      SUBROUTINE H7etoi (n,x)
      INTEGER n,i
      DOUBLE PRECISION x(n)
      DO 10, i=1,n
         x(i)=103.844*(x(i)**7)-167.749*(x(i)**5)+
     $           76.2494*(x(i)**3)-8.47215*x(i)
 10      CONTINUE
      RETURN
      END
```

Programme H8etoile.f

```
c Debut-Commentaires
c Nom de la sous-routine: H8etoi
c Entrees:
c n (entier), longueur du vecteur x
c x (double), un vecteur de n nombres
c Sorties: void
c Description:
c Calcul de H8etoile(x), H8etoile etant le huitieme polynome de Legendre modifie
c Modifie le vecteur x dans le programme principal, en le remplacant par H8etoile(x)
c Utilisation dans une fonction main:
c-------------------------------------------
c       PROGRAM main
c       INTEGER n, i
cc       longueur du vecteur x, a modifier dans le programme principal
c       PARAMETER(n=3)
cc       initialisation de x, y et xavant
c       DOUBLE PRECISION x(n), y(n), xavant(n)
cc       affectation de valeurs a x
c       x(1)=1
c       x(2)=2
c       x(3)=3
cc       On sauvegarde les valeurs de x dans xavant
c       DO 10, i=1,n
c       xavant(i)=x(i)
c 10    CONTINUE
cc       calcul de H8etoile(x), remplace x par H8etoile(x)
c       CALL H8etoi(n,x)
cc       affecte a y le resultat
c       DO 20, i=1,n
c       y(i)=x(i)
```

```
c 20     CONTINUE
cc        remet les bonnes valeurs dans x
cc        affecte a y le resultat
c        DO 30, i=1,n
c        x(i)=xavant(i)
c 30     CONTINUE
cc        Affichage des valeurs de y=H8etoile(x)
c        WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de y'
c        WRITE(UNIT=6,FMT=*) y
cc        Affichage des valeurs de x
c        WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de x'
c        WRITE(UNIT=6,FMT=*) x
c        END
c-------------------------------------------
c Instructions de compilation: alias f77='fort77'
c f77 -c nom_du_fichier_contenant_la_fonction_main.f
c f77 -c H8etoile.f
c f77 nom_du_fichier_contenant_la_fonction_main.o H8etoile.o
c Fonctions exterieures appelees:
c Auteur: Pierre Lafaye de Micheaux
c Date: 15/02/2001
c Fin-Commentaires
      SUBROUTINE H8etoi (n,x)
      INTEGER n,i
      DOUBLE PRECISION x(n)
      DO 10, i=1,n
         x(i)=260.07*(x(i)**8)-450.179*(x(i)**6)+
     $           247.184*(x(i)**4)-38.7299*(x(i)**2)-1.11233
 10      CONTINUE
      RETURN
      END
```

Programme H9etoile.f

```
c Debut-Commentaires
c Nom de la sous-routine: H9etoi
c Entrees:
c n (entier), longueur du vecteur x
c x (double), un vecteur de n nombres
c Sorties: void
c Description:
c Calcul de H9etoile(x), H9etoile etant le neuvieme polynome de Legendre modifie
c Modifie le vecteur x dans le programme principal, en le remplacant par H9etoile(x)
c Utilisation dans une fonction main:
c-------------------------------------------
c        PROGRAM main
c        INTEGER n, i
cc        longueur du vecteur x, a modifier dans le programme principal
c        PARAMETER(n=3)
cc        initialisation de x, y et xavant
c        DOUBLE PRECISION x(n), y(n), xavant(n)
cc        affectation de valeurs a x
c        x(1)=1
c        x(2)=2
c        x(3)=3
cc        On sauvegarde les valeurs de x dans xavant
c        DO 10, i=1,n
c        xavant(i)=x(i)
c 10     CONTINUE
cc        calcul de H9etoile(x), remplace x par H9etoile(x)
c        CALL H9etoi(n,x)
cc        affecte a y le resultat
c        DO 20, i=1,n
c        y(i)=x(i)
c 20     CONTINUE
```

```
cc       remet les bonnes valeurs dans x
cc       affecte a y le resultat
c       DO 30, i=1,n
c       x(i)=xavant(i)
c 30    CONTINUE
cc       Affichage des valeurs de y=H9etoile(x)
c       WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de y'
c       WRITE(UNIT=6,FMT=*) y
cc       Affichage des valeurs de x
c       WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de x'
c       WRITE(UNIT=6,FMT=*) x
c       END
c-------------------------------------------
c Instructions de compilation: alias f77='fort77'
c f77 -c nom_du_fichier_contenant_la_fonction_main.f
c f77 -c H9etoile.f
c f77 nom_du_fichier_contenant_la_fonction_main.o H9etoile.o
c Fonctions exterieures appelees:
c Auteur: Pierre Lafaye de Micheaux
c Date: 15/02/2001
c Fin-Commentaires
      SUBROUTINE H9etoi (n,x)
      INTEGER n,i
      DOUBLE PRECISION x(n)
      DO 10, i=1,n
         x(i)=413.925*(x(i)**9)-876.547*(x(i)**7)+
     $           613.583*(x(i)**5)-157.329*(x(i)**3)+10.727*x(i)
 10      CONTINUE
      RETURN
      END
```

Programme H10etoile.f

```
c Debut-Commentaires
c Nom de la sous-routine: H10eto
c Entrees:
c n (entier), longueur du vecteur x
c x (double), un vecteur de n nombres
c Sorties: void
c Description:
c Calcul de H10etoile(x), H10etoile etant le dixieme polynome de Legendre modifie
c Modifie le vecteur x dans le programme principal, en le remplacant par H10etoile(x)
c Utilisation dans une fonction main:
c-------------------------------------------
c       PROGRAM main
c       INTEGER n, i
cc       longueur du vecteur x, a modifier dans le programme principal
c       PARAMETER(n=3)
cc       initialisation de x, y et xavant
c       DOUBLE PRECISION x(n), y(n), xavant(n)
cc       affectation de valeurs a x
c       x(1)=1
c       x(2)=2
c       x(3)=3
cc       On sauvegarde les valeurs de x dans xavant
c       DO 10, i=1,n
c       xavant(i)=x(i)
c 10    CONTINUE
cc       calcul de H10etoile(x), remplace x par H10etoile(x)
c       CALL H10eto(n,x)
cc       affecte a y le resultat
c       DO 20, i=1,n
c       y(i)=x(i)
c 20    CONTINUE
cc       remet les bonnes valeurs dans x
```


```
cc        affecte a y le resultat
c        DO 30, i=1,n
c        x(i)=xavant(i)
c 30     CONTINUE
cc        Affichage des valeurs de y=H10etoile(x)
c        WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de y'
c        WRITE(UNIT=6,FMT=*) y
cc        Affichage des valeurs de x
c        WRITE(UNIT=6,FMT=*) 'Affichage des valeurs de x'
c        WRITE(UNIT=6,FMT=*) x
c        END
c-------------------------------------------
c Instructions de compilation: alias f77='fort77'
c f77 -c nom_du_fichier_contenant_la_fonction_main.f
c f77 -c H10etoile.f
c f77 nom_du_fichier_contenant_la_fonction_main.o H10etoile.o
c Fonctions exterieures appelees:
c Auteur: Pierre Lafaye de Micheaux
c Date: 15/02/2001
c Fin-Commentaires
      SUBROUTINE H10eto (n,x)
      INTEGER n,i
      DOUBLE PRECISION x(n)
      DO 10, i=1,n
         x(i)=994.507*(x(i)**10)-2250.43*(x(i)**8)+
     $           1782.83*(x(i)**6)-569.924*(x(i)**4)+67.5436*(x(i)**2)
     $           -3.58201
 10      CONTINUE
      RETURN
      END
```

A.12. LES AUTRES PETITS PROGRAMMES UTILES

Programme max.f

```
c Debut-Commentaires
c Nom de la fonction: maxi
c Entrees:
c n (entier), longueur du vecteur x
c x (double), x est le vecteur des points
c dont on veut calculer le maximum).
c Sorties: double maxim
c Description:
c Cette fonction calcule le maximum d'un vecteur
c Utilisation dans une fonction main:
c-------------------------------------------
c        PROGRAM main
c        INTEGER n
c        DOUBLE PRECISION maxim, maxi
cc        longueur du vecteur x, a modifier dans le programme principal
c        PARAMETER(n=3)
cc        initialisation de x
c        DOUBLE PRECISION x(n)
cc        affectation de valeurs a x
c        x(1)=1
c        x(2)=2
c        x(3)=3
c        maxim=maxi(n,x)
cc        Affichage du maximum de x
c        WRITE(UNIT=6,FMT=*) 'Le maximum de x est'
c        WRITE(UNIT=6,FMT=*) maxim
c        END
c-------------------------------------------
c Instructions de compilation: alias f77='fort77'
c f77 -c nom_du_fichier_contenant_la_fonction_main.f
```

```
c f77 -c max.f
c f77 nom_du_fichier_contenant_la_fonction_main.o max.o
c Fonctions exterieures appelees:
c Auteur: Pierre Lafaye de Micheaux
c Date: 15/02/2001
c Fin-Commentaires
      DOUBLE PRECISION FUNCTION maxi (n,x)
      INTEGER n,i
      DOUBLE PRECISION x(n)
      maxi=x(1)
      DO 10, i=2,n
         maxi=dmax1(maxi,x(i))
 10      CONTINUE
      RETURN
      END
```

Programme mean.f

```
c Debut-Commentaires
c Nom de la fonction: meanp
c Entrees:
c n (entier), longueur du vecteur x
c x (double), vecteur des points
c dont on veut calculer la moyenne
c Sorties: double moyenn
c Description:
c Cette fonction calcule la moyenne d'un vecteur
c Utilisation dans une fonction main:
c-----------------------------------------
c       PROGRAM main
c       INTEGER n
c       DOUBLE PRECISION moyenn, meanp
cc        longueur du vecteur x, a modifier dans le programme principal
c       PARAMETER(n=3)
cc        initialisation de x
c       DOUBLE PRECISION x(n)
cc        affectation de valeurs a x
c       x(1)=1
c       x(2)=2
c       x(3)=3
c       moyenn=meanp(n,x)
cc        Affichage de la moyenne de x
c       WRITE(UNIT=6,FMT=*) 'La moyenne de x est'
c       WRITE(UNIT=6,FMT=*) moyenn
c       END
c-----------------------------------------
c Instructions de compilation: alias f77='fort77'
c f77 -c nom_du_fichier_contenant_la_fonction_main.f
c f77 -c mean.f
c f77 nom_du_fichier_contenant_la_fonction_main.o mean.o
c Fonctions exterieures appelees:
c Auteur: Pierre Lafaye de Micheaux
c Date: 15/02/2001
c Fin-Commentaires
      DOUBLE PRECISION FUNCTION meanp (n,x)
      INTEGER n,i
      DOUBLE PRECISION x(n)
      meanp=DBLE(0.0)
      DO 10, i=1,n
         meanp=meanp+x(i)
 10      CONTINUE
      meanp=meanp/DBLE(n)
      RETURN
      END
```

Programme min.f

```
c Debut-Commentaires
c Nom de la fonction: mini
c Entrees:
c n (entier), longueur du vecteur x
c x (double), vecteur des points
c dont on veut calculer le minimum
c Sorties: double minim
c Description:
c Cette fonction calcule le minimum d'un vecteur
c Utilisation dans une fonction main:
c-------------------------------------------
c       PROGRAM main
c       INTEGER n
c       DOUBLE PRECISION minim, mini
cc        longueur du vecteur x, a modifier dans le programme principal
c       PARAMETER(n=3)
cc        initialisation de x
c       DOUBLE PRECISION x(n)
cc        affectation de valeurs a x
c       x(1)=1
c       x(2)=2
c       x(3)=3
c       minim=mini(n,x)
cc        Affichage du minimum de x
c       WRITE(UNIT=6,FMT=*) 'Le minimum de x est'
c       WRITE(UNIT=6,FMT=*) minim
c       END
c-------------------------------------------
c Instructions de compilation: alias f77='fort77'
c f77 -c nom_du_fichier_contenant_la_fonction_main.f
c f77 -c min.f
c f77 nom_du_fichier_contenant_la_fonction_main.o min.o
c Fonctions exterieures appelees:
c Auteur: Pierre Lafaye de Micheaux
c Date: 15/02/2001
c Fin-Commentaires
      DOUBLE PRECISION FUNCTION mini (n,x)
      INTEGER n,i
      DOUBLE PRECISION x(n)
      mini=x(1)
      DO 10, i=2,n
         mini=dmin1(mini,x(i))
 10      CONTINUE
      RETURN
      END
```

Programme var.f

```
c Debut-Commentaires
c Nom de la fonction: var
c Entrees:
c n (entier), longueur du vecteur x
c x (double), vecteur des points
c dont on veut calculer la variance
c Sorties: double varian
c Description:
c Cette fonction calcule la variance d'un vecteur
c Utilisation dans une fonction main:
c-------------------------------------------
c       PROGRAM main
c       INTEGER n
c       DOUBLE PRECISION varian, var
cc        longueur du vecteur x, a modifier dans le programme principal
c       PARAMETER(n=3)
```

```
cc          initialisation de x
c         DOUBLE PRECISION x(n)
cc          affectation de valeurs a x
c         x(1)=1
c         x(2)=2
c         x(3)=3
c         varian=var(n,x)
cc          Affichage de la variance de x
c         WRITE(UNIT=6,FMT=*) 'La variance de x est'
c         WRITE(UNIT=6,FMT=*) varian
c         END
c-------------------------------------------
c Instructions de compilation: alias f77='fort77'
c f77 -c nom_du_fichier_contenant_la_fonction_main.f
c f77 -c var.f
c f77 nom_du_fichier_contenant_la_fonction_main.o var.o
c Fonctions exterieures appelees:
c mean
c Auteur: Pierre Lafaye de Micheaux
c Date: 15/02/2001
c Fin-Commentaires
      DOUBLE PRECISION FUNCTION var (n,x)
      INTEGER n,i
      DOUBLE PRECISION x(n),res
      res=DBLE(0.0)
      DO 10, i=1,n
         res=res+x(i)
 10      CONTINUE
      res=res/DBLE(n)
      var=DBLE(0.0)
      DO 20, i=1,n
         var=var+(x(i)-res)**2
 20      CONTINUE
      var=var/DBLE(n)
      RETURN
      END
```

A.13. LE PROGRAMME DU CALCUL DES QUANTILES DU TEST DE BROCKWELL ET DAVIS

Programme statBD.f

```
c     Ce programme calcule le quantile du test de BD
      PROGRAM main
c     On calcule la stat de BD
      INTEGER i, nT, IFAIL, nbcle, val
      PARAMETER(nT=50, nbcle=50000)
      DOUBLE PRECISION Z(nT), BDA(nT), BDb(nT), BDc(nT), BDd(nT), statBD
      DOUBLE PRECISION sumbd, sumc, sumd2, alpha, QBDcal, stanBD(nbcle)
      EXTERNAL G05CBF, G05FDF, M01CAF, qnorm
      alpha=DBLE(0.05)
      CALL G05CBF(0)
      DO 90, j=1,nbcle
      CALL G05FDF(0.0D0,1.0D0,nT,Z)
      IFAIL=0
      CALL M01CAF(Z, 1, nT, 'Ascending', IFAIL)
      DO 20, i=1,nT
         BDa(i)=Z(i)
 20   CONTINUE
      DO 30, i=1,nT
         BDb(i)=BDa(i)
 30    CONTINUE
      DO 40, i=1,nT
         BDc(i)=(BDb(i))**2
```

```
40      CONTINUE
      DO 50, i=1,nT
         BDd(i)=(i-0.375)/(nT+0.125)
50      CONTINUE
      CALL qnorm(nT,BDd,0.0D0,1.0D0)
      sumbd=0
      sumc=0
      sumd2=0
      DO 60, i=1,nT
         sumbd=sumbd+(BDb(i)*BDd(i))
60      CONTINUE
      DO 70, i=1,nT
         sumc=sumc+BDc(i)
70      CONTINUE
      DO 80, i=1,nT
         sumd2=sumd2+(BDd(i))**2
80      CONTINUE
         statBD=(sumbd**2)/(sumc*sumd2)
         WRITE(6,*) j
         stanBD(j)=statBD
90    CONTINUE
      val=IDInt(nbcle*alpha)
      CALL M01CAF(stanBD, 1, nbcle, 'A', 0)
      QBDcal=stanBD(val)
      WRITE(6,*) QBDcal
      END
      INCLUDE "qnorm.f"
```

ANNEXE B

Les Programmes C++ du deuxième article

Vu l'ampleur des programmes (presque 4500 lignes de code), le listing des différents programmes n'est fourni que dans la version électronique de ce document sur le site Internet http://www.theses.umontreal.ca.
Tous ces programmes ont été utilisés sur un PC (gracieusement mis à disposition par Gilles Ducharme) disposant d'un processeur de 1 mégaHz et pourvu, par mes soins, du système d'exploitation Linux Redhat 7.2.

Ces programmes utilisent les librairies *newmat10* et *newran02* programmées par Robert Davies et téléchargeables (fichiers *newmat10.tar.gz* et *newran02.tar.gz*) sur son site Internet via l'URL http://www.robertnz.net/index.html.

Pour compiler les programmes, il faut placer tous les fichiers *.c dans un même répertoire avec les fichiers *libnewmat.a* et *libnewran.a* (obtenus à partir des fichiers source mentionnés ci-dessus), et utiliser la commande :

```
g++ -Wall -O main.c -L. -lnewmat -lnewran -o main
```

Un fichier nommé *main* est alors créé, et c'est ce fichier qu'il faut utiliser pour le calcul des quantiles obtenus par la méthode de Imhof (ou Davies, ou Deheuvels et Martynov) ainsi que pour le calcul des valeurs prises par la fonction de répartition de la loi limite de la statistique de Cramér-von Mises :

```
./main
```

Pour obtenir les quantiles empiriques dans le cas non sériel, il faut pour compiler utiliser la commande :

```
g++ -Wall -O test.c -L. -lnewmat -lnewran -o test
```

et pour lancer la simulation de Monte-Carlo, la commande :

```
./test > resultat.txt
```

Lorsque la simulation est terminée, les résultats se trouvent dans le fichier *resultat.txt*.

Un autre fichier nommé *manova.c* contient le programme permettant d'obtenir les résultats de la section 5 du deuxième article.
Il faut à nouveau utiliser les commandes

```
g++ -Wall -O manova.c -L. -lnewmat -lnewran -o manova
```

et

```
./manova
```

On obtient ainsi les puissances de notre test et de celui de Wilks dans le cadre considéré.

B.1. QUANTILES THÉORIQUES

Le fichier *zeta.c*

```
double zeta(int m)

{

  double somme;
  somme=0.0;
  int k;

  for (k=1;k<=10000;k=k+1) somme=somme+pow(k,-m);

  return(somme);

}
```

Le fichier *binarycode.c*

```
/* Debut-Commentaires

Nom de la fonction: binarycode

Entrees: i entier, L entier, *y pointeur

Sorties: void

Description:
Traduit en binaire sur L bits l'entier i.
Les resultats sont recuperes via le pointeur *y

Utilisation dans une fonction main:
-----------------------------------

#include <iostream.h>
#include<math.h>

int main()

{

  void binarycode(int i,int L, int *y);

  int *y;
  int i, L;
  int j;

  i=39;
  L=8;
  y=new int [L];

  binarycode(i,L,y);
```

```
  if (*y >=0) {
  cout << "Affichage de la version binaire de i="<<i<<" exprimee sur L="<<L<<" bits \n
      ";
  for (j=0;j<L;j=j+1) cout << *(y+j) << " ";
  }

return(0);

}
```

```
Instructions de compilation:
g++ -Wall -O nom_du_fichier_contenant_la_fonction_main.cc

Fonctions exterieures appelees:


Auteur: Pierre Lafaye de Micheaux

Date: 24/08/2002

Fin-Commentaires */


void binarycode(int i,int L, int *y)
 {

   int n, j;
   int R, q, *x;

   x=new int [L];
   q=i;
   n=0;

 do
   {
     R=q-2*(int)floor(q/2.0);
     q=(int)floor(q/2.0);
     for (j=L-1; j>0; j=j-1) *(x+j)=*(x+j-1);
     *(x+0)=R;
     n=n+1;
   }
 while (q>0);

 if (n==L)
   for (j=0;j<L;j=j+1) *(y+j)=*(x+j);
 else if (L-n>0)
    {
    for (j=0;j<L-n;j=j+1) *(y+j)=0;
    for (j=L-n;j<L;j=j+1) *(y+j)=*(x+n-L+j);
    }
 else {cout << "Il faut prendre L plus grand.\n";*(y+0)=L-n;}



 }
```

Le fichier *combn.c*

```
void combn(int **combmat, int n, int m)
{

  /*
       DESCRIPTION:
            Generate all combinations of the elements of seq(n) taken m at a time.
```


```
      REFERENCE:
            Nijenhuis, A. and Wilf, H.S. (1978) Combinatorial Algorithms for
            Computers and Calculators. NY: Academic Press.
*/

int i, j, e, h, nmmp1, mp1;

Matrix a(m,1);
for (i=1;i<=m;i=i+1) a(i,1)=i;

if((n < m)|(n==m)) cout <<"Attention: n <= m";
e=0;
h=m;



for (i=1;i<=m;i=i+1) combmat[0][i-1]=i;

i=2;
nmmp1=n - m + 1;
mp1=m + 1;
       while(a(1,1) != nmmp1) {

                if(e < n - h) {
                  h=1;
                  e=(int)a(m,1);


                  a(m - h + 1,1)=e + 1;

                  for (j=1;j<=m;j=j+1) combmat[i-1][j-1]=(int)a(j,1);



                i=i+1;

                }
                else {
                  h=h + 1;
                  e=(int)a(mp1 - h,1);

                  for (j=1;j<=h;j=j+1) a(m - h + j,1)=e + j;

                for (j=1;j<=m;j=j+1) combmat[i-1][j-1]=(int)a(j,1);



                i=i + 1;


                        }

       }

}
```

Le fichier *cornish.c*

```
/* Debut-Commentaires

Nom de la fonction: valpmultiv

Entrees: X double, cum matrice des cumulants
```

```
Sorties: D DiagonalMatrix

Description:
CALCUL DES QUANTILES PAR LA METHODE DE CORNISH–FISHER
On cherche yp tel que F(yp)=1−p
Soit X tel que Phi(X)=1−p ou Phi est la fonction de repartition d'une N(0,1)
cum est le vecteur des cumulants, il doit etre de longueur NORD+2 ou NORD est l'ordre
     de l'expansion souhaite


Utilisation dans une fonction main:
```

```
#include <iostream.h>
//#include<math.h>
#define WANT_STREAM                  // include.h will get stream fns
#define WANT_MATH                    // include.h will get math fns
                                     // newmatap.h will get include.h
#include "newmatap.h"                // need matrix applications

#include "newmatio.h"                // need matrix output routines

#ifdef use_namespace
using namespace NEWMAT;              // access NEWMAT namespace
#endif

#include "cornish.c"

int main()

{

   double cornish(double X, Matrix cum);
   int NORD, i;
   double X, resul;

  NORD=10;

   Matrix cum(NORD+2,1);


   //      Definition de X
  X=2.3;

   //      Definition des cumulants
  cum(1,1)=1.0;
  cum(2,1)=2.0;
   for (i=3;i<=NORD+2;i=i+1)     cum(i,1)=i;

   resul=cornish(X,cum);



   cout << setprecision(15)  << resul << "\n";


}
```

```
Instructions de compilation:
g++ −Wall −O nom_du_fichier_contenant_la_fonction_main.cc −L. −lnewmat
```


```
Fonctions exterieures appelees:
CF, et aussi la librairie libnewmat.a et les fichiers qui vont avec cette librairie:
newmatap.h, newmatio.h, newmat.h, include.h, boolean.h, myexcept.h

Auteur: Pierre Lafaye de Micheaux

Date: 24/08/2002

Fin-Commentaires */



double cornish(double X, Matrix cum)

{

  //Algorithm AS 269 comme dans Lee et Lin (1992) Appl. Statist.

  Matrix CF(int NORD, double X, Matrix AC, int IFAULT);

  int IFAULT, i, NORD;

  double somme, yp;

  NORD=cum.Nrows()-2;

  Matrix AC(NORD,1), DEL(NORD,1);

  IFAULT=0;

  //    Definition des cumulants ajustes
  for (i=3;i<=NORD+2;i=i+1)      AC(i-2,1)=cum(i,1)/(pow(sqrt(cum(2,1)),i));


  DEL=CF(NORD, X, AC, IFAULT);

  somme=X;

//    DEL represente les "adjustements"
  for (i=1;i<=NORD;i=i+1)  somme=somme+DEL(i,1);


  yp=cum(1,1)+sqrt(cum(2,1))*somme;

  return(yp);

}


Matrix CF(int NORD, double X, Matrix AC, int IFAULT)

{



  //    ALGORITHM AS 269 APPL.STATIST. (1992), VOL.41, NO.1

  //    Calculates the Cornish-Fisher adjustment to the normal deviate.
```


```
//      Local variables

Matrix DEL(NORD,1), A(NORD,1), H(3*NORD,1), P(3 * NORD * (NORD+1)/2,1), D(NORD,1);
int J, JA, JAL, JB, JBL, K, L;
double AA, BC, CC, DD, FAC, LIMIT, ONE, ZERO;

LIMIT=3.719017274;
ONE=1.0;
ZERO=0.0;


//      Check input arguments

IFAULT=0;
if (NORD>18)
  IFAULT=1;
else  if ((X<-LIMIT) || (X>LIMIT))
  IFAULT = 2;

if (IFAULT!=0) exit(0);

//      Compute the adjusted cumulants

CC = -ONE;
for (J=1;J<=NORD;J=J+1)
  {
    A(J,1) = CC * AC(J,1) / ((J+1) * (J+2));
    CC = -CC;

  }


//      Compute the Hermite polynomial values.

H(1,1) = -X;
H(2,1) = X * X - ONE;
for (J=3;J<=3*NORD;J=J+1) H(J,1) = - (X * H(J-1,1) + (J-1) * H(J-2,1));


//      Clear the polynomial array.

for (J=1;J<=3 * NORD * (NORD+1)/2;J=J+1) P(J,1) = ZERO;

D(1,1) = - A(1,1) * H(2,1);
DEL(1,1) = D(1,1);
P(1,1) = D(1,1);
P(3,1) = A(1,1);
JA = (int)ZERO;
FAC = ONE;

//      Main loop

for (J=2;J<=NORD;J=J+1)

  {

    //      Initialize.

    FAC = FAC * J;
    JA = JA + 3 * (J-1);
    JB = JA;
    BC = ONE;

    //      Calculate coefficients of Hermite polynomials.
```


```
      for (K=1;K<=J-1;K=K+1)

        {

          DD = BC * D(K,1);
          AA = BC * A(K,1);
          JB = JB - 3 * (J - K);

      for (L=1;L<=3 * (J - K);L=L+1)

        {
          JBL = JB + L;
          JAL = JA + L;
          P(JAL+1,1) = P(JAL+1,1) + DD * P(JBL,1);
          P(JAL+K+2,1) = P(JAL+K+2,1) + AA * P(JBL,1);
        }
      BC = BC * (J - K) / K;
        }
      P(JA+J+2,1) = P(JA+J+2,1) + A(J,1);


      //      Calculate the adjustments.

      D(J,1) = ZERO;
      for (L=2;L<=3 * J;L=L+1)          D(J,1) = D(J,1) - P(JA+L,1) * H(L-1,1);


      P(JA+1,1) = D(J,1);
      DEL(J,1) = D(J,1) / FAC;




    }

        return(DEL);
}
```

Le fichier *cumul.c*

```
/* Debut-Commentaires

Nom de la fonction: cum

Entrees: m entier ordre du cumulant, b double facteur de lissage,
q entier dimension de l'espace

Sorties: double

Description:
Calcule le cumulant d'ordre m, pour b et q fixes

Utilisation dans une fonction main:
-----------------------------------


#include <iostream.h>
#include<math.h>
#include "cum.c"

int main()

{

  double g1(double alpha, double beta, double gamma, double b);
  double g2(double beta, double b);
  double g3(double beta, double gamma, double b);
```

```
  double f(int m, double alpha, double beta, double gamma, double b, int q);
  double cum(int m, double b, int q);

  double b, val;
  int q, m;

  b=0.7;
  q=3;
  m=5;


  val=cum(m,b,q);

  cout << "Affichage du cumulant\n";

  cout << val;

  cout << "\n";


return(0);

}
```

```
Instructions de compilation:
g++ -Wall -O nom_du_fichier_contenant_la_fonction_main.cc

Fonctions exterieures appelees:
g1, g2, g3, f, fact

Auteur: Pierre Lafaye de Micheaux

Date: 23/08/2002

Fin-Commentaires */

double cum(int m, double b, int q)

{

  double f(int m, double alpha, double beta, double gamma, double b, int q);
  int fact(int n);
  double resul;

  resul=pow(2,(m-1))*fact(m-1)*(f(m,-0.5,-0.5,1.0,b,q) - f(m,-0.5,-0.5,0.0,b,q));

  return(resul);

}


double g1(double alpha, double beta, double gamma, double b)

{

  double resul;

  resul=alpha+gamma*gamma*(b*b/(2+2*b*b-4*b*b*beta));

  return(resul);

}
```

```
double g2(double beta, double b)

{

  double resul;

  resul=-0.5+b*b/(2+2*b*b-4*b*b*beta);

  return(resul);

}

double g3(double beta, double gamma, double b)

{

  double resul;

  resul=2*gamma*(b*b/(2+2*b*b-4*b*b*beta));


  return(resul);

}

double f(int m, double alpha, double beta, double gamma, double b, int q)

{

  double resul;

  if (m==1) resul=pow(1-2*b*b*(alpha+beta+gamma),-q/2.0);

    else resul=pow(1+b*b-2*b*b*beta,-q/2.0)*(f(m-1,g1(alpha,beta,gamma,b),g2(beta,b),
        g3(beta,gamma,b),b,q)-f(m-1,g1(alpha,beta,gamma,b),-0.5,0.0,b,q));

  return(resul);


}


int fact(int n)
    {
    if (n>1) return (n * fact(n-1));
    else return(1);
    }
```

Le fichier *deheuvels.c*

```
double beta(double u, Matrix valp, Matrix multiplicity, double a)

        {

          int i, n;
          double res;

          res=1.0;
          n=valp.Nrows();

          Matrix valpu2(n,1);

          for (i=1;i<=n;i=i+1)

            {
```

```
            valpu2(i,1)=pow(pow(1-a*u*valp(i,1),2.0)+pow(valp(i,1)*u,2.0),0.25*
                multiplicity(i,1));


            res=res*valpu2(i,1);

            }

          return(res);

        }

double zeta(double u, Matrix valp, Matrix multiplicity, double x, double a)

        {

          int i, n;
          double res;


          n=valp.Nrows();

          Matrix hratanvalpu(n,1);

          for (i=1;i<=n;i=i+1)

            {

              hratanvalpu(i,1)=multiplicity(i,1)*atan(u/(1.0/valp(i,1)-a*u));

            }
              res=0.5*hratanvalpu.Sum()-0.5*x*u;

              return(res);

        }

double devfunc(double u, Matrix valp, Matrix multiplicity, double x, double a)

        {
          //c'est la fonction sous l'integrale dans (1.9) de Martynov (1975)
          double zeta(double u, Matrix valp, Matrix multiplicity, double x);
          double beta(double u, Matrix valp, Matrix multiplicity);

          double res;

          res=(exp(-a*u*x*0.5)*sin(zeta(u,valp,multiplicity,x,a)))/(u*beta(u,valp,
              multiplicity,a));


          return(res);

        }

double integraldev(double x, Matrix valp, Matrix multiplicity, double U, int n, double
    res, double a)

        {

          //J'utilise la routine trapzd de numerical recipes, section 4.2 elementary
              algorithms

          double devfunc(double u, Matrix valp, Matrix multiplicity, double x, double
              a);
          int it, tnm, j;
```


```
        double del, y, sum;
        //0.5*(b-a)*(FUNC(a)+FUNC(b))
if (n==1) { res=0.5*U*((0.5*(SP(valp,multiplicity)).Sum()-0.5*x)+devfunc(U,valp,
    multiplicity,x,a)); } else

      {

        it=(int)pow(2,n-2);

        tnm=it;

        del=U/tnm;

        y=0.5*del;

        sum=0.0;

        for (j=1;j<=it;j=j+1) {sum=sum+devfunc(y,valp,multiplicity,x,a);

        y=y+del;
      }

        res=0.5*(res+U*sum/tnm);


      }
 return(res);

      }

double deheuvels(double x, Matrix valp, Matrix multiplicity, double U, int N, double a
    )

      {

        // U est un terme de troncation dans le calcul de l'integrale (1.9) (voir
            Imhof pp. 422-423)

        double integraldev(double x, Matrix valp, Matrix multiplicity, double U, int
             n, double res, double a);
        double PI, resul, res;
        int i;



        PI=3.14159265358979311 60;
        res=0;

      for (i=1;i<=N;i=i+1)

      {

        res=integraldev(x,valp,multiplicity,U,i,res,a);

      }


      resul=1.0-atan(1.0/a)/PI-res/PI;

      return(resul);

      }
```

Le fichier *gauher.c*

```
#define pi 3.141592653589793115998
```

```
Matrix gauher(double b, int N, double prec, int MAXIT)

{

/*

Renvoie une matrice de taille Nx2
La premiere colonne contient les abscisses.
La derniere colonne contient les poids.

*/



  double EPS, pim4, z, var, p1, p2, p3, pp, z1;
  int its, i, j;

  z=0.0;
  pp=0.0;
  EPS=prec;
  pim4=1.0/(pow(pi,0.25));

  Matrix x(N,1);
  x=0.0;

  Matrix w(N,1);
  w=x;

  int m;
  m=(int)((N+1)/2.0);

  for (i=1;i<=m;i=i+1)

    {

      if (i==1) { z=(sqrt(2*N+1)-1.85575*(pow((2*N+1),(-1.0/6.0))));}

      else     { if (i==2) {z=(z-1.14*(pow(N,0.426))/z);}

      else     { if (i==3) {z=(1.86*z-0.86*x(1,1));}

      else     { if (i==4) {z=(1.91*z-0.91*x(2,1));}

      else {z=(2*z-x(i-2,1));}

      }}}

      var=0.0;

      for (its=1;its<=MAXIT;its=its+1)

        {

          p1=pim4;

          p2=0.0;

          for (j=1;j<=N;j=j+1) {

            p3=p2;

            p2=p1;
```

```
            p1=z*(sqrt(2.0/j))*p2-sqrt((j-1.0)/(double)j)*p3;
          }
          pp=sqrt(2*N)*p2;
          z1=z;
          z=z1-p1/pp;
          if (fabs(z-z1) < EPS) {var=1.0;
                                 break;}
        }
      if (var==0.0) cout << "Trop d'iterations\n";
      x(i,1)=z;
      x(N+1-i,1)=-z;
      w(i,1)=2.0/(pp*pp);
      w(N+1-i,1)=w(i,1);
    }
  x=sqrt(2.0)*b*x;
  w=w/sqrt(pi);
  Matrix poidsetabscisses(N,2);
  poidsetabscisses.SubMatrix(1,N,1,1)=x;
  poidsetabscisses.SubMatrix(1,N,2,2)=w;

  return(poidsetabscisses);
       }
```

Le fichier *Imhof.c*

```
/* Debut-Commentaires

Nom de la fonction: probQsupx

Entrees: x double, valp Matrix, U double, N entier

Sorties: resul double

Description:
Calcule les quantiles a partir des valeurs propres, en inversant la fonction
    caracteristique suivant
la methode d'Imhof.

Utilisation dans une fonction main:
-----------------------------------


#include <iostream.h>
#include<math.h>
#define WANT_STREAM                  // include.h will get stream fns
#define WANT_MATH                    // include.h will get math fns
                                     // newmatap.h will get include.h
```

```
#include "newmatap.h"                // need matrix applications

#include "newmatio.h"                // need matrix output routines

#ifdef use_namespace
using namespace NEWMAT;              // access NEWMAT namespace
#endif

#include "Imhof.c"

int main()

{

  double rho(double u, Matrix valp);
  double theta(double u, Matrix valp, double x);
  double imhoffunc(double u, Matrix valp, double x);
  double integral(double x, Matrix valp, double U, int n, double res);
  double probQsupx(double x, Matrix valp, double U, int N);

  int  N, q;

  q=1;

  Matrix valp(2*(2*q+(int)pow(2,q)+2*(int)pow(q,2)-2*q),1);

  double x, U, resultat;

  valp(1,1)=0.8;
  valp(2,1)=0.7;
  valp(3,1)=0.6;
  valp(4,1)=0.5;
  valp(5,1)=0.4;
  valp(6,1)=0.3;
  valp(7,1)=0.2;
  valp(8,1)=0.1;

  x=0.05;

  U=0.1;

  N=10;


  resultat=probQsupx(x,valp,U,N);


  cout << resultat << "\n";

return(0);

}
```

```
Instructions de compilation:
g++ -Wall -O nom_du_fichier_contenant_la_fonction_main.cc

Fonctions exterieures appelees:
rho, theta, imhoffunc, integral

Auteur: Pierre Lafaye de Micheaux
```


```
Date: 25/08/2002

Fin-Commentaires */



double rho(double u, Matrix valp, Matrix multiplicity)

        {

          int i, n;
          double res;

          res=1.0;
          n=valp.Nrows();

          Matrix valpu2(n,1);

          for (i=1;i<=n;i=i+1)

            {

            valpu2(i,1)=pow(1+pow(valp(i,1)*u,2),0.25*multiplicity(i,1));
            res=res*valpu2(i,1);

            }

          return(res);

        }

double theta(double u, Matrix valp, Matrix multiplicity, double x)

        {

          int i, n;
          double res;

          n=valp.Nrows();

          Matrix hratanvalpu(n,1);

          for (i=1;i<=n;i=i+1)

            {

              hratanvalpu(i,1)= multiplicity(i,1)*atan(valp(i,1)*u);

            }
              res=0.5*hratanvalpu.Sum()-0.5*x*u;

              return(res);

        }

double imhoffunc(double u, Matrix valp, Matrix multiplicity, double x)

        {
          //c'est la fonction sous l'integrale dans (3.2) de Imhof (1961)
          double theta(double u, Matrix valp, Matrix multiplicity, double x);
          double rho(double u, Matrix valp, Matrix multiplicity);

          double res;
```

```
          res=(sin(theta(u,valp,multiplicity,x)))/(u*rho(u,valp,multiplicity));


          return(res);

        }

double integral(double x, Matrix valp, Matrix multiplicity, double U, int n, double
    res)

        {

          //J'utilise la routine trapzd de numerical recipes, section 4.2 elementary
              algorithms

          double imhoffunc(double u, Matrix valp, Matrix multiplicity, double x);
          int it, tnm, j;
          double del, y, sum;
          //0.5*(b-a)*(FUNC(a)+FUNC(b))
if (n==1) { res=0.5*U*((0.5*(SP(valp,multiplicity)).Sum()-0.5*x)+imhoffunc(U,valp,
    multiplicity,x)); } else

        {

          it=(int)pow(2,n-2);

          tnm=it;

          del=U/tnm;

          y=0.5*del;

          sum=0.0;

          for (j=1;j<=it;j=j+1) {sum=sum+imhoffunc(y,valp,multiplicity,x);

          y=y+del;
        }

          res=0.5*(res+U*sum/tnm);


        }
 return(res);

        }

double probQsupx(double x, Matrix valp, Matrix multiplicity, double U, int N)

        {

          // U est un terme de troncation dans le calcul de l'integrale (3.2) (voir
              Imhof pp. 422-423)

          double integral(double x, Matrix valp, Matrix multiplicity, double U, int n
              , double res);
          double PI, resul, res;
          int i;

          PI=3.1415926535897931160;
          res=0;

        for (i=1;i<=N;i=i+1)
```

```
        {
          res=integral(x,valp,multiplicity,U,i,res);
        }
        resul=0.5+res/PI;
        return(resul);
        }
```

Le fichier *main.c*

```
// DEBUT DES DEFINITIONS POUR POUVOIR UTILISER LES LIBRAIRIES
//-------------------------------------------------------------

#include <string.h>
#define WANT_STREAM                  // include.h will get stream fns
#define WANT_MATH                    // include.h will get math fns
                                     // newmatap.h will get include.h
#include "./include/newmatap.h"              // need matrix applications

#include "./include/newmatio.h"              // need matrix output routines

#ifdef use_namespace
using namespace NEWMAT;              // access NEWMAT namespace
#endif

#ifdef use_namespace
using namespace NEWRAN;
#endif

#include "./include/newran.h" // Pour la generation de nombres aleatoires


//debut definitions pour qfdavies

#define UseDouble 0             /* all floating point double */

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <malloc.h>
#include <setjmp.h>

#define TRUE  1
#define FALSE 0
typedef int BOOL;

#ifdef UseDouble
   typedef double real;
#else
   typedef float real;
#endif

#define pi 3.141592653589793115998
#define log28 .0866  /*  log(2.0) / 8.0  */

static real sigsq, lmax, lmin, mean, c;
static double intl, ersm;
static int count, r, lim;  static BOOL ndtsrt, fail;
static int *n,*th; static real *lb,*nc;
static jmp_buf env;


static real exp1(real x)              /* to avoid underflows  */
```

```
{ return x < -50.0 ? 0.0 : exp(x); }

//fin definitions pour qfdavies

//INCLUSION DES FICHIERS CONTENANT LES FONCTIONS DONT JE ME SERS
//-------------------------------------------------------------

#include "Imhof.c"
#include "deheuvels.c"
#include "valpmultiv.c"
#include "binarycode.c"
#include "cornish.c"
#include "cumul.c"
#include "qnorm.c"
#include "combn.c"
#include "nCm.c"
#include "permrep.c"
#include "zeta.c"
#include "rank.c"
#include "qfdavies.c"

//-------------------------------------------------------------
// DEBUT DU PROGRAMME
//-------------------------------------------------------------

//Pour compiler le programme taper:
//g++ -Wall -O main.c -L. -lnewmat -lnewran -o main

int main()

{


  char info[1];

  cout << "Voulez-vous de l'information sur l'utilisation de ce programme? (o)ui n(on
      ): ";
  //  cin >> info;
  if ((strcmp(info,"o")==0) | (strcmp(info,"y")==0)) {

  cout << "\nSTRATEGIE POUR UTILISER CE PROGRAMME:";
  cout << "\nElle se deroule en 3 etapes.";
  cout << "\n1)";
  cout << "\nLe calcul des cumulants est exact. Il faut donc commencer par trouver le
      bon jeu de parametres qui va mener aux bonnes valeurs propres.";
  cout << "\nPour cela, on se base sur le fait que la somme des valeurs propres dans
      le cas CardA=1 est egale au premier cumulant dans le cas CardA=1.";
  cout << "\nOn commence par essayer de calculer les valeurs propres par une formule
      de cubature. Le nombre de valeurs propres obtenues est alors fixe ";
  cout << "\nsauf quand q=1 ou on peut entrer le nombre d'abscisses de la cubature.";
  cout << "\nSi la cubature ne donne pas des resultats satisfaisants, on passe alors a
      la methode de Monte-Carlo. Il faut alors tatonner jusqu'a obtenir ";
  cout << "\nla bonne taille d'echantillon.";
  cout << "\n2)";
  cout << "\nQuand, on a trouve les bonne valeurs propres pour le cas CardA=1, on
      cherche a obtenir les bonnes valeurs propres pour le cas CardA>1.";
  cout << "\nPour cela, on se base encore sur le fait que la somme des valeurs propres
      pour le cas CardA>1 est egale au premier cumulant dans le cas CardA>1.";
  cout << "\nLe critere critarret1 permet de limiter le nombre de valeurs propres
      calculees pour le cas CardA>1, en eliminant d'emblee les valeurs propres ";
  cout << "\ndu cas CardA=1 qui vont mener a des valeurs propres du cas CardA>1 trop
      petites. Il faut donc ajuster ce critere pour ne pas obtenir trop ";
  cout << "\nde valeurs propres mais assez pour que leur somme soit egale au premier
      cumulant du cas CardA>1.";
  cout << "\n3)  ";
```

```
cout << "\nQuand␣on␣est␣confiant␣sur␣la␣qualite␣des␣valeurs␣propres,␣il␣faut␣s'
    interroger␣sur␣la␣qualite␣des␣quantiles␣et␣des␣distributions.";
cout << "\nPour␣cela,␣il␣faut␣jouer␣sur␣le␣parametre␣NORD␣qui␣donne␣l'ordre␣d'
    expansion␣de␣Cornish-Fisher␣de␣facon␣a␣ce␣que␣si␣on␣calcule␣le␣quantile␣";
cout << "\nd'ordre␣1-alpha␣et␣que␣l'on␣utilise␣ce␣quantile␣dans␣la␣procedure␣de␣
    Imhof␣ou␣de␣Davies␣on␣retrouve␣bien␣1-alpha.";
cout << "\nOn␣peut␣aussi␣essayer␣d'utiliser␣l'une␣des␣regles␣du␣pouce␣suivante:";
cout << "\nLe␣chiffre␣choisi␣ci-dessous␣(NORD=5)␣l'a␣ete␣en␣utilisant␣la␣methode␣
    suivante:";
cout << "\nj'ai␣trace␣dans␣Splus␣plot(1:10,c(quantQ(NORD=1),...,quantQ(NORD=10)),
    type=\"l\")␣et␣";
cout << "\nj'ai␣choisi␣la␣valeur␣de␣NORD␣juste␣avant␣l'oscillation␣de␣la␣courbe.";
cout << "\nLa␣valeur␣NORD=5␣(pour␣Remillard␣c'est␣NORD=4)␣a␣ete␣trouvee␣pour␣b=0.1␣
    et␣q=3,␣il␣faudra␣peut-etre␣changer␣";
cout << "\npour␣d'autres␣valeurs␣de␣q␣et/ou␣b.";
cout << "\nNORD␣est␣le␣nombre␣de␣cumulants␣ajustes,␣comme␣dans␣Lee␣et␣Lin␣(1992),␣il
    ␣y␣a␣en␣fait␣NORD+2␣cumulants␣utilises.";
cout << "\nIl␣y␣a␣aussi␣une␣regle␣du␣pouce␣dans␣Lee␣et␣Lin␣(1992)␣p236␣pour␣savoir␣
    combien␣en␣prendre␣...␣a␣voir/";
cout << "\nUn␣autre␣critere␣a␣ajuster␣est␣critarret2␣qui␣permet␣de␣limiter␣le␣nombre
    ␣de␣valeurs␣propres␣qui␣vont␣rentrer␣dans␣les␣procedures␣";
cout << "\nde␣Imhof␣et␣de␣Davies.␣S'il␣y␣a␣trop␣de␣valeurs␣propres,␣ces␣procedures␣
    ne␣fonctionnent␣pas␣bien.␣Il␣faut␣toutefois␣que␣la␣somme␣";
cout << "\ndes␣valeurs␣propres␣selectionnees␣qui␣vont␣rester␣soit␣encore␣egale␣ou␣
    proche␣au␣premier␣cumulant␣du␣cas␣CardA>1.\n";

cout << "\nRemarque:";
cout << "\nQuand␣b␣est␣grand,␣utiliser␣plutot␣Monte-Carlo.";
cout << "\nQuand␣b␣est␣petit,␣utiliser␣plutot␣la␣cubature.\n";

cout << "\nVoir␣aussi␣si␣je␣trouve␣des␣formules␣de␣cubature␣avec␣plus␣de␣points␣pour
    ␣le␣cas␣ou␣b␣est␣>␣0.1";

}



// Declaration des fonctions utilisees
double cornish(double X, Matrix cumul);
double cum(int m, double b, int q);
//  double rho(double u, Matrix valp, Matrix multiplicity);
//  double theta(double u, Matrix valp, Matrix multiplicity, double x);
//  double imhoffunc(double u, Matrix valp, Matrix multiplicity, double x);
//  double integral(double x, Matrix valp, Matrix multiplicity, double U, int n,
    double res);
double probQsupx(double x, Matrix valp, Matrix multiplicity, double U, int N);
double qnorm(double P, int IFAULT);
int nCm(int n, int m);
double zeta(int m);
void  rank(Matrix RV,char ORDER[2],int *IRANK);
real qfdavies(real *lb, real *nc, int *n, int r, real sigma, real c,int lim, real
    acc, real *trace, int *ifault);
void combn(int **combmat, int n, int m);
void permrep(int **permrepmat, int *indices, int n, int k);
int minimum(int x, int y);
double deheuvels(double u, Matrix valp, Matrix multiplicity, double x, double a);


//Declaration des variables
DiagonalMatrix valpmultiv(double b, int q, int choix, int N, double prec, int MAXIT
    );
Random::Set(0.46875);
int  N, Nbre, q, i, j, k, choix, NORD, IFAULT, CardA;
double b, alpha;
```


```
double x, z,  X, U, resultat, quantQA, quantRA;
int **permrepmat;
int *indicesperm;
int count;
double critarret1;
int Narret;
int MAXIT;
double prec;
int NORDm2;


//Choix du parametre de lissage b, marche bien si b<=0.1
//Choix de la taille q de l'espace: marche pour q<=8
//Choix du Cardinal de A
//Choix du niveau
//Et affichage des parametres choisis
cout << "\n\n
    ----------------------------------------------------------------------------\n
    ";
cout << "|";
cout << "\n| Taille de l'espace considere: q=";cin >> q;
cout << "| Cardinal de A: "; cin >> CardA ;
cout << "| Parametre de lissage choisi: b="; cin >> b ;
cout << "| Niveau choisi: alpha="; cin >> alpha;
cout << "| Nombre de cumulants (>=3) choisis pour l'approximation de Cornish-Fisher
    : "; cin >> NORDm2;
cout << "|";
cout << "\n
    ----------------------------------------------------------------------------\n
    ";

//-------------------------------------------------------------------------
//CALCUL DES CUMULANTS
//-------------------------------------------------------------------------


if (NORDm2<3) {cout << "\n\n Le nombre de cumulants doit etre superieur a 3\n\n";
    return(1);}

//Nombre de cumulants ajustes utilises dans l'approximation de Cornish-Fisher
NORD=NORDm2-2;


Matrix cumulQ(NORD+2,1);
Matrix cumulRA(NORD+2,1);


//      Definition de X : Phi^(-1)(1-alpha)
IFAULT=0;
X=qnorm(1-alpha,IFAULT);


cout << "\n\nCalcul des cumulants et application de Cornish-Fisher\n";
cout << "-------------------------------------------------------";

// Calcul des cumulants de Q=\sum_{i=1}^{\infty}\lambda_iZ_i^2 pour Card(A)=1
cumulQ(1,1)=cum(1,b,q);
cumulQ(2,1)=cum(2,b,q);
for (i=3;i<=NORD+2;i=i+1)     cumulQ(i,1)=cum(i,b,q);

//Affichage des cumulants pour pouvoir utiliser la stopping rule de Lee (1992)
cout << "\n\nCumulants ajustes d'ordre impair :";
for (i=1;i<=NORD;i=i+2) cout << setprecision(10) << cumulQ(i+2,1)/pow(sqrt(cumulQ
    (2,1)),i+2) << " ";
cout << "\nCumulants ajustes d'ordre pair :";
```

```
for (i=2;i<=NORD;i=i+2) cout << setprecision(10) << cumulQ(i+2,1)/pow(sqrt(cumulQ
    (2,1)),i+2) << " ";

//Cumulants de Remillard de xi_1 pour CardA=1: commenter cette ligne pour se baser
    sur mes cumulants
//      for (i=1;i<=NORD+2;i=i+1)  cumulQ(i,1)=pow(2,i-1)*fact(i-1)*pow(zeta(2*i
    ),1)/pow(pi,2*i);


cout << "\n\n Le premier cumulant de Q pour CardA=1 vaut:\n ";
cout << setprecision(8) << cumulQ(1,1) << "\n";

// Calcul des cumulants de Q=\sum_{i=1}^{\infty}\lambda_iZ_i^2 pour Card(A)
    quelconque
//CumulQA(m)=[cumulQ(m)]^|A|*[2^(m-1)*(m-1)!]^(1-|A|)
Matrix cumulQA(NORD+2,1);
for (i=1;i<=NORD+2;i=i+1)
  {

    cumulQA(i,1)=pow(cumulQ(i,1),CardA)*pow(pow(2,i-1)*fact(i-1),1-CardA);

  }

// Calcul des cumulants de R=(Q-cumulQA(1))/sqrt(cumulQA(2)) pour Card(A) quelconque
    :
// cumulRA(1)=0  cumulRA(2)=1
// cumulRA(m)=cumulQA(m)/([cumulQA(2)]^(m/2))   pour tout m>2

cout << "\n Le premier cumulant de Q pour |A|=" << CardA << " vaut:\n ";
cout << setprecision(8) << cumulQA(1,1) << "\n";


//-------------------------------------------------------------------------
//CALCUL DES QUANTILES PAR LA METHODE DE CORNISH FISHER
//-------------------------------------------------------------------------


//Calcul du quantile quantQA par la methode de Cornish Fisher base sur NORD+2
    cumulants
quantQA=cornish(X,cumulQA);
cout << "\n Le quantile P[Q<?]=" << 1-alpha << " correspondant a X=Phi^(-1)("<<1-
    alpha<<")=" << X << " pour |A|=" << CardA << " vaut:\n ";
cout << setprecision(8)  << quantQA << "\n";

//Calcul du quantile quantRA par la methode de Cornish Fisher base sur NORD+2
    cumulants
quantRA=(quantQA-cumulQA(1,1))/(sqrt(cumulQA(2,1)));
cout << "\n Le quantile P[R<?]=" << 1-alpha << " correspondant a X=Phi^(-1)("<<1-
    alpha<<")=" << X << " pour |A|=" << CardA << " vaut:\n ";
cout << setprecision(8)  << quantRA << "\n";



//-------------------------------------------------------------------------
//CALCUL DES VALEURS PROPRES PAR CUBATURE OU PAR MONTE-CARLO
//-------------------------------------------------------------------------


cout << "\nCalcul des valeurs propres\n";
cout << "-------------------------------";
cout << "\n Entrez votre choix: Methode par cubature (1) ou Methode de Monte-Carlo
    (2): ";
cin >>choix;
```

```
prec=0.0;
MAXIT=0;

// On fixe le nombre de valeurs propres suivant la methode choisie
if (choix==1)
  {
    if (q==1)
      {
        cout << "␣Nombre␣de␣valeurs␣propres␣souhaitees:␣";cin >> N;
        cout << "␣Precision␣souhaitee:␣";cin >> prec;
        cout << "␣Nombre␣maximum␣d'iteration␣souhaite:␣";cin >> MAXIT;

      }

    //mettre ici la bonne valeur suivant la formule de cubature choisie dans
        valpmultiv.c
    //j'ai choisi dans valpmultiv.c la cubature 1 qui me donne les meilleurs
        resultats
    else if (q==2) N=44;

    else if (q>=3) N=(int)pow(2,q+1)+4*(int)pow(q,2);

  }



if (choix==2) {
   cout << "␣␣Taille␣N␣de␣l'echantillon␣de␣Monte-carlo␣souhaitee␣:␣";
   cin >> N;
}


//Calcul des valeurs propres pour CardA=1
//-----------------------------------------
DiagonalMatrix D(N);
Matrix valp(N,1);

D=valpmultiv(b,q,choix,N,prec,MAXIT);

for (i=1;i<=N;i=i+1)
  {
    valp(i,1)=D(i);

  }


//Calcul des valeurs propres de Remillard: commenter cette ligne pour calculer mes
    valeurs propres
//       for (i=1;i<=N;i=i+1) valp(i,1)=pow(i*pi,-2);

if (choix==1) cout <<"\n␣Nombre␣de␣valeurs␣propres␣calculees␣pour␣CardA=1,␣par␣
    cubature:␣" << N << "\n";
if (choix==2) cout <<"\n␣Nombre␣de␣valeurs␣propres␣calculees␣pour␣CardA=1,␣par␣Monte
    -Carlo:␣" << N << "\n";


// Pour verifier si la somme des valeurs propres est bien egale au premier cumulant
cout << "\n␣La␣somme␣des␣valeurs␣propres␣pour␣CardA=1␣(qui␣doit␣etre␣egale␣au␣
    premier␣cumulant␣pour␣CardA=1)␣vaut:\n␣";
cout << setprecision(8) << valp.Sum() << "\n";


//Calcul des valeurs propres pour CardA quelconque
//-------------------------------------------------
```

```
//Calcul de toutes les permutations avec repetitions de 1:n qui sont donnees sous
    forme des combinaisons avec repetitions et du nombre de fois
//que chaque combinaison doit etre utilisee (indicesperm). Cela va permettre d'
    obtenir les valeurs propres pour CardA quelconque comme produits de valeurs
//propres pour CardA=1
//Il y a n^k permutations avec repetitions de k objets pris parmi n.
//Ces n^k permutations peuvent se repartir en nCm(n+k-1,k)=(n+k-1)!/((n-1)!k!)
    combinaisons avec repetitions ou chaque combinaison doit etre repetee
//(N_1+N_2+...+N_n)!/(N_1!N_2!...N_n!) (en changeant l'ordre des elements), ou N_1
    est le nombre de 1 dans dans la combinaison consideree,
//N_2 le nombre de 2, ..., N_n le nombre de n.

//Puisqu'un double est code sur 4 octets, on est limite dans la taille de par la
    borne conservatrice: n<exp[Log(x/4)/k] ou x est la RAM disponible en octets.

//ordre d'arret des valeurs propres: astuce qui permet d'eliminer une partie des
    valeurs propres de CardA=1 pouvant
//ammener a des valeurs propres trop petites (< a 10^-critarret1) de CardA
    quelconque.
Narret=N;
cout << "\ncritarret1 (un entier positif) ? : "; cin >> critarret1;
for (i=1;i<=N;i=i+1) {if (pow(valp(1,1),CardA-1)*valp(i,1)<pow(10,-critarret1)) {
    Narret=i;break;}}

cout << "\n Nombre de valeurs propres de CardA=1 selectionnees (par le critere: max{
    p} tel que lambda_1^(|A|-1)*lambda_p<" << pow(10,-critarret1) << ") \n";
cout << " pour constituer les valeurs propres de CardA>1: " << Narret << " sur " <<
    N << ",\n";
cout << " ce qui donne " << Narret << "^" << CardA << "=" << (int)pow(Narret,CardA
    ) << " valeurs propres en comptant leur ";
cout << "ordre de multiplicite dans le cas CardA>1.\n";

N=Narret;
count=nCm(N+CardA-1,CardA);

//permrepmat est un tableau a count lignes et CardA colonnes
permrepmat=new int*[count];
for (i=0;i<count;i=i+1) permrepmat[i]=new int[CardA];
indicesperm=new int[count];

//Calcul des repetitions et des indicesperm donnant le nombre de foisqu'il faut
    utiliser chaque combinaison pour obtenir toutes les permutations
//avec repetitions
permrep(permrepmat,indicesperm,N,CardA);


//Calcul des valeurs propres (pas forcement distinctes) dans le cas CardA quelconque
     avec leur ordre de multiplicite.
//Une valeur propre avec ordre de multiplicite n1+n2 pourrait etre comptee deux fois
    , la premiere avec ordre de multiplicite n1 et la deuxieme
//avec ordre de multiplicite n2
Matrix valpA(count,1);
valpA=1.0;
Matrix multiplicity(count,1);

for (i=1;i<=count;i=i+1) {

  multiplicity(i,1)=indicesperm[i-1];

    for (k=1;k<=CardA;k=k+1) {

    valpA(i,1)=valpA(i,1)*valp(permrepmat[i-1][k-1],1);

    }
}
```

```
//On a plus besoin de ces tableaux donc on les decharge de la memoire.
delete permrepmat;
delete indicesperm;

//Pour verifier que la somme des valeurs propres est bien egale au premier cumulant
    pour CardA quelconque
//SP est le produit terme a terme
cout << "\n La somme des valeurs propres pour |A|=" << CardA << " (qui doit etre
    egale au premier cumulant pour |A|=" << CardA << ") vaut:\n ";
cout << setprecision(8) << (SP(valpA,multiplicity)).Sum() << "\n";



//On classe les valeurs propres par ordre decroissant et on reordonne les
    multiplicite en consequence
Matrix indicescl(count,1);
int *IRANK;
IRANK=new int[count];

rank(valpA,"d",IRANK);

for (i=1;i<=count;i=i+1) indicescl(*(IRANK+i-1),1)=multiplicity(i,1);

//On a plus besoin de ce tableau donc on le decharge de la memoire
delete IRANK;

SortDescending(valpA);


N=count;

//Classement et comptage des valeurs propres distinctes, et calcul du veritable
    ordre de multiplicite
Matrix valdistemp(N,1);
Matrix indicestemp(N,1);
indicestemp=0.0;
valdistemp(1,1)=valpA(1,1);

j=1;

//ce critere permet de selectionner les valeurs propres distinctes (dans le sens que
     leur difference en valeur absolue est inferieure a 10^(-critarret2))
//et de laisser tomber les plus petites a 10^(-precis) pres. Elles seront encore
    classees par ordre decroissant.
//ce critere permet de limiter le nombre de valeurs propres qui vont rentrer dans la
     procedure de Imhof et de Davies
int critarret2;
cout << "\ncritarret2 (un entier positif) ? :"; cin >> critarret2;


for (i=1;i<=N-1;i=i+1)

  {

     if (valpA(i,1)>=pow(10.0,-critarret2))

       {

          if (fabs(valpA(i,1)-valpA(i+1,1))>=pow(10.0,-critarret2))
            {
              valdistemp(j,1)=valpA(i,1);
              indicestemp(j,1)=indicestemp(j,1)+indicescl(i,1);
```


```
            j=j+1;
          }

        if (fabs(valpA(i,1)-valpA(i+1,1))<pow(10.0,-critarret2)) indicestemp(j,1)=
            indicestemp(j,1)+indicescl(i,1);

      }


  }

//r represente le nombre de valeurs propres distinctes selectionnees
int r;
r=j-1;
Matrix valpdist(r,1);
Matrix indices(r,1);

for (i=1;i<=r;i=i+1)
  {
  valpdist(i,1)=valdistemp(i,1);
  indices(i,1)=indicestemp(i,1);
  }

cout << "\n\n Nombre de valeurs propres distinctes et superieures a " << pow(10.0,-
    critarret2) << " selectionnees: " << r << " sur " << (int)pow(Narret,CardA) <<
    "\n";
cout <<" Nombre total de valeurs propres en comptant leur ordre de multiplicite: "<<
    indices.Sum() << "\n";
cout << setprecision(8) << " Somme de ces valeurs propres: " << (SP(valpdist,indices
    )).Sum() << "\n";

/------------------------------------------------------------------------
 //APPLICATION DE LA METHODE DE DAVIES (1973,1980)
/------------------------------------------------------------------------

//Valeur z du quantile pour lequel on souhaite obtenir la probabilite P[R<z]
//avec R=(Q-cumulQA(1))/sqrt(cumulQA(2))
//On a P[R<=z]=P[Q<=z*sqrt(cumulQA(2))+cumulQA(1)]
//Si on met z=quantRA; on peut verifier que le calcul du quantile effectue plus haut
//et de la probabilite P[r<z] effectue plus bas donnent les memes resultats
z=quantRA;

//Valeur x du quantile pour lequel on souhaite obtenir la probabilite P[Q<x]
//Si on met x=quantQA; on peut verifier que le calcul du quantile effectue plus haut
//et de la probabilite P[Q<x] effectue plus bas donnent les memes resultats
x=quantQA;


 int lim; real c,sigma,acc;
 real *lambda, *noncen;
 int *n;
 real ansQ, ansR;
 int *fault; real *trace;

    sigma=0.0;
    lim=1000;
    acc=0.0001;

 lambda=new real[r];
 noncen=new real[r];
 trace=new real[7];
 n=new int[r];
 fault=new int[6];

 for ( i = 0; i<r; i++)
```

```
  {
    *(noncen+i)=0.0;
    *(n+i)=(int)indices(i+1,1);
    *(lambda+i)=valpdist(i+1,1);
  }

  cout << "\n La methode de Robert Davies (1973,1980) donne:\n";

  c=x;
  ansQ=qfdavies(lambda, noncen, n, r, sigma, c, lim, acc, trace, fault);

  cout << setprecision(8) << "       x: " << x  << "       P[Q<x]: " << ansQ << "\n";
  cout << setprecision(8) << "         FAULT: " << *fault << "\n";
  cout << setprecision(8) << "         trace: " << trace[0] << "   " << trace[1] <<
      "   " << trace[2] << "   " << trace[3] << "   " << trace[4] << "   " << trace
      [5] << "   " << trace[6] << "\n";

  c=z*sqrt(cumulQA(2,1))+cumulQA(1,1);
  ansR=qfdavies(lambda, noncen, n, r, sigma, c, lim, acc, trace, fault);

  cout << setprecision(8) << "\n       z: " << z << "       P[R<z]: " << ansR << "\n";
  cout << setprecision(8) << "         FAULT: " << *fault << "\n";
  cout << setprecision(8) << "         trace: " << trace[0] << "   " << trace[1] <<
      "   " << trace[2] << "   " << trace[3] << "   " << trace[4] << "   " << trace
      [5] << "   " << trace[6] << "\n";


  //on n'a plus besoin de ces tableaux donc on les decharge de la memoire
  delete lambda;
  delete noncen;

 /------------------------------------------------------------------------
  //APPLICATION DE LA METHODE DE IMHOF (1961)
 /------------------------------------------------------------------------


 //Borne d'intervalle d'integration dans la methode de Imhof, plus U est grand
 //plus c'est precis, mais si c'est trop grand ca deconne
  U=trace[4];

//Nombre d'evaluations pour l'integrale pour la methode du trapeze dans numerical
    recipies
 Nbre=minimum((int)trace[1],15);

 //on n'a plus besoin de ce tableau donc on le decharge de la memoire
 delete trace;



 //On utilise la methode de Imhof
 resultat=1-probQsupx(x,valpdist,indices,U,Nbre);
 cout << "\n La methode de Imhof (1961) donne:\n";
 cout << setprecision(8) << "       x: " << x << "       P[Q<x] : " << resultat << "\n";

 //On utilise la methode de Imhof
 resultat=1-probQsupx(z*sqrt(cumulQA(2,1))+cumulQA(1,1),valpdist,indices,U,Nbre);
 cout << setprecision(8) << "       z: " << z << "       P[R<z] : " << resultat << "\n";

 double a;
 cout << "\nValeur de a dans la methode de Deheuvels ?: ";
 cin >> a;

 //On utilise la methode de Deheuvels (en fait Martynov 1975)
 resultat=deheuvels(x,valpdist,indices,U,Nbre,a);
```

```
  cout << "\n La methode de Deheuvels donne:\n";
  cout << setprecision(8) << "      x: " << x << "    P[Q<x] : " << resultat << "\n";

  //On utilise la methode de Deheuvels (en fait Martynov 1975)
  resultat=deheuvels(z*sqrt(cumulQA(2,1))+cumulQA(1,1),valpdist,indices,U,Nbre,a);

  cout << "\n La methode de Deheuvels donne:\n";
  cout << setprecision(8) << "      z: " << z << "    P[R<z] : " << resultat << "\n";

  cout << setprecision(8) << "        U: " << U << "    Nbre: " << Nbre << "\n";


return(0);

}

//-------------------------------------------------------------------------
// FIN DU PROGRAMME
//-------------------------------------------------------------------------


int minimum(int x, int y)

{
  int min;
  min=x;

  if (y<=x) min=y;

  return(min);

}
```

Le fichier *permrep.c*

```
void permrep(int **permrepmat, int *indices, int n, int k)

{

  /*
        Calcule toutes les nCm(n+k-1,k) combinaisons de k objets
        pris parmi n avec repetitions.
        et donne dans le vecteur indices le nombre de fois que chaque combinaison
        doit etre repetee (en changeant l'ordre des termes de la combinaison en
        question) pour obtenir toutes les n^k permutations avec repetitions.

  */

  int nCm(int n, int m);
  void combn(int **combmat, int n, int m);
  int i,j,l,count, produit, somme;
  count=nCm(n+k-1,k);



  combn(permrepmat,n+k-1,k);
  for (i=1;i<=count;i=i+1) {
    for (j=1;j<=k;j=j+1) {

      permrepmat[i-1][j-1]=permrepmat[i-1][j-1]-(j-1);

        }
  }
```

```
//indices<-rep(0,count)
 Matrix Nkind(count,1);
 Matrix N(n+1,1);


for (j=1;j<=count;j=j+1) {

  Nkind=0;
  N=0;

for (i=1;i<=k;i=i+1) {

  if (N(permrepmat[j-1][i-1],1)==0) {Nkind(i,1)=permrepmat[j-1][i-1];} else Nkind(i
      ,1)=n+1;
  N(permrepmat[j-1][i-1],1)=N(permrepmat[j-1][i-1],1)+1;
}

 produit=1;
 somme=0;
for (l=1;l<=k;l=l+1) {
  produit=produit*fact((int)N((int)Nkind(l,1),1));
somme=somme+(int)N((int)Nkind(l,1),1);
}



 indices[j-1]=fact(somme)/produit;

}


}
```

Le fichier *qfc.c*

```
#define UseDouble 0             /* all floating point double */

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <malloc.h>
#include <setjmp.h>

#define TRUE  1
#define FALSE 0
typedef int BOOL;

#ifdef UseDouble
   typedef double real;
#else
   typedef float real;
#endif

#define pi 3.14159265358979
#define log28 .0866  /*  log(2.0) / 8.0  */

static real sigsq, lmax, lmin, mean, c;
static double intl, ersm;
static int count, r, lim;  static BOOL ndtsrt, fail;
static int *n,*th; static real *lb,*nc;
static jmp_buf env;
```

```
real qf(real*,real*,int*,int,real,real,int,real,real*,int*);

static real exp1(real x)                /* to avoid underflows  */
{ return x < -50.0 ? 0.0 : exp(x); }

void main(void)
{
   int r,lim; real c,sigma,acc;
   real lambda[100], noncen[100]; int n[100];
   real ans1; int fault; real trace[7]; int i;
l1 :
   puts("ENTER␣R,C,SIGMA,LIM,ACC\n");
   #ifdef UseDouble
      scanf("%d%lf%lf%d%lf",&r,&c,&sigma,&lim,&acc);
   #else
      scanf("%d%f%f%d%f",&r,&c,&sigma,&lim,&acc);
   #endif
   if (r<0) goto l2;
   puts("DF,LAMBDA,NON-CEN\n");
   for ( i = 0; i<r; i++)
   {
      #ifdef UseDouble
         scanf("%d%lf%lf",&n[i],&lambda[i],&noncen[i]);
      #else
         scanf("%d%f%f",&n[i],&lambda[i],&noncen[i]);
      #endif
   }
   ans1=qf(lambda, noncen, n, r, sigma, c, lim, acc, trace, &fault);
   #ifdef UseDouble
      printf("QF,FAULT=%10.6lf%15d\n\n",ans1,fault);
      for (i=0;i<7;i++) printf("%11.5lf",trace[i]);
   #else
      printf("QF,FAULT=%10.6f%15d\n\n",ans1,fault);
      for (i=0;i<7;i++) printf("%11.5f",trace[i]);
   #endif
   puts("\n\n");
   goto  l1;
l2 : ;
}


   static void counter(void)
   /*  count number of calls to errbd, truncation, cfe */
   {
      extern int count,lim;
      count = count + 1;
      if ( count > lim ) longjmp(env,1);
   }

   static real square(real x)  { return x*x; }

   static real cube(real x)  { return x*x*x; }

   static real  log1(real x, BOOL first)
   /* if (first) log(1 + x) ; else  log(1 + x) - x */
   {
      if (fabs(x) > 0.1)
      {
         return (first ? log(1.0 + x) : (log(1.0 + x) - x));
      }
      else
      {
         real s, s1, term, y, k;
         y = x / (2.0 + x);  term = 2.0 * cube(y);  k = 3.0;
         s = (first ? 2.0 : - x) * y;
```

```
        y = square(y);
        for (s1 = s + term / k; s1 != s; s1 = s + term / k)
        { k = k + 2.0; term = term * y; s = s1; }
        return s;
    }
}

static void order(void)
/* find order of absolute values of lb */
{
    int j, k; real lj;
    extern real *lb; extern int *th; extern int r; extern BOOL ndtsrt;
    for ( j=0; j<r; j++ )
    {
        lj = fabs(lb[j]);
        for (k = j-1; k>=0; k--)
        {
            if ( lj > fabs(lb[th[k]]) )  th[k + 1] = th[k];
            else goto l1;
        }
        k = -1;
    l1 :
        th[k + 1] = j;
    }
    ndtsrt = FALSE;
}


static real  errbd(real u, real* cx)
/*  find bound on tail probability using mgf, cutoff
    point returned to *cx */
{
    real sum1, lj, ncj, x, y, xconst; int j, nj;
    extern real sigsq,*lb,*nc; extern int *n; extern int r;
    counter();
    xconst = u * sigsq;  sum1 = u * xconst;  u = 2.0 * u;
    for (j=r-1; j>=0; j--)
    {
        nj = n[j]; lj = lb[j]; ncj = nc[j];
        x = u * lj; y = 1.0 - x;
        xconst = xconst + lj * (ncj / y + nj) / y;
        sum1 = sum1 + ncj * square(x / y)
            + nj * (square(x) / y + log1(-x, FALSE ));
    }
    *cx = xconst; return exp1(-0.5 * sum1);
}

static real  ctff(real accx, real* upn)
/*  find ctff so that p(qf > ctff) < accx  if (upn > 0,
     p(qf < ctff) < accx otherwise */
{
    real u1, u2, u, rb, xconst, c1, c2;
    extern real lmin,lmax,mean;
    u2 = *upn;   u1 = 0.0;  c1 = mean;
    rb = 2.0 * ((u2 > 0.0) ? lmax : lmin);
    for (u = u2 / (1.0 + u2 * rb); errbd(u, &c2) > accx;
        u = u2 / (1.0 + u2 * rb))
    {
        u1 = u2;  c1 = c2;  u2 = 2.0 * u2;
    }
    for (u = (c1 - mean) / (c2 - mean); u < 0.9;
        u = (c1 - mean) / (c2 - mean))
    {
        u = (u1 + u2) / 2.0;
        if (errbd(u / (1.0 + u * rb), &xconst) > accx)
```

```
            {  u1 = u; c1 = xconst;  }
         else
            {  u2 = u;  c2 = xconst; }
      }
      *upn = u2; return c2;
}

static real truncation(real u, real tausq)
/* bound integration error due to truncation at u */
{
      real sum1, sum2, prod1, prod2, prod3, lj, ncj,
         x, y, err1, err2;
      int j, nj, s;
      extern real sigsq,*lb,*nc; extern int *n; extern int r;

      counter();
      sum1  = 0.0; prod2 = 0.0;  prod3 = 0.0;  s = 0;
      sum2 = (sigsq + tausq) * square(u); prod1 = 2.0 * sum2;
      u = 2.0 * u;
      for (j=0; j<r; j++)
      {
         lj = lb[j];  ncj = nc[j]; nj = n[j];
         x = square(u * lj);
         sum1 = sum1 + ncj * x / (1.0 + x);
         if (x > 1.0)
         {
            prod2 = prod2 + nj * log(x);
            prod3 = prod3 + nj * log1(x, TRUE );
            s = s + nj;
         }
         else  prod1 = prod1 + nj * log1(x, TRUE );
      }
      sum1 = 0.5 * sum1;
      prod2 = prod1 + prod2;  prod3 = prod1 + prod3;
      x = exp1(-sum1 - 0.25 * prod2) / pi;
      y = exp1(-sum1 - 0.25 * prod3) / pi;
      err1 =  ( s  ==  0 )  ? 1.0 : x * 2.0 / s;
      err2 =  ( prod3 > 1.0 )  ? 2.5 * y : 1.0;
      if (err2 < err1) err1 = err2;
      x = 0.5 * sum2;
      err2 =  ( x  <=  y )  ? 1.0  : y / x;
      return  ( err1 < err2 )  ? err1  :  err2;
}

static void findu(real* utx, real accx)
/*  find u such that truncation(u) < accx and truncation(u / 1.2) > accx */
{
      real u, ut; int i;
      static real divis[]={2.0,1.4,1.2,1.1};
      ut = *utx; u = ut / 4.0;
      if ( truncation(u, 0.0) > accx )
      {
         for ( u = ut; truncation(u, 0.0) > accx; u = ut) ut = ut * 4.0;
      }
      else
      {
         ut = u;
         for ( u = u / 4.0; truncation(u, 0.0) <=  accx; u = u / 4.0 )
         ut = u;
      }
      for ( i=0;i<4;i++)
         { u = ut/divis[i]; if ( truncation(u, 0.0)  <=  accx )  ut = u; }
      *utx = ut;
}
```

```
static void integrate(int nterm, real interv, real tausq, BOOL mainx)
/*  carry out integration with nterm terms, at stepsize
   interv.  if (! mainx) multiply integrand by
      1.0-exp(-0.5*tausq*u^2) */
{
   real inpi, u, sum1, sum2, sum3, x, y, z;
   int k, j, nj;
   extern double intl,ersm; extern real sigsq,c;
   extern int *n; extern real *lb,*nc; extern int r;
   inpi = interv / pi;
   for ( k = nterm; k>=0; k--)
   {
      u = (k + 0.5) * interv;
      sum1 = - 2.0 * u * c;  sum2 = fabs(sum1);
      sum3 = - 0.5 * sigsq * square(u);
      for ( j = r-1; j>=0; j--)
      {
         nj = n[j];  x = 2.0 * lb[j] * u;  y = square(x);
         sum3 = sum3 - 0.25 * nj * log1(y, TRUE );
         y = nc[j] * x / (1.0 + y);
         z = nj * atan(x) + y;
         sum1 = sum1 + z;   sum2 = sum2 + fabs(z);
         sum3 = sum3 - 0.5 * x * y;
      }
      x = inpi * exp1(sum3) / u;
      if ( !  mainx )
      x = x * (1.0 - exp1(-0.5 * tausq * square(u)));
      sum1 = sin(0.5 * sum1) * x;  sum2 = 0.5 * sum2 * x;
      intl = intl + sum1; ersm = ersm + sum2;
   }
}

static real cfe(real x)
/*  coef of tausq in error when convergence factor of
   exp1(-0.5*tausq*u^2) is used when df is evaluated at x */
{
   real axl, axl1, axl2, sxl, sum1, lj; int j, k, t;
   extern BOOL ndtsrt,fail; extern int *th,*n; extern real *lb,*nc;
   extern int r;
   counter();
   if (ndtsrt) order();
   axl = fabs(x);  sxl = (x>0.0) ? 1.0 : -1.0;  sum1 = 0.0;
   for ( j = r-1; j>=0; j-- )
   { t = th[j];
      if ( lb[t] * sxl > 0.0 )
      {
         lj = fabs(lb[t]);
         axl1 = axl - lj * (n[t] + nc[t]);  axl2 = lj / log28;
         if ( axl1 > axl2 )  axl = axl1  ; else
         {
            if ( axl > axl2 )  axl = axl2;
            sum1 = (axl - axl1) / lj;
            for ( k = j-1; k>=0; k--)
            sum1 = sum1 + (n[th[k]] + nc[th[k]]);
            goto  l;
         }
      }
   }
l:
   if (sum1 > 100.0)
   { fail = TRUE; return 1.0; } else
   return pow(2.0,(sum1 / 4.0)) / (pi * square(axl));
}
```

```
real   qf(real* lb1, real* nc1, int* n1, int r1, real sigma, real c1,
   int lim1, real acc, real* trace, int* ifault)

/*  distribution function of a linear combination of non-central
   chi-squared random variables :

input:
   lb[j]            coefficient of j-th chi-squared variable
   nc[j]            non-centrality parameter
   n[j]             degrees of freedom
   j = 0, 2 ... r-1
   sigma            coefficient of standard normal variable
   c                point at which df is to be evaluated
   lim              maximum number of terms in integration
   acc              maximum error

output:
   ifault = 1       required accuracy NOT achieved
            2       round-off error possibly significant
            3       invalid parameters
            4       unable to locate integration parameters
            5       out of memory

   trace[0]         absolute sum
   trace[1]         total number of integration terms
   trace[2]         number of integrations
   trace[3]         integration interval in final integration
   trace[4]         truncation point in initial integration
   trace[5]         s.d. of initial convergence factor
   trace[6]         cycles to locate integration parameters     */

{
      int j, nj, nt, ntm;  real acc1, almx, xlim, xnt, xntm;
      real utx, tausq, sd, intv, intv1, x, up, un, d1, d2, lj, ncj;
      extern real sigsq, lmax, lmin, mean;
      extern double intl,ersm;
      extern int r,lim; extern real c;
      extern int *n,*th; extern real *lb,*nc;
      real qfval;
      static int rats[]={1,2,4,8};

      if (setjmp(env) != 0) { *ifault=4; goto endofproc; }
      r=r1; lim=lim1; c=c1;
      n=n1; lb=lb1; nc=nc1;
      for ( j = 0; j<7; j++ )  trace[j] = 0.0;
      *ifault = 0; count = 0;
      intl = 0.0; ersm = 0.0;
      qfval = -1.0; acc1 = acc; ndtsrt = TRUE;  fail = FALSE;
      xlim = (real)lim;
      th=(int*)malloc(r*(sizeof(int)));
      if (! th) { *ifault=5;  goto  endofproc; }

      /* find mean, sd, max and min of lb,
         check that parameter values are valid */
      sigsq = square(sigma); sd = sigsq;
      lmax = 0.0; lmin = 0.0; mean = 0.0;
      for (j=0; j<r; j++)
      {
         nj = n[j];  lj = lb[j];  ncj = nc[j];
         if ( nj < 0  ||  ncj < 0.0 ) { *ifault = 3;  goto  endofproc;  }
         sd  = sd  + square(lj) * (2 * nj + 4.0 * ncj);
         mean = mean + lj * (nj + ncj);
         if (lmax < lj) lmax = lj ; else if (lmin > lj) lmin = lj;
      }
```

```
   if ( sd == 0.0  )
   {  qfval = (c > 0.0) ? 1.0 : 0.0; goto  endofproc;  }
   if ( lmin == 0.0 && lmax == 0.0 && sigma == 0.0 )
      { *ifault = 3;  goto  endofproc;  }
   sd = sqrt(sd);
   almx = (lmax < - lmin) ? - lmin : lmax;

   /* starting values for findu, ctff */
   utx = 16.0 / sd;  up = 4.5 / sd;  un = - up;
   /* truncation point with no convergence factor */
   findu(&utx, .5 * acc1);
   /* does convergence factor help */
   if (c != 0.0  &&(almx > 0.07 * sd))
   {
      tausq = .25 * acc1 / cfe(c);
      if (fail) fail = FALSE ;
      else if (truncation(utx, tausq) < .2 * acc1)
      {
         sigsq = sigsq + tausq;
         findu(&utx, .25 * acc1);
         trace[5] = sqrt(tausq);
      }
   }
   trace[4] = utx;  acc1 = 0.5 * acc1;

   /* find RANGE of distribution, quit if outside this */
l1:
   d1 = ctff(acc1, &up) - c;
   if (d1 < 0.0) { qfval = 1.0; goto endofproc; }
   d2 = c - ctff(acc1, &un);
   if (d2 < 0.0) { qfval = 0.0; goto endofproc; }
   /* find integration interval */
   intv = 2.0 * pi / ((d1 > d2) ? d1 : d2);
   /* calculate number of terms required for main and
      auxillary integrations */
   xnt = utx / intv;  xntm = 3.0 / sqrt(acc1);
   if (xnt > xntm * 1.5)
   {
      /* parameters for auxillary integration */
      if (xntm > xlim) { *ifault = 1; goto endofproc; }
      ntm = (int)floor(xntm+0.5);
      intv1 = utx / ntm;  x = 2.0 * pi / intv1;
      if (x <= fabs(c)) goto l2;
      /* calculate convergence factor */
      tausq = .33 * acc1 / (1.1 * (cfe(c - x) + cfe(c + x)));
      if (fail) goto l2;
      acc1 = .67 * acc1;
      /* auxillary integration */
      integrate(ntm, intv1, tausq, FALSE );
      xlim = xlim - xntm;  sigsq = sigsq + tausq;
      trace[2] = trace[2] + 1; trace[1] = trace[1] + ntm + 1;
      /* find truncation point with new convergence factor */
      findu(&utx, .25 * acc1);  acc1 = 0.75 * acc1;
      goto l1;
   }

   /* main integration */
l2:
   trace[3] = intv;
   if (xnt > xlim) { *ifault = 1; goto endofproc; }
   nt = (int)floor(xnt+0.5);
   integrate(nt, intv, 0.0, TRUE );
   trace[2] = trace[2] + 1; trace[1] = trace[1] + nt + 1;
   qfval = 0.5 - intl;
   trace[0] = ersm;
```


```
      /* test whether round-off error could be significant
         allow for radix 8 or 16 machines */
      up=ersm; x = up + acc / 10.0;
      for (j=0;j<4;j++) { if (rats[j] * x == rats[j] * up) *ifault = 2; }

   endofproc :
      free((char*)th);
      trace[6] = (real)count;
      return qfval;
}
```

Le fichier *qfdavies.c*

```
   static void counter(void)
   /*  count number of calls to errbd, truncation, cfe */
   {
      extern int count,lim;
      count = count + 1;
      if ( count > lim ) longjmp(env,1);
   }

   static real square(real x)  { return x*x; }

   static real cube(real x)  { return x*x*x; }

   static real  log1(real x, BOOL first)
   /* if (first) log(1 + x) ; else  log(1 + x) - x */
   {
      if (fabs(x) > 0.1)
      {
         return (first ? log(1.0 + x) : (log(1.0 + x) - x));
      }
      else
      {
         real s, s1, term, y, k;
         y = x / (2.0 + x);  term = 2.0 * cube(y);  k = 3.0;
         s = (first ? 2.0 : - x) * y;
         y = square(y);
         for (s1 = s + term / k; s1 != s; s1 = s + term / k)
         { k = k + 2.0; term = term * y; s = s1; }
         return s;
      }
  }

  static void order(void)
  /* find order of absolute values of lb */
  {
      int j, k; real lj;
      extern real *lb; extern int *th; extern int r; extern BOOL ndtsrt;
      for ( j=0; j<r; j++ )
      {
         lj = fabs(lb[j]);
         for (k = j-1; k>=0; k--)
         {
            if ( lj > fabs(lb[th[k]]) )  th[k + 1] = th[k];
            else goto l1;
         }
         k = -1;
      l1 :
         th[k + 1] = j;
      }
      ndtsrt = FALSE;
   }
```

```
static real   errbd(real u, real* cx)
/*  find bound on tail probability using mgf, cutoff
   point returned to *cx */
{
   real sum1, lj, ncj, x, y, xconst; int j, nj;
   extern real sigsq,*lb,*nc; extern int *n; extern int r;
   counter();
   xconst = u * sigsq;  sum1 = u * xconst;  u = 2.0 * u;
   for (j=r-1; j>=0; j--)
   {
      nj = n[j]; lj = lb[j]; ncj = nc[j];
      x = u * lj; y = 1.0 - x;
      xconst = xconst + lj * (ncj / y + nj) / y;
      sum1 = sum1 + ncj * square(x / y)
         + nj * (square(x) / y + log1(-x, FALSE ));
   }
   *cx = xconst; return exp1(-0.5 * sum1);
}

static real  ctff(real accx, real* upn)
/*  find ctff so that p(qf > ctff) < accx  if (upn > 0,
    p(qf < ctff) < accx otherwise */
{
   real u1, u2, u, rb, xconst, c1, c2;
   extern real lmin,lmax,mean;
   u2 = *upn;   u1 = 0.0;  c1 = mean;
   rb = 2.0 * ((u2 > 0.0) ? lmax : lmin);
   for (u = u2 / (1.0 + u2 * rb); errbd(u, &c2) > accx;
      u = u2 / (1.0 + u2 * rb))
   {
      u1 = u2;  c1 = c2;  u2 = 2.0 * u2;
   }
   for (u = (c1 - mean) / (c2 - mean); u < 0.9;
      u = (c1 - mean) / (c2 - mean))
   {
      u = (u1 + u2) / 2.0;
      if (errbd(u / (1.0 + u * rb), &xconst) > accx)
         {  u1 = u; c1 = xconst;  }
      else
         {  u2 = u;  c2 = xconst; }
   }
   *upn = u2; return c2;
}

static real truncation(real u, real tausq)
/* bound integration error due to truncation at u */
{
   real sum1, sum2, prod1, prod2, prod3, lj, ncj,
      x, y, err1, err2;
   int j, nj, s;
   extern real sigsq,*lb,*nc; extern int *n; extern int r;

   counter();
   sum1  = 0.0; prod2 = 0.0;  prod3 = 0.0;  s = 0;
   sum2 = (sigsq + tausq) * square(u); prod1 = 2.0 * sum2;
   u = 2.0 * u;
   for (j=0; j<r; j++ )
   {
      lj = lb[j];  ncj = nc[j]; nj = n[j];
      x = square(u * lj);
      sum1 = sum1 + ncj * x / (1.0 + x);
      if (x > 1.0)
      {
         prod2 = prod2 + nj * log(x);
         prod3 = prod3 + nj * log1(x, TRUE );
```

```
            s = s + nj;
         }
         else  prod1 = prod1 + nj * log1(x, TRUE );
      }
      sum1 = 0.5 * sum1;
      prod2 = prod1 + prod2;  prod3 = prod1 + prod3;
      x = exp1(-sum1 - 0.25 * prod2) / pi;
      y = exp1(-sum1 - 0.25 * prod3) / pi;
      err1 =  ( s  ==  0 )  ? 1.0 : x * 2.0 / s;
      err2 =  ( prod3 > 1.0 )  ? 2.5 * y : 1.0;
      if (err2 < err1) err1 = err2;
      x = 0.5 * sum2;
      err2 =  ( x  <=  y )  ? 1.0  : y / x;
      return  ( err1 < err2 )  ? err1  :  err2;
}

static void findu(real* utx, real accx)
/*  find u such that truncation(u) < accx and truncation(u / 1.2) > accx */
{
      real u, ut; int i;
      static real divis[]={2.0,1.4,1.2,1.1};
      ut = *utx; u = ut / 4.0;
      if ( truncation(u, 0.0) > accx )
      {
         for ( u = ut; truncation(u, 0.0) > accx; u = ut) ut = ut * 4.0;
      }
      else
      {
         ut = u;
         for ( u = u / 4.0; truncation(u, 0.0) <=  accx; u = u / 4.0 )
         ut = u;
      }
      for ( i=0;i<4;i++)
         { u = ut/divis[i]; if ( truncation(u, 0.0)  <=  accx )  ut = u; }
      *utx = ut;
}


static void integrate(int nterm, real interv, real tausq, BOOL mainx)
/*  carry out integration with nterm terms, at stepsize
      interv.  if (! mainx) multiply integrand by
         1.0-exp(-0.5*tausq*u^2) */
{
      real inpi, u, sum1, sum2, sum3, x, y, z;
      int k, j, nj;
      extern double intl,ersm; extern real sigsq,c;
      extern int *n; extern real *lb,*nc; extern int r;
      inpi = interv / pi;
      for ( k = nterm; k>=0; k--)
      {
         u = (k + 0.5) * interv;
         sum1 = - 2.0 * u * c;  sum2 = fabs(sum1);
         sum3 = - 0.5 * sigsq * square(u);
         for ( j = r-1; j>=0; j--)
         {
            nj = n[j];  x = 2.0 * lb[j] * u;  y = square(x);
            sum3 = sum3 - 0.25 * nj * log1(y, TRUE );
            y = nc[j] * x / (1.0 + y);
            z = nj * atan(x) + y;
            sum1 = sum1 + z;   sum2 = sum2 + fabs(z);
            sum3 = sum3 - 0.5 * x * y;
         }
         x = inpi * exp1(sum3) / u;
         if ( !  mainx )
         x = x * (1.0 - exp1(-0.5 * tausq * square(u)));
```

```
        sum1 = sin(0.5 * sum1) * x;  sum2 = 0.5 * sum2 * x;
        intl = intl + sum1; ersm = ersm + sum2;
      }
   }

   static real cfe(real x)
   /*  coef of tausq in error when convergence factor of
      exp1(-0.5*tausq*u^2) is used when df is evaluated at x */
   {
      real axl, axl1, axl2, sxl, sum1, lj; int j, k, t;
      extern BOOL ndtsrt,fail; extern int *th,*n; extern real *lb,*nc;
      extern int r;
      counter();
      if (ndtsrt) order();
      axl = fabs(x);  sxl = (x>0.0) ? 1.0 : -1.0;  sum1 = 0.0;
      for ( j = r-1; j>=0; j--)
      { t = th[j];
         if ( lb[t] * sxl > 0.0 )
         {
            lj = fabs(lb[t]);
            axl1 = axl - lj * (n[t] + nc[t]);  axl2 = lj / log28;
            if ( axl1 > axl2 )  axl = axl1  ; else
            {
               if ( axl > axl2 )  axl = axl2;
               sum1 = (axl - axl1) / lj;
               for ( k = j-1; k>=0; k--)
               sum1 = sum1 + (n[th[k]] + nc[th[k]]);
               goto  l;
            }
         }
      }
   l:
      if (sum1 > 100.0)
      { fail = TRUE; return 1.0; } else
      return pow(2.0,(sum1 / 4.0)) / (pi * square(axl));
   }


real   qfdavies(real* lb1, real* nc1, int* n1, int r1, real sigma, real c1,int lim1,
    real acc, real* trace, int* ifault)

/*  distribution function of a linear combination of non-central
   chi-squared random variables :

input:
   lb[j]            coefficient of j-th chi-squared variable
   nc[j]            non-centrality parameter
   n[j]             degrees of freedom
   j = 0, 2 ... r-1
   sigma            coefficient of standard normal variable
   c                point at which df is to be evaluated
   lim              maximum number of terms in integration
   acc              maximum error

output:
   ifault = 1       required accuracy NOT achieved
            2       round-off error possibly significant
            3       invalid parameters
            4       unable to locate integration parameters
            5       out of memory

   trace[0]         absolute sum
   trace[1]         total number of integration terms
   trace[2]         number of integrations
   trace[3]         integration interval in final integration
```

```
   trace[4]         truncation point in initial integration
   trace[5]         s.d. of initial convergence factor
   trace[6]         cycles to locate integration parameters     */

{
      int j, nj, nt, ntm;  real acc1, almx, xlim, xnt, xntm;
      real utx, tausq, sd, intv, intv1, x, up, un, d1, d2, lj, ncj;
      extern real sigsq, lmax, lmin, mean;
      extern double intl,ersm;
      extern int r,lim; extern real c;
      extern int *n,*th; extern real *lb,*nc;
      real qfval;
      qfval = -1.0;
      static int rats[]={1,2,4,8};

      if (setjmp(env) != 0) { *ifault=4; goto endofproc; }
      r=r1; lim=lim1; c=c1;
      n=n1; lb=lb1; nc=nc1;
      for ( j = 0; j<7; j++ )  trace[j] = 0.0;
      *ifault = 0; count = 0;
      intl = 0.0; ersm = 0.0;
      acc1 = acc; ndtsrt = TRUE;  fail = FALSE;
      xlim = (real)lim;
      th=(int*)malloc(r*(sizeof(int)));
      if (! th) { *ifault=5;  goto  endofproc; }

      /* find mean, sd, max and min of lb,
         check that parameter values are valid */
      sigsq = square(sigma); sd = sigsq;
      lmax = 0.0; lmin = 0.0; mean = 0.0;
      for (j=0; j<r; j++)
      {
         nj = n[j];  lj = lb[j];  ncj = nc[j];
         if ( nj < 0  ||  ncj < 0.0 ) { *ifault = 3;  goto  endofproc;  }
         sd  = sd  + square(lj) * (2 * nj + 4.0 * ncj);
         mean = mean + lj * (nj + ncj);
         if (lmax < lj) lmax = lj ; else if (lmin > lj) lmin = lj;
      }
      if ( sd == 0.0  )
      {  qfval = (c > 0.0) ? 1.0 : 0.0; goto  endofproc;  }
      if ( lmin == 0.0 && lmax == 0.0 && sigma == 0.0 )
         { *ifault = 3;  goto  endofproc;  }
      sd = sqrt(sd);
      almx = (lmax < - lmin) ? - lmin : lmax;

      /* starting values for findu, ctff */
      utx = 16.0 / sd;  up = 4.5 / sd;  un = - up;
      /* truncation point with no convergence factor */
      findu(&utx, .5 * acc1);
      /* does convergence factor help */
      if (c != 0.0  &&(almx > 0.07 * sd))
      {
         tausq = .25 * acc1 / cfe(c);
         if (fail) fail = FALSE ;
         else if (truncation(utx, tausq) < .2 * acc1)
         {
            sigsq = sigsq + tausq;
            findu(&utx, .25 * acc1);
            trace[5] = sqrt(tausq);
         }
      }
      trace[4] = utx;  acc1 = 0.5 * acc1;

      /* find RANGE of distribution, quit if outside this */
   l1:
```

```
      d1 = ctff(acc1, &up) - c;
      if (d1 < 0.0) { qfval = 1.0; goto endofproc; }
      d2 = c - ctff(acc1, &un);
      if (d2 < 0.0) { qfval = 0.0; goto endofproc; }
      /* find integration interval */
      intv = 2.0 * pi / ((d1 > d2) ? d1 : d2);
      /* calculate number of terms required for main and
         auxillary integrations */
      xnt = utx / intv;  xntm = 3.0 / sqrt(acc1);
      if (xnt > xntm * 1.5)
      {
         /* parameters for auxillary integration */
         if (xntm > xlim) { *ifault = 1; goto endofproc; }
         ntm = (int)floor(xntm+0.5);
         intv1 = utx / ntm;  x = 2.0 * pi / intv1;
         if (x <= fabs(c)) goto l2;
         /* calculate convergence factor */
         tausq = .33 * acc1 / (1.1 * (cfe(c - x) + cfe(c + x)));
         if (fail) goto l2;
         acc1 = .67 * acc1;
         /* auxillary integration */
         integrate(ntm, intv1, tausq, FALSE );
         xlim = xlim - xntm;  sigsq = sigsq + tausq;
         trace[2] = trace[2] + 1; trace[1] = trace[1] + ntm + 1;
         /* find truncation point with new convergence factor */
         findu(&utx, .25 * acc1);  acc1 = 0.75 * acc1;
         goto l1;
      }

      /* main integration */
   l2:
      trace[3] = intv;
      if (xnt > xlim) { *ifault = 1; goto endofproc; }
      nt = (int)floor(xnt+0.5);
      integrate(nt, intv, 0.0, TRUE );
      trace[2] = trace[2] + 1; trace[1] = trace[1] + nt + 1;
      qfval = 0.5 - intl;
      trace[0] = ersm;

      /* test whether round-off error could be significant
         allow for radix 8 or 16 machines */
      up=ersm; x = up + acc / 10.0;
      for (j=0;j<4;j++) { if (rats[j] * x == rats[j] * up) *ifault = 2; }

   endofproc :
      free((char*)th);
      trace[6] = (real)count;
      return qfval;
}
```

Le fichier *qnorm.c*

```
      double qnorm(double P, int IFAULT)

{

     /*
       ALGORITHM AS241  APPL. STATIST. (1988) VOL. 37, NO. 3

       Produces the normal deviate Z corresponding to a given lower
       tail area of P; Z is accurate to about 1 part in 10**16.
       The hash sums below are the sums of the mantissas of the
       coefficients.   They are included for use in checking
       transcription.

     */
```


```
double ZERO, ONE, HALF, SPLIT1, SPLIT2, CONST1,CONST2, A0, A1, A2, A3, A4, A5, A6,
    A7, B1, B2, B3,B4, B5, B6, B7,C0, C1, C2, C3, C4, C5, C6, C7, D1, D2, D3, D4,
    D5,D6, D7, E0, E1, E2, E3, E4, E5, E6, E7, F1, F2, F3,F4, F5, F6, F7, Q, R,
    PPND16;
ZERO = (double)0.0;
ONE = (double)1.0;
HALF = (double)0.5;
SPLIT1 = (double)0.425;
SPLIT2 = (double)5.0;
CONST1 = (double)0.180625;
CONST2 = (double)1.6;


//        Coefficients for P close to 0.5

A0 = (double)3.3871328727963666080;
A1 = (double)1.3314166789178437745*pow(10,2);
A2 = (double)1.9715909503065514427*pow(10,3);
A3 = (double)1.3731693765509461125*pow(10,4);
A4 = (double)4.5921953931549871457*pow(10,4);
A5 = (double)6.7265770927008700853*pow(10,4);
A6 = (double)3.3430575583588128105*pow(10,4);
A7 = (double)2.5090809287301226727*pow(10,3);
B1 = (double)4.2313330701600911252*pow(10,1);
B2 = (double)6.8718700749205790830*pow(10,2);
B3 = (double)5.3941960214247511077*pow(10,3);
B4 = (double)2.1213794301586595867*pow(10,4);
B5 = (double)3.9307895800092710610*pow(10,4);
B6 = (double)2.8729085735721942674*pow(10,4);
B7 = (double)5.2264952788528545610*pow(10,3);

//      HASH SUM AB     55.88319 28806 14901 4439

//        Coefficients for P not close to 0, 0.5 or 1.

C0 = (double)1.42343711074968357734;
C1 = (double)4.63033784615654529590;
C2 = (double)5.76949722146069140550;
C3 = (double)3.64784832476320460504;
C4 = (double)1.27045825245236838258;
C5 = (double)2.41780725177450611770*pow(10,-1);
C6 = (double)2.27238449892691845833*pow(10,-2);
C7 = (double)7.74545014278341407640*pow(10,-4);
D1 = (double)2.05319162663775882187;
D2 = (double)1.67638483018380384940;
D3 = (double)6.89767334985100004550*pow(10,-1);
D4 = (double)1.48103976427480074590*pow(10,-1);
D5 = (double)5.51986665636164571966*pow(10,-2);
D6 = (double)5.47593808499534494600*pow(10,-4);
D7 = (double)1.05075007164441684324*pow(10,-9);

//      HASH SUM CD     49.33206 50330 16102 89036

//        Coefficients for P near 0 or 1.

E0 = (double)6.65790464350110377720*pow(10,0);
E1 = (double)5.46378491116411436990*pow(10,0);
E2 = (double)1.78482653991729133580*pow(10,0);
E3 = (double)2.96560571828504891230*pow(10,-1);
E4 = (double)2.65321895265761230930*pow(10,-2);
E5 = (double)1.24266094738807843860*pow(10,-3);
E6 = (double)2.71155556874348757815*pow(10,-5);
E7 = (double)2.01033439929228813265*pow(10,-7);
F1 = (double)5.99832206555887937690*pow(10,-1);
F2 = (double)1.36929880922735805310*pow(10,-1);
```

```
  F3 = (double)1.48753612908506148525*pow(10,-2);
  F4 = (double)7.86869131145613259100*pow(10,-4);
  F5 = (double)1.84631831751005468180*pow(10,-5);
  F6 = (double)1.42151175831644588870*pow(10,-7);
  F7 = (double)2.04426310338993978564*pow(10,-15);

//       HASH SUM EF     47.52583 31754 92896 71629

  IFAULT = 0;
  Q = P - HALF;
  if (fabs(Q)<=SPLIT1) {
    R = CONST1 - Q * Q;
    PPND16 = Q * (((((((A7 * R + A6) * R + A5) * R + A4) * R + A3)* R + A2) * R + A1
         ) * R + A0) /(((((((B7 * R + B6) * R + B5) * R + B4) * R + B3)* R + B2) * R
          + B1) * R + ONE);
    return(PPND16);}
  else {
    if (Q<ZERO) {
      R = P;
    }
    else {
      R = ONE - P;}
  }
  if (R<=ZERO) {
    IFAULT = 1;
    PPND16 = ZERO;
    return(PPND16);}

  R = sqrt(-log(R));
  if (R<=SPLIT2) {
    R = R - CONST2;
    PPND16 = (((((((C7 * R + C6) * R + C5) * R + C4) * R + C3)* R + C2) * R + C1) * R
          + C0) /(((((((D7 * R + D6) * R + D5) * R + D4) * R + D3)* R + D2) * R + D1
         ) * R + ONE);}
  else {
    R = R - SPLIT2;
    PPND16 = (((((((E7 * R + E6) * R + E5) * R + E4) * R + E3)* R + E2) * R + E1) * R
          + E0) /(((((((F7 * R + F6) * R + F5) * R + F4) * R + F3)* R + F2) * R + F1
         ) * R + ONE);
  }
  if (Q<ZERO) {PPND16 = - PPND16;
  return(PPND16);}

  return(PPND16);
}
```

Le fichier *rank.c*

```
void   rank(Matrix RV,char ORDER[2],int *IRANK)
{
     /*

      rank RANKS A VECTOR OF REAL NUMBERS IN ASCENDING
      OR DESCENDING ORDER.

      rank USES A VARIANT OF LIST-MERGING, AS DESCRIBED
      BY KNUTH. THE ROUTINE TAKES ADVANTAGE OF NATURAL
      ORDERING IN THE DATA, AND USES A SIMPLE LIST INSERTION
      IN A PREPARATORY PASS TO GENERATE ORDERED LISTS OF
      LENGTH AT LEAST 10. THE RANKING IS STABLE: EQUAL ELEMENTS
      PRESERVE THEIR ORDERING IN THE INPUT DATA.

      THE MINIMUM LENGTH OF THE LISTS AT THE END OF THE
      PREPARATORY PASS IS DEFINED BY THE VARIABLE MAXINS.

*/
  //      .. Parameters ..
```

```
int MAXINS;
MAXINS=10;

int M1, M2;
M1=1;
M2=RV.Nrows();


//      .. Local Scalars ..
double   A, B, C;
int I, I1, I2, ILIST, J, J1, J2, K, K1, K2, L,LIST1, LIST2, NLAST, NPREV;


I1=0;
J1=0;
LIST1=0;
LIST2=0;
NLAST=0;
NPREV=0;

//      .. Executable Statements ..

// DEAL WITH THE TRIVIAL CASE.

if (M1==M2) *(IRANK+M2-1) = M2;


/*
  INITIALISE, USING NATURAL RUNS IN BOTH DIRECTIONS AND
  STRAIGHT LIST INSERTION FOR SMALL LISTS.
  I   POINTS TO THE SMALLEST ELEMENT IN THE CURRENT LIST
  J   POINTS TO THE LARGEST  ELEMENT IN THE CURRENT LIST
  B   IS THE VALUE OF THE SMALLEST ELEMENT IN CURRENT LIST
  C   IS THE VALUE OF THE LARGEST  ELEMENT IN CURRENT LIST
*/

ILIST = -1;
K = M1;
I = K;
J = K;
L = K + MAXINS;
B = RV(K,1);
C = B;

for (K=M1+1;K<=M2;K=K+1)
  {

    //              DEAL WITH ADDITIONS AT EITHER END.

    A = RV(K,1);
    if (A>=C)
      {
        *(IRANK+J-1) = K;
        J = K;
        C = A;
      }
       else if  (A<B)
         {
           *(IRANK+K-1) = I;
           I = K;
           B = A;
         }
          else
            {
```

```
                //              DO AN ASCENDING LIST INSERTION.

              if (K<L)
                {
                  I2 = I;
                Vingt:
                  I1 = I2;
                  I2 = *(IRANK+I1-1);
                  if (A>=RV(I2,1)) goto Vingt;
                  *(IRANK+I1-1) = K;
                  *(IRANK+K-1) = I2;
                }
              else
                {

                  //                ADD THE CURRENT LIST ON TO THE OTHERS.

                  if (ILIST<0)
                    {
                      LIST1 = -I;
                      ILIST = 0;
                    }
                  else if (ILIST==0)
                    {
                      LIST2 = -I;
                      ILIST = 1;
                      NPREV = NLAST;
                    }
                  else
                    {
                      *(IRANK+NPREV-1) = -I;
                      NPREV = NLAST;
                    }



                  NLAST = J;
                  I = K;
                  J = K;
                  L = K + MAXINS;
                  B = RV(K,1);
                  C = B;
                }
            }
  }

//        TIDY UP AT THE END.

*(IRANK+J-1) = 0;
      if (ILIST<0)
        {
          LIST1 = -I;
          goto Deuxcentquatrevingt;
        }
      else if (ILIST==0)
        {
          LIST2 = -I;
        }
      else
        {
          *(IRANK+NPREV-1) = -I;
        }
      *(IRANK+NLAST-1) = 0;

      //        AT THIS POINT:
```

```
        //          LIST1 = - (INDEX OF LEAST ELEMENT IN THE FIRST LIST)
        //          LIST2 = - (INDEX OF LEAST ELEMENT IN THE SECOND LIST)
        //          FOR EACH K, IRANK(K) IS THE INDEX OF THE NEXT ELEMENT IN THE
        //          CURRENT LIST, EXCEPT THAT, IF THERE IS NO SUCH ELEMENT,
        //          IRANK(K) IS - (INDEX OF THE LEAST ELEMENT IN THE NEXT LIST
        //          BUT 1)  OR 0 IF THERE IS NO SUCH LIST.

        //          START MERGING LISTS BY PAIRS.

 Soixante:
  ILIST = -1;
 I = -LIST1;
 J = -LIST2;
 Quatrevingt:
  K = I;
 if (RV(I,1)>RV(J,1)) K = J;
 if (ILIST<0)
   {
     LIST1 = -K;
     ILIST = 0;
   }
    else if (ILIST==0)
      {
        LIST2 = -K;
        ILIST = 1;
        NLAST = L;
      }
       else
         {
           *(IRANK+NLAST-1) = -K;
           NLAST = L;
         }

 //          MOVE ALONG THE LISTS UNTIL ONE FINISHES.

 //          NEW VARIABLES I2, J2 AND K2 ARE USED INSTEAD OF I, J AND K
 //          WITHIN THE INNERMOST BLOCK TO ENCOURAGE OPTIMISING COMPILERS TO
 //          STORE THEM IN REGISTERS.
 //            I2 POINTS TO THE CURRENT ELEMENT IN THE FIRST LIST
 //            J2 POINTS TO THE CURRENT ELEMENT IN THE SECOND LIST
 //            K2 POINTS TO THE CURRENT ELEMENT IN THE MERGED LIST

 I2 = I;
 J2 = J;
 if (K!=I2) goto Centquarante;
Cent:
  A = RV(J2,1);
K2 = I2;
Centvingt:
  I2 = K2;
K2 = *(IRANK+I2-1);
if (K2<=0) goto Centquatrevingt;
if (A>=RV(K2,1)) goto Centvingt;
*(IRANK+I2-1) = J2;
I2 = K2;
Centquarante:
  A = RV(I2,1);
K2 = J2;
Centsoixante:
  J2 = K2;
K2 = *(IRANK+J2-1);
if (K2<=0) goto Deuxcent;
if (A>RV(K2,1)) goto Centsoixante;
*(IRANK+J2-1) = I2;
J2 = K2;
```


```
goto Cent;

//          ADD THE REMAINS OF ONE LIST TO THE OTHER.

Centquatrevingt:
  K = 1;
I1 = K2;
goto Deuxcentvingt;
Deuxcent:
  K = 2;
J1 = K2;
Deuxcentvingt:
  I = I2;
J = J2;
if (K==1)
  {

    //              FIRST LIST IS EXHAUSTED

    *(IRANK+I-1) = J;
    I = -I1;
    J1 = J;
Deuxcentquarante:
      J = J1;
J1 = *(IRANK+J-1);
if (J1>0) goto Deuxcentquarante;
L = J;
J = -J1;
  }
   else
     {

       //              SECOND LIST IS EXHAUSTED

       *(IRANK+J-1) = I;
       J = -J1;
       I1 = I;
Deuxcentsoixante:
         I = I1;
I1 = *(IRANK+I-1);
if (I1>0) goto Deuxcentsoixante;
L = I;
I = -I1;
     }

//          TIDY UP AND CARRY ON if NOT FINISHED.

if ((I!=0) & (J!=0)) goto Quatrevingt;
*(IRANK+L-1) = 0;
K = I + J;
       if (ILIST>0)
         {
           *(IRANK+NLAST-1) = -K;
           goto Soixante;
         }
       else if (K!=0)
         {
           LIST2 = -K;
           goto Soixante;
         }

       //          IF DESCENDING, REVERSE ALL POINTERS BETWEEN EQUALITY
       //          BLOCKS.

Deuxcentquatrevingt:
```

```
    if ((ORDER=="D") | (ORDER=="d"))
      {
        I = 0;
        J = -LIST1;
  Troiscent:
          K = J;
  K1 = K;
  A = RV(K,1);
  Troiscentvingt:
          K = K1;
  K1 = *(IRANK+K-1);
            if (K1!=0)
              {
                if (A==RV(K1,1)) goto Troiscentvingt;
              }
            *(IRANK+K-1) = I;
            I = J;
            J = K1;
            if (J!=0) goto Troiscent;
            LIST1 = -I;
      }

  //        CONVERT THE LIST FORM TO RANKS AND RETURN.

  K = M1;
  I = -LIST1;
  Troiscentquarante:
    I1 = *(IRANK+I-1);
  *(IRANK+I-1) = K;
  K = K + 1;
  I = I1;
  if (I>0) goto Troiscentquarante;

}
```

Le fichier *valpmultiv.c*

```
/* Debut-Commentaires

Nom de la fonction: valpmultiv

Entrees: b double facteur de lissage, q entier dimension de l'espace

Sorties: D DiagonalMatrix

Description:
Calcule la matrice D des valeurs propres pour la formule de cubature suivante
E_n^{r^2}: 7-2 (n>2) Degree 7, 2^{n+1}+4n^2 Points.

Utilisation dans une fonction main:
-----------------------------------

#include <iostream.h>
#include<math.h>
#define WANT_STREAM                  // include.h will get stream fns
#define WANT_MATH                    // include.h will get math fns
                                     // newmatap.h will get include.h
#include "newmatap.h"                // need matrix applications

#include "newmatio.h"                // need matrix output routines

#ifdef use_namespace
using namespace NEWMAT;              // access NEWMAT namespace
#endif
#include "binarycode.c"
#include "valpmultiv.c"
```

```
int main()

{

  DiagonalMatrix valpmultiv(double b, int q);

  double b;
  int q;

  b=0.7;
  q=3;

  DiagonalMatrix D(2*(2*q+(int)pow(2,q)+2*(int)pow(q,2)-2*q));



  D=valpmultiv(b,q);

  cout << "Affichage des valeurs propres\n";

  cout << D;

  cout << "\n";


return(0);

}
```

```
Instructions de compilation:
g++ -Wall -O nom_du_fichier_contenant_la_fonction_main.cc -L. -lnewmat

Fonctions exterieures appelees:
binarycode, Absolute, et aussi la librairie libnewmat.a et les fichiers qui vont avec
    cette librairie:
newmatap.h, newmatio.h, newmat.h, include.h, boolean.h, myexcept.h

Auteur: Pierre Lafaye de Micheaux

Date: 24/08/2002

Fin-Commentaires */
#define WANT_STREAM
#define WANT_MATH
#define WANT_TIME

#include "./include/include.h"
#include "./include/newran.h"
#include "./include/tryrand.h"

#ifdef use_namespace
using namespace NEWRAN;
#endif


#include "gauher.c"


DiagonalMatrix valpmultiv(double b, int q, int choix, int N, double prec, int MAXIT)
```

```
{

  //Declaration des variables
  int i, j, k;
  Matrix poids(N,1);
  Matrix Ddemi(N,N);
  Ddemi=0.0;
  SymmetricMatrix DKD(N);
  DiagonalMatrix D(N);
  Matrix abscisses(N,q);
  Matrix poidsetabscisses(N,q+1);

  //Declaration des fonctions
  Matrix gauher(double b, int N, double prec, int MAXIT);

//------------------------------------------------------------------

  //CUBATURE
//------------------------------------------------------------------


  if (choix==1) {



      if (q==1)

     {




      poidsetabscisses=gauher(b,N,prec,MAXIT);

      poids=poidsetabscisses.SubMatrix(1,N,q+1,q+1);

      abscisses=poidsetabscisses.SubMatrix(1,N,1,q);


     }
      //fin de if q==1


      if (q==2)

     {


      int cuba;

      //Choix de la cubature du plan
      cuba=1;


      if (cuba==1)

        {
      //E_2^(r^2) : 15-1. Degree 15, 44 Points dans Stroud (1971) p.326

      double s1,s2,s3,s4,s5,s6,s7,s8,s9;
      double r1,r2,r3,r4,r5,r6,r7,r8,r9;
      double B1,B2,B3,B4,B5,B6,B7,B8,B9;


      r1=3.538388728121806997598448164209 77;
```

```
s1=0.0;
B1=8.00648356965962968878302657382053*pow(10,-6);

r2=2.359676416877928584561091634362 14;
s2=0.0;
B2=3.60457742083826400410253786082198*pow(10,-3);

r3=1.312801844620926639460031910583 49;
s3=0.0;
B3=0.11876093307591367444310266343 1639;

r4=0.538955948211420514863252356676544;
s4=0.0;
B4=0.43724885437914023746791822368 9153;

r5=2.300279949805657894652399753115 49;
s5=2.300279949805657894652399753115 49;
B5=3.67173507583298936096131754975575*pow(10,-5);

r6=1.581138830084189665999446772216 35;
s6=1.581138830084189665999446772216 35;
B6=5.65486677646162782923275808990310*pow(10,-3);

r7=0.841850433581927898923665650469697;
s7=0.841850433581927898923665650469697;
B7=0.17777742684242396740337600231 8122;

r8=2.685533581755340900630942141671 63;
s8=1.112384431771457249718213420474 72;
B8=2.73544964785329001953807301017241*pow(10,-4);

r9=1.740847514397402607079300756635 72;
s9=0.721082650486896005766801022499833;
B9=0.02087984556938594547036132481306 47;

abscisses(1,1)=r1;
abscisses(1,2)=s1;
poids(1,1)=B1;

abscisses(2,1)=r2;
abscisses(2,2)=s2;
poids(2,1)=B2;

abscisses(3,1)=r3;
abscisses(3,2)=s3;
poids(3,1)=B3;

abscisses(4,1)=r4;
abscisses(4,2)=s4;
poids(4,1)=B4;

abscisses(5,1)=r5;
abscisses(5,2)=s5;
poids(5,1)=B5;

abscisses(6,1)=r6;
abscisses(6,2)=s6;
poids(6,1)=B6;

abscisses(7,1)=r7;
abscisses(7,2)=s7;
poids(7,1)=B7;

abscisses(8,1)=r8;
abscisses(8,2)=s8;
```

```
poids(8,1)=B8;

abscisses(9,1)=r9;
abscisses(9,2)=s9;
poids(9,1)=B9;

abscisses(10,1)=r5;
abscisses(10,2)=-s5;
poids(10,1)=B5;

abscisses(11,1)=r6;
abscisses(11,2)=-s6;
poids(11,1)=B6;

abscisses(12,1)=r7;
abscisses(12,2)=-s7;
poids(12,1)=B7;

abscisses(13,1)=r8;
abscisses(13,2)=-s8;
poids(13,1)=B8;

abscisses(14,1)=r9;
abscisses(14,2)=-s9;
poids(14,1)=B9;

abscisses(15,1)=-r1;
abscisses(15,2)=s1;
poids(15,1)=B1;

abscisses(16,1)=-r2;
abscisses(16,2)=s2;
poids(16,1)=B2;

abscisses(17,1)=-r3;
abscisses(17,2)=s3;
poids(17,1)=B3;

abscisses(18,1)=-r4;
abscisses(18,2)=s4;
poids(18,1)=B4;

abscisses(19,1)=-r5;
abscisses(19,2)=s5;
poids(19,1)=B5;

abscisses(20,1)=-r6;
abscisses(20,2)=s6;
poids(20,1)=B6;

abscisses(21,1)=-r7;
abscisses(21,2)=s7;
poids(21,1)=B7;

abscisses(22,1)=-r8;
abscisses(22,2)=s8;
poids(22,1)=B8;

abscisses(23,1)=-r9;
abscisses(23,2)=s9;
poids(23,1)=B9;

abscisses(24,1)=-r5;
abscisses(24,2)=-s5;
poids(24,1)=B5;
```

```
abscisses(25,1)=-r6;
abscisses(25,2)=-s6;
poids(25,1)=B6;

abscisses(26,1)=-r7;
abscisses(26,2)=-s7;
poids(26,1)=B7;

abscisses(27,1)=-r8;
abscisses(27,2)=-s8;
poids(27,1)=B8;

abscisses(28,1)=-r9;
abscisses(28,2)=-s9;
poids(28,1)=B9;

abscisses(29,1)=s1;
abscisses(29,2)=r1;
poids(29,1)=B1;

abscisses(30,1)=s2;
abscisses(30,2)=r2;
poids(30,1)=B2;

abscisses(31,1)=s3;
abscisses(31,2)=r3;
poids(31,1)=B3;

abscisses(32,1)=s4;
abscisses(32,2)=r4;
poids(32,1)=B4;

abscisses(33,1)=s8;
abscisses(33,2)=r8;
poids(33,1)=B8;

abscisses(34,1)=s9;
abscisses(34,2)=r9;
poids(34,1)=B9;

abscisses(35,1)=s1;
abscisses(35,2)=-r1;
poids(35,1)=B1;

abscisses(36,1)=s2;
abscisses(36,2)=-r2;
poids(36,1)=B2;

abscisses(37,1)=s3;
abscisses(37,2)=-r3;
poids(37,1)=B3;

abscisses(38,1)=s4;
abscisses(38,2)=-r4;
poids(38,1)=B4;

abscisses(39,1)=s8;
abscisses(39,2)=-r8;
poids(39,1)=B8;

abscisses(40,1)=s9;
abscisses(40,2)=-r9;
poids(40,1)=B9;
```


```
        abscisses(41,1)=-s8;
        abscisses(41,2)=-r8;
        poids(41,1)=B8;

        abscisses(42,1)=-s9;
        abscisses(42,2)=-r9;
        poids(42,1)=B9;

        abscisses(43,1)=-s8;
        abscisses(43,2)=r8;
        poids(43,1)=B8;

        abscisses(44,1)=-s9;
        abscisses(44,2)=r9;
        poids(44,1)=B9;



          }


        if (cuba==2)

          {



          //Degree 15, 44 Points
          //Hae76 A. Haegemans, Circularly symmetrical integration formulas
          //for two-dimensional circularly symmetrical regions, BIT 16 (1976), 52--59.


        Matrix M(9,4);

M<<
  1 << 8 << 1.8842797714821495959716078946321 5*pow
      (10,0) << 2.0879845569385945470361324813064 7*pow(10,-2) <<
  2 << 8 << 2.9068006025152771214975435667614 9*pow
      (10,0) << 2.7354496478532900195380730101724 1*pow(10,-4) <<
  3 << 4 << 1.1905563006612329013649853671029 3*pow
      (10,0) << 1.7777742684242396740337600231812 2*pow(10,-1) <<
  4 << 4 << 2.2360679774997896964091736687312 7*pow
      (10,0) << 5.6548667764616278292327580899031 0*pow(10,-3) <<
  5 << 4 << 3.2530871022700637190800790278631 5*pow
      (10,0) << 3.6717350758329893609613175497557 5*pow(10,-5) <<
  6 << 4 << 5.3895594821142051486325235667654 4*pow
      (10,-1) << 4.3724885437914023746791822368915 3*pow(10,-1) <<
  7 << 4 << 1.3128018446209266394600319105834 9*pow
      (10,0) << 1.1876093307591367444310266343163 9*pow(10,-1) <<
  8 << 4 << 2.3596764168779285845610916343621 4*pow
      (10,0) << 3.6045774208382640041025378608219 8*pow(10,-3) <<
    9 << 4 << 3.5383887281218069975984481642097 7*pow
        (10,0) << 8.0064835696596296887830265738205 3*pow(10,-6);


        int somme, somme2;
        somme=0;
        somme2=0;

        for (i=1;i<=(M.Nrows());i=i+1)
          {

            for (j=1;j<=((int)M(i,2));j=j+1)
              {
                poids(somme+j,1)=M(i,4);
```

```
        }

      somme2=0;
      for (j=1;j<=((int)M(i,2)/4);j=j+1)
        {
          for (k=1;k<=4;k=k+1)
            {
              abscisses(somme+somme2+k,1)=M(i,3)*cos((k-1)*pi/2.0);
              abscisses(somme+somme2+k,2)=M(i,3)*sin((k-1)*pi/2.0);
            }
          somme2=somme2+((int)M(i,2))/4;
        }

      somme=somme+(int)M(i,2);
    }

    }


  if (cuba==3)

    {



  // Degree 31, 172 Points
  //Hae75 A. Haegemans, Tables of circularly symmetrical integration formulas of
       degree 2d-1
  //for two-dimensional circularly symmetrical regions, Report TW 27, K.U.
      Leuven Applied Mathematics
  //and Programming Division, 1975.

  Matrix M(25,4);


M<< 1 << 16 << 2.19342435218571098000852165314022*pow
     (10,0) << 2.732364560898213699388468994558 53*pow(10,-3)
 << 2 << 16 << 2.941430727858826584438904641213 76*pow
     (10,0) << 7.458827087445797803349674518750 29*pow(10,-5)
 << 3 << 16 << 3.700117405214835909099843918228 18*pow
     (10,0) << 6.515792027583594475485824353301 21*pow(10,-7)
 << 4 << 16 << 4.565742663803208432746366972745 91*pow
     (10,0) << 7.756796167072629543399023907796 25*pow(10,-10)
 << 5 << 8 << 1.21580764835581239743824121978989*pow
     (10,0) << 6.295108948883821897212671772984 24*pow(10,-2)
 << 6 << 8 << 1.75022957356728645712052732323077*pow
     (10,0) << 1.586926092237830758945860189409 27*pow(10,-2)
 << 7 << 8 << 2.56265642157773665089210281907111*pow
     (10,0) << 5.276602208670586478074078439388 69*pow(10,-4)
 << 8 << 8 << 3.30618113478644619137730123417901*pow
     (10,0) << 8.778725892938517502569520947127 42*pow(10,-6)
 << 9 << 8 << 4.10268615562676937418264214216800*pow
     (10,0) << 3.382470751230419953740228391987 93*pow(10,-8)
 << 10 << 8 << 5.13066763122356142415670471249892*pow
     (10,0) << 5.266858265076377455526355965376 58*pow(10,-12)
 << 11 << 4 << 7.81924734415062654910759895512688*pow
     (10,-1) << 1.672330608645080927201037203128 52*pow(10,-1)
 << 12 << 4 << 1.39054046557164805349275297468774*pow
     (10,0) << 3.855692941884140230003552520367 18*pow(10,-2)
 << 13 << 4 << 1.81698526640070493276888680298209*pow
     (10,0) << 9.932756404313634390420429007582 30*pow(10,-3)
 << 14 << 4 << 2.58066849114911903633979005815686*pow
     (10,0) << 4.804144327033073035530967554989 15*pow(10,-4)
```

```
<< 15 << 4 << 3.3232180249450385519228351984745 9*pow
    (10,0) << 7.9074397120962214293228755139503 2*pow(10,-6)
<< 16 << 4 << 4.1258903792434740766969662340077 8*pow
    (10,0) << 2.8484113751162585622291411601396 3*pow(10,-8)
<< 17 << 4 << 5.2011664474077627174968517365613 0*pow
    (10,0) << 2.9722099836175491442919568707439 7*pow(10,-12)
<< 18 << 4 << 3.5704785520064025824467780192408 8*pow
    (10,-1) << 2.2474145171424757352743948699452 6*pow(10,-1)
<< 19 << 4 << 8.5758226983760005344043267466427 5*pow
    (10,-1) << 1.3261906753994189757239010053188 1*pow(10,-1)
<< 20 << 4 << 1.4562719277216477685078359374991 5*pow
    (10,0) << 3.4163514949449074086289723291373 7*pow(10,-2)
<< 21 << 4 << 1.8590835457207681517581881951896 1*pow
    (10,0) << 7.2527748323079542834817149679073 6*pow(10,-3)
<< 22 << 4 << 2.5890404715899838159591919065214 8*pow
    (10,0) << 4.5858819068284511543370555496990 6*pow(10,-4)
<< 23 << 4 << 3.3300469877048942117037317871089 0*pow
    (10,0) << 7.5753161335863780237588860166129 4*pow(10,-6)
<< 24 << 4 << 4.1346078781449015495113700444154 9*pow
    (10,0) << 2.6683532734955860827280658742941 6*pow(10,-8)
<< 25 << 4 << 5.2291046716963438612688618344613 7*pow
    (10,0) << 2.3761740461239643801211443400040 6*pow(10,-12);
```

```
int somme, somme2;
somme=0;
somme2=0;

for (i=1;i<=(M.Nrows());i=i+1)
  {

    for (j=1;j<=((int)M(i,2));j=j+1)
      {
        poids(somme+j,1)=M(i,4);
      }

    somme2=0;
    for (j=1;j<=((int)M(i,2)/4);j=j+1)
      {
        for (k=1;k<=4;k=k+1)
          {
            abscisses(somme+somme2+k,1)=M(i,3)*cos((k-1)*pi/2.0);
            abscisses(somme+somme2+k,2)=M(i,3)*sin((k-1)*pi/2.0);
          }
        somme2=somme2+((int)M(i,2))/4;
      }

    somme=somme+(int)M(i,2);
  }




  }
```

```
for (j=1;j<=N;j=j+1) poids(j,1)=pow(pi,(-q/2.0))*poids(j,1);
```

```
      abscisses=sqrt(2.0)*b*abscisses;



    }
      //fin de if q==2



  if (q>=3) {

    //E_q^(r^2): 7-2 (q>=3) Degree 7, 2^(q+1)+4q^2 Points dans Stroud (1971) p319
    //Attention il y a une erreur dans le livre, voir plutot Stroud (1967) : Some
        seventh degree
    //integration formulas for symmetric regions, SIAM J. Numer. Anal. pp. 37-44


void binarycode(int i,int q, int *y);
double Absolute(double x);
int *y;
double A1, A2, r1, r2, sval, tval, PI;

PI=3.14159265358979311600;


y=new int [q];



Matrix Bval(2*q,1);
Matrix Cval((int)pow(2,q),1);
Matrix Dval(2*(int)pow(q,2)-2*q,1);
Matrix B((int)pow(2,q)+2*(int)pow(q,2),1);



Matrix pointsB(q,q);
pointsB=0.0;

Matrix pointsC((int)pow(2,q),q);
pointsC=0.0;

Matrix signes((int)pow(2,q),q);

Matrix pointsDtemp((q*(q-1))/2,q);
pointsDtemp=0.0;

Matrix AbspointsDtemp((q*(q-1))/2,q);

Matrix pointsD(4*(q*(q-1))/2,q);

Matrix uj(2*q+(int)pow(2,q)+4*(q*(q-1))/2,q);
```

```
A1=(q+2+sqrt(2*(q+2)))/(4*(q+2))*exp(gamma(q/2.0));

A2=(q+2-sqrt(2*(q+2)))/(4*(q+2))*exp(gamma(q/2.0));

r1=sqrt((q+2-sqrt(2*(q+2)))/2);

r2=sqrt((q+2+sqrt(2*(q+2)))/2);


for (j=1;j<=2*q;j=j+1) Bval(j,1)=(8.0-q)/(q*(q+2)*(q+4));


for (j=1;j<=(int)pow(2,q);j=j+1) Cval(j,1)=(pow(2.0,-q)*pow(q,3))/(q*(q+2)*(q+4));


for (j=1;j<=2*(int)pow(q,2)-2*q;j=j+1) Dval(j,1)=4.0/(q*(q+2)*(q+4));


B=Bval&(Cval&Dval);

sval=1/sqrt(q);

tval=1/(sqrt(2));


pointsB(1,1)=1;

for (j=2;j<=q-1;j=j+1) pointsB(j,j)=1;

pointsB(q,q)=1;


pointsB=pointsB&-pointsB;


for (i=0;i<=(int)pow(2,q)-1;i=i+1)

  {

    binarycode(i,q,y);
    for (j=1;j<=q;j=j+1) signes(i+1,j)=pow(-1,*(y+j-1));


  }

delete y;


for (i=1;i<=(int)pow(2,q);i=i+1) pointsC.Row(i)=sval*signes.Row(i);



for (i=1;i<=q-1;i=i+1) {
```

```
        for (j=1;j<=q-i;j=j+1) {

          pointsDtemp((q*(i-1)-i*(i-1)/2)+j,i)=-tval;

                        if (i+j<q+1) {

                        pointsDtemp((q*(i-1)-i*(i-1)/2)+j,i+j)=tval; }

                        }
                        }

for (i=1;i<=(q*(q-1))/2;i=i+1){
  for (j=1;j<=q;j=j+1){

     AbspointsDtemp(i,j)=Absolute(pointsDtemp(i,j));

  }}

pointsD=pointsDtemp&-pointsDtemp&AbspointsDtemp&-AbspointsDtemp;


uj=pointsB&pointsC&pointsD;


poids=((A1*B)&(A2*B))*((2*pow(PI,q/2.0))/(exp(gamma(q/2.0))));

for (j=1;j<=N;j=j+1) poids(j,1)=pow(PI,(-q/2.0))*poids(j,1);



abscisses=((r1*uj)&(r2*uj));

abscisses=sqrt(2.0)*b*abscisses;


   }
   // fin de if q>=3


 }
 //fin de if choix==1


//-------------------------------------------------------------------------
 //MONTE-CARLO
//-------------------------------------------------------------------------

 Normal Z;

 if (choix==2)

   {

for (i=1;i<=N;i=i+1){
  for(j=1;j<=q;j=j+1) {
    abscisses(i,j)=b*Z.Next();}}

for (i=1;i<=N;i=i+1)
  poids(i,1)=1.0/N;

   }
 //fin de if choix==2
```

```
  //On fabrique la matrice Ddemi
  for (j=1;j<=N;j=j+1) Ddemi(j,j)=sqrt(poids(j,1));


  //On fabrique la matrice K
  Matrix K(N,N);
  K=0.0;
  for (i=1;i<=N;i=i+1)
    {

      for (j=1;j<=N;j=j+1)
        {

  K(i,j)=exp(-((abscisses.Row(i)-abscisses.Row(j)).SumSquare())/2.0)-exp(-((abscisses.
      Row(i)).SumSquare()+(abscisses.Row(j)).SumSquare())/2.0);

        }
    }



  //On fabrique Ddemi*K*Ddemi
  DKD<<(Ddemi*K*Ddemi);

  //Calcul des valeurs propres.
  EigenValues(DKD,D);


  //On classe les valeurs propres par ordre decroissant
  SortDescending(D);


   return(D);

}



double Absolute(double x)

{


  if (x>0) return(x);
  else return(-x);

}
```

Le fichier *nCm.c*

```
int nCm(int n, int m)

{

  /*
  DESCRIPTION:
        Compute the binomial coefficient ("n choose m"),  where n  and m are
        any integer.

  REFERENCE:
        Feller (1968) An Introduction to Probability Theory and Its
        Applications, Volume I, 3rd Edition, pp 50, 63.
  */

  int out;
```

```
  out=(int)floor((exp(lgamma(n + 1) - lgamma(m + 1) - lgamma(n - m
       + 1)))*10.0+0.5)/10;

  return(out);

}
```

B.2. QUANTILES EMPIRIQUES

Le fichier *test.c*

```
// DEBUT DES DEFINITIONS POUR POUVOIR UTILISER LES LIBRAIRIES
//-------------------------------------------------------------

#include <string.h>
#define WANT_STREAM                    // include.h will get stream fns
#define WANT_MATH                      // include.h will get math fns
// newmatap.h will get include.h
#include "./include/newmatap.h"                // need matrix applications

#include "./include/newmatio.h"                // need matrix output routines

#ifdef use_namespace
using namespace NEWMAT;                // access NEWMAT namespace
#endif

#ifdef use_namespace
using namespace NEWRAN;
#endif

#include "./include/newran.h" // Pour la generation de nombres aleatoires

#include<time.h>

//Pour compiler:
//g++ -Wall -O test.c -L. -lnewmat -lnewran -o test

int main(void)

{
  int debut;
  int fin;
  debut=time(0);

  int p, q, n, CardA, i, j, k, l, lprim, N, nval, bval, serial;
  double b, prodA, statA, alpha, quant;
  Random::Set(0.46875);
  Normal Z;

  //Nombre de boucles de Monte-Carlo     **
  N=10000;

  //Si serial=0: non-serial case, si serial=1: serial case    **
  serial=0;


  RowVector bvalues(4);
  bvalues(1)=0.1;
  bvalues(2)=0.5;
  bvalues(3)=1.0;
  bvalues(4)=3.0;

  RowVector nvalues(3);
  nvalues(1)=20;
```

```
nvalues(2)=50;
nvalues(3)=100;

if (serial==0) {

  for (alpha=0.1;alpha>=0.05;alpha=alpha/2.0)
    {

      for (bval=1;bval<=4;bval=bval+1)
        {
          b=bvalues(bval);
          for (nval=1;nval<=3;nval=nval+1)
            {
              n=(int)nvalues(nval);

              for (q=2;q<=3;q=q+1) {

                for (CardA=2;CardA<=4;CardA=CardA+1) {
                  p=CardA;

                  prodA=1.0;
                  Matrix A(1,CardA);
                  for (i=1;i<=CardA;i=i+1) A(1,i)=i;

                  Matrix Vecteps(n,p*q);
                  Matrix mean(q,1);
                  SymmetricMatrix S(q);
                  Matrix Smudm(q,q);
                  DiagonalMatrix D(q);
                  Matrix U(q,q);
                  Matrix V(q,q);
                  Matrix Vecte(n,p*q);

                  ColumnVector vectstatA(N);

                  // Debut de la boucle de Monte-Carlo
                  for (i=1;i<=N;i=i+1)

                    {

                      //Remplissage de Vecteps
                      for (k=1;k<=n;k=k+1)
                        {
                          for (l=1;l<=p*q;l=l+1)
                            {

                              Vecteps(k,l)=Z.Next();
                            }
                        }

                      //Calcul de mean
                      mean=0.0;
                      for (j=1;j<=n;j=j+1)
                        {
                          for (k=1;k<=p;k=k+1)
                            {
                              mean=mean+(Vecteps.SubMatrix(j,j,q*(k-1)+1,q*k)).t();
                            }
                        }
                      mean=mean/(n*p);

                      //Calcul de S
                      S=0.0;
                      for (j=1;j<=n;j=j+1)
                        {
```

```
        for (k=1;k<=p;k=k+1)
          {
            S<<S+((Vecteps.SubMatrix(j,j,q*(k-1)+1,q*k)).t()-mean
                )*(Vecteps.SubMatrix(j,j,q*(k-1)+1,q*k)-mean.t
                ());
          }
      }

    S=S/(n*p);

    //Calcul de S^{-1/2}
    Jacobi(S,D,V);
    for (j=1;j<=q;j=j+1) D(j,j)=pow(D(j,j),-1.0/2.0);
    Smudm=V*D*(V.t());


    //Calcul des \gras{e}_j^{(k)}
    for (j=1;j<=n;j=j+1)
      {
        for (k=1;k<=p;k=k+1)
          {
            Vecte.SubMatrix(j,j,q*(k-1)+1,q*k)=(Vecteps.SubMatrix(
                j,j,q*(k-1)+1,q*k)-mean.t())*(Smudm.t());
          }
      }

    //Calcul de la statistique, cas non seriel
    statA=0.0;

    for (l=1;l<=n;l=l+1)
      {
        for (lprim=1;lprim<=n;lprim=lprim+1)
          {

            prodA=1.0;
            for (k=1;k<=CardA;k=k+1)
              {

                prodA=prodA*(exp(-b*b*0.5*(Vecte.SubMatrix(l,l,q
                    *((int)A(1,k)-1)+1,q*(int)A(1,k))-Vecte.
                    SubMatrix(lprim,lprim,q*((int)A(1,k)-1)+1,q*(
                    int)A(1,k))).SumSquare())-pow(b*b+1.0,-q
                    /2.0)*exp(-0.5*b*b/(b*b+1)*(Vecte.SubMatrix(l
                    ,l,q*((int)A(1,k)-1)+1,q*(int)A(1,k))).
                    SumSquare())-pow(b*b+1.0,-q/2.0)*exp(-0.5*b*b
                    /(b*b+1)*(Vecte.SubMatrix(lprim,lprim,q*((int
                    )A(1,k)-1)+1,q*(int)A(1,k))).SumSquare())+pow
                    (2*b*b+1,-q/2.0));
              }
            statA=statA+prodA;

          }
      }

    statA=statA/n;

    vectstatA(i)=statA;

  }

SortAscending(vectstatA);

quant=vectstatA((int)((1-alpha)*N));

cout << "\nCas non seriel.";
```

```
                    cout << "\nParametres:";
                    cout << "\nalpha=" << alpha;
                    cout << "\nN="<<N;
                    cout << "\np=" << p;
                    cout << "\nb=" << b;
                    cout << "\nq=" << q;
                    cout << "\nn=" << n;
                    cout << "\nA=" << A;
                    cout << "CardA=" << CardA;


                    cout << "\nQuantile␣d'ordre␣1-alpha=1-" << alpha << ":␣" << quant
                         <<"\n";
                    fin=time(0);
                    cout << "Temps␣de␣calcul␣en␣secondes:␣" << fin-debut <<"\n\n";

                  }
                }
              }
            }
        }
}


if (serial==1) {
  p=4;


  for (alpha=0.1;alpha>=0.05;alpha=alpha/2.0)
    {

      for (bval=1;bval<=4;bval=bval+1)
        {
          b=bvalues(bval);
          for (nval=1;nval<=3;nval=nval+1)
            {
              n=(int)nvalues(nval);

              for (q=2;q<=3;q=q+1) {

                for (CardA=2;CardA<=4;CardA=CardA+1) {


                  prodA=1.0;
                  Matrix A(1,CardA);
                  for (i=1;i<=CardA;i=i+1) A(1,i)=i;

                  Matrix Vectu(n,q);
                  Matrix Vecteps(n-p+1,p*q);
                  Matrix mean(q,1);
                  SymmetricMatrix S(q);
                  Matrix Smudm(q,q);
                  DiagonalMatrix D(q);
                  Matrix U(q,q);
                  Matrix V(q,q);
                  Matrix Vecte(n-p+1,p*q);

                  ColumnVector vectstatA(N);

                  for (i=1;i<=N;i=i+1)

                    {
```

```
//Remplissage de Vectu
for (k=1;k<=n;k=k+1)
  {
    for (l=1;l<=q;l=l+1)
      {

        Vectu(k,l)=Z.Next();
      }
  }


//Remplissage de Vecteps
for (k=1;k<=(n-p+1);k=k+1)
  {
    for (l=1;l<=p;l=l+1)
      {

        Vecteps.SubMatrix(k,k,(l-1)*q+1,l*q)=Vectu.SubMatrix(k
            +l-1,k+l-1,1,q);
      }
  }

//Calcul de mean
mean=0.0;
for (j=1;j<=n;j=j+1)
  {

    mean=mean+(Vectu.SubMatrix(j,j,1,q)).t();

  }
mean=mean/n;

//Calcul de S
S=0.0;
for (j=1;j<=n;j=j+1)
  {

    S<<S+((Vectu.SubMatrix(j,j,1,q)).t()-mean)*(Vectu.
        SubMatrix(j,j,1,q)-mean.t());

  }

S=S/n;

//Calcul de S^{-1/2}
Jacobi(S,D,V);
for (j=1;j<=q;j=j+1) D(j,j)=pow(D(j,j),-1.0/2.0);
Smudm=V*D*(V.t());


//Calcul des \gras{e}_j^{(k)}
for (j=1;j<=(n-p+1);j=j+1)
  {
    for (k=1;k<=p;k=k+1)
      {
        Vecte.SubMatrix(j,j,q*(k-1)+1,q*k)=(Vectu.SubMatrix(j+
            k-1,j+k-1,1,q)-mean.t())*(Smudm.t());
      }
  }

//Calcul de la statistique, cas seriel
statA=0.0;

for (l=1;l<=(n-p+1);l=l+1)
```

```
                    {
                      for (lprim=1;lprim<=(n-p+1);lprim=lprim+1)
                        {

                          prodA=1.0;
                          for (k=1;k<=CardA;k=k+1)
                            {

                              prodA=prodA*(exp(-b*b*0.5*(Vecte.SubMatrix(l,l,q
                                   *((int)A(1,k)-1)+1,q*(int)A(1,k))-Vecte.
                                   SubMatrix(lprim,lprim,q*((int)A(1,k)-1)+1,q*(
                                   int)A(1,k))).SumSquare())-pow(b*b+1.0,-q
                                   /2.0)*exp(-0.5*b*b/(b*b+1)*(Vecte.SubMatrix(l
                                   ,l,q*((int)A(1,k)-1)+1,q*(int)A(1,k))).
                                   SumSquare())-pow(b*b+1.0,-q/2.0)*exp(-0.5*b*b
                                   /(b*b+1)*(Vecte.SubMatrix(lprim,lprim,q*((int
                                   )A(1,k)-1)+1,q*(int)A(1,k))).SumSquare())+pow
                                   (2*b*b+1,-q/2.0));
                            }
                          statA=statA+prodA;

                        }
                      }

                    statA=statA/(n-p+1);

                    vectstatA(i)=statA;



                  }

                SortAscending(vectstatA);

                quant=vectstatA((int)((1-alpha)*N));

                cout << "\nCas seriel.";
                cout << "\nParametres:";
                cout << "\nalpha=" << alpha;
                cout << "\nN="<<N;
                cout << "\np=" << p;
                cout << "\nb=" << b;
                cout << "\nq=" << q;
                cout << "\nn=" << n;
                cout << "\nA=" << A;
                cout << "CardA=" << CardA;


                cout << "\nQuantile d'ordre 1-alpha=1-" << alpha << ": " << quant
                     <<"\n";
                fin=time(0);
                cout << "Temps de calcul en secondes: " << fin-debut <<"\n\n";

              }
            }
          }
        }
      }
    }



}
```

B.3. PUISSANCE COMPARÉE AVEC CELLE DU TEST DE WILKS

Le fichier *manova.c*

```
// DEBUT DES DEFINITIONS POUR POUVOIR UTILISER LES LIBRAIRIES
//-----------------------------------------------------------

#include <string.h>
#define WANT_STREAM                  // include.h will get stream fns
#define WANT_MATH                    // include.h will get math fns
// newmatap.h will get include.h
#include "./include/newmatap.h"              // need matrix applications

#include "./include/newmatio.h"              // need matrix output routines

#ifdef use_namespace
using namespace NEWMAT;              // access NEWMAT namespace
#endif

#ifdef use_namespace
using namespace NEWRAN;
#endif

#include "./include/newran.h" // Pour la generation de nombres aleatoires

#include<time.h>

//Pour compiler:
//g++ -Wall -O manova.c -L. -lnewmat -lnewran -o manova

int main(void)

{

  Random::Set(0.46875);
  Normal Z;

  double theta, gamma, Lambda, stat, valcrit90, valcrit95, puiss90, puiss95;

  int p, q, i, n, j, k, l, nuE, nuH, rejet90, rejet95, nbcle, CardA;

  double ourstat, prodA, b;

  int l1, l2, binc;


  // Valeurs des parametres
  q=2;
  p=2;
  CardA=2;
  nbcle=10000; // Nombre de boucles de Monte-Carlo

  RowVector bvalues(6); //Differentes valeurs de b
  bvalues(1)=0.01;
  bvalues(2)=0.05;
  bvalues(3)=0.1;
  bvalues(4)=0.5;
  bvalues(5)=1.0;
  bvalues(6)=3.0;

  prodA=1.0;
  Matrix mean(q,1);
  SymmetricMatrix S(q);
  Matrix Smudm(q,q);
  DiagonalMatrix D(q);
  Matrix U(q,q);
```

```
Matrix V(q,q);

RowVector ourvalcrit90(6);
RowVector ourvalcrit95(6);
RowVector ourpuiss90(6);
RowVector ourpuiss95(6);
RowVector ourrejet90(6);
RowVector ourrejet95(6);

//Initialisations
stat=0;
n=0;
valcrit90=0.0;
valcrit95=0.0;
ourvalcrit90=0.0;
ourvalcrit95=0.0;
theta=0.0;
gamma=0.0;



cout << "Valeur␣de␣theta?␣";
cin >> theta;

cout << "\nValeur␣de␣gamma?␣";
cin >> gamma;


// On suppose que \mu=0 (qx1), \Sigma=\gamma.I_q (qxq), \Psi=\theta.I_q (qxq).
// On veut simuler \epsilonlon_i^{(j)}=\alpha_i+\delta_i^{(j)} (qx1).
// \alpha_i suit N_q(0,\theta.I_q) (qx1).
// \delta_i^{(j)} suit N_q(0,\gamma.I_q) (qx1).

ColumnVector alpha(q);
ColumnVector delta(q);

// Matrice E:qxq
Matrix E(q,q);E=0.0;
// Matrice H:qxq
Matrix H(q,q);H=0.0;

// \epsilon^{(.)}
ColumnVector epsilonp(q);epsilonp=0.0;

// \epsilon_.^{(.)}
ColumnVector epsilonpp(q);epsilonpp=0.0;



// Taille de l'echantillon
for (n=20;n<=50;n=n+30) {

// Matrice qui contient l'echantillon des n matrices qxp.
Matrix epsilon(q,n*p);

Matrix Vecte(q,n*p);

rejet90=0;
rejet95=0;

ourrejet90=0;
ourrejet95=0;

nuE=n*(p-1);
nuH=n-1;
```

```
// Quantiles. Prob[ F(2nuH,2(nuE-1)) < valcritalpha]=1-alpha:
// Valable pour p=2 et 1-alpha=0.9
if (n==20) valcrit90=1.5220424707334;
if (n==50) valcrit90=1.29684892953112;
// Valable pour p=2 et 1-alpha=0.95
if (n==20) valcrit95=1.71668714444190;
if (n==50) valcrit95=1.39644309133888;


// Nos quantiles
if (n==20) {

    ourvalcrit90(1)=7.58279e-08;
    ourvalcrit95(1)=9.10616e-08;
    ourvalcrit90(2)=4.67959e-05;
    ourvalcrit95(2)=5.61555e-05;
    ourvalcrit90(3)=0.000718;
    ourvalcrit95(3)=0.000854;
    ourvalcrit90(4)=0.170;
    ourvalcrit95(4)=0.191;
    ourvalcrit90(5)=0.561;
    ourvalcrit95(5)=0.606;
    ourvalcrit90(6)=0.938;
    ourvalcrit95(6)=0.955;

}

 if (n==50) {

    ourvalcrit90(1)=7.7359e-08;
    ourvalcrit95(1)=9.24039e-08;
    ourvalcrit90(2)=4.77215e-05;
    ourvalcrit95(2)=5.68904e-05;
    ourvalcrit90(3)=0.000718;
    ourvalcrit95(3)=0.000883;
    ourvalcrit90(4)=0.167;
    ourvalcrit95(4)=0.192;
    ourvalcrit90(5)=0.558;
    ourvalcrit95(5)=0.603;
    ourvalcrit90(6)=0.940;
    ourvalcrit95(6)=0.954;

}



// Debut de la boucle de Monte-Carlo
for (l=1;l<=nbcle;l=l+1) {

  E=0.0;H=0.0;

  // On cree l'echantillon des epsilon

  for (i=1;i<=n;i=i+1) {

    for (k=1;k<=q;k=k+1) {alpha(k)=(sqrt(theta))*(Z.Next());}

    for (j=1;j<=p;j=j+1) {

      for (k=1;k<=q;k=k+1) {delta(k)=(sqrt(gamma))*(Z.Next());}

      // C'est \epsilon_i^{(j)}
      epsilon.Column((i-1)*p+j)=alpha+delta;
```

```
    }
  }


  // On calcule la statistique de test de Wilks

  // Calcul de epsilonpp=\overline{\epsilon}
  epsilonpp=0.0;
  for (i=1;i<=n;i=i+1) {
    for (j=1;j<=p;j=j+1) {
      epsilonpp=epsilonpp+epsilon.Column((i-1)*p+j);
    }
  }
  epsilonpp=epsilonpp/(n*p);

  for (i=1;i<=n;i=i+1) {

  // Calcul de epsilonp
    epsilonp=0.0;
    for (j=1;j<=p;j=j+1) {
    epsilonp=epsilonp+epsilon.Column((i-1)*p+j);
    }
    epsilonp=epsilonp/p;

    // Calcul de H
   H=H+(epsilonp-epsilonpp)*((epsilonp-epsilonpp).t());

    //Calcul de E
    for (j=1;j<=p;j=j+1) {
      E=E+(epsilon.Column((i-1)*p+j)-epsilonp)*((epsilon.Column((i-1)*p+j)-epsilonp
          ).t());
    }

  }

H=p*H;

// Calcul de Lambda
Lambda=(E.Determinant())/((E+H).Determinant());



// Valable si q=2

if (q==2) {

  stat=((nuE-1)/nuH)*((1-sqrt(Lambda))/(sqrt(Lambda)));

}



if (stat>valcrit90) rejet90=rejet90+1;
if (stat>valcrit95) rejet95=rejet95+1;



//Calcul de mean
for (j=1;j<=q;j=j+1)
  {
    mean(j,1)=epsilonpp(j);
  }
```

```
  //Calcul de S
  S=0.0;
  for (j=1;j<=n;j=j+1)
    {
      for (k=1;k<=p;k=k+1)
        {
          S<<S+(epsilon.Column((j-1)*p+k)-mean)*((epsilon.Column((j-1)*p+k)-mean).t
               ());
        }
    }

  S=S/(n*p);

  //Calcul de S^{-1/2}
  Jacobi(S,D,V);
  for (j=1;j<=q;j=j+1) D(j,j)=pow(D(j,j),-1.0/2.0);
  Smudm=V*D*(V.t());



  //Calcul des \gras{e}_j^{(k)}
  for (j=1;j<=n;j=j+1)
    {
      for (k=1;k<=p;k=k+1)
        {
          Vecte.Column((j-1)*p+k)=Smudm*(epsilon.Column((j-1)*p+k)-mean);
        }
    }


  for (binc=1;binc<=6;binc=binc+1){

//Calcul de la statistique, cas non seriel
    ourstat=0.0; b=bvalues(binc);

    for (l1=1;l1<=n;l1=l1+1)
      {
        for (l2=1;l2<=n;l2=l2+1)
          {

            prodA=1.0;
            for (k=1;k<=CardA;k=k+1)
              {

                prodA=prodA*(exp(-b*b*0.5*((Vecte.Column((l1-1)*p+k)-Vecte.Column((l2
                     -1)*p+k)).SumSquare()))-pow(b*b+1.0,-q/2.0)*exp(-0.5*b*b/(b*b
                     +1)*((Vecte.Column((l1-1)*p+k)).SumSquare()))-pow(b*b+1.0,-q
                     /2.0)*exp(-0.5*b*b/(b*b+1)*((Vecte.Column((l2-1)*p+k)).SumSquare
                     ()))+pow(2*b*b+1,-q/2.0));
              }
            ourstat=ourstat+prodA;

          }
      }

    ourstat=ourstat/n;


  if (ourstat>ourvalcrit90(binc)) ourrejet90(binc)=ourrejet90(binc)+1;
  if (ourstat>ourvalcrit95(binc)) ourrejet95(binc)=ourrejet95(binc)+1;

                        }


  }
```

```
  // Fin de la boucle de Monte-Carlo

  // Calcul de la puissance de Wilks (ou du niveau si theta=0).
  puiss90=(rejet90*100)/double(nbcle);
  puiss95=(rejet95*100)/double(nbcle);

for (binc=1;binc<=6;binc=binc+1){

  // Calcul de notre puissance (ou du niveau si theta=0).
  ourpuiss90(binc)=(ourrejet90(binc)*100)/double(nbcle);
  ourpuiss95(binc)=(ourrejet95(binc)*100)/double(nbcle);

}

  // Affichage des resultats
  cout << "n=" << n << "\n";
  cout << "nbcle=" << nbcle << "\n";
  cout << "puiss90=" << puiss90 << "%\n";
  cout << "puiss95=" << puiss95 << "%\n";
for (binc=1;binc<=6;binc=binc+1){
  cout << "b=" << bvalues(binc) << "\n";
  cout << "ourpuiss90=" << ourpuiss90(binc) << "%\n";
  cout << "ourpuiss95=" << ourpuiss95(binc) << "%\n";
}


  }


}
```

CURRICULUM VITAE

Formation académique

- 1998-2002 PhD en statistique - Doctorat de Mathématiques Appliquées.
  Cotutelle France-Québec.
  Tests d'indépendance en analyse multivariée et tests de normalité dans les modèles ARMA.
  Directeurs de thèse : G. Ducharme et M. Bilodeau.
  Université Montpellier II et Université de Montréal.

- 1997-1998 D.E.A. de Biostatistique
  Equivalent à la 2ème année de maîtrise au Québec.
  Université Montpellier II.

- 1996-1997 Maîtrise d'Ingénierie Mathématique.
  Equivalent à la 1ère année de maîtrise au Québec.
  Université Montpellier II.

- 1994-1996 Licence de Mathématiques.
  Equivalent à la 3ème année de Baccalauréat au Québec.
  Université Montpellier II.

- 1992-1994 Deug A MP' Physique.
  Equivalent aux deux premières années de Baccalauréat au Québec.
  Université Montpellier II.

- 1991-1992 Première année de Mathématiques Supérieures.
  Lycée Joffre, Montpellier.

- 1991 Baccalauréat français série C.

Coordonnées

- Email 1 : lafaye@dms.umontreal.ca
- Email 2 : plafaye@club-internet.fr
- Site Internet 1 : http ://www.dms.umontreal.ca/∼lafaye
- Site Internet 2 : http ://perso.club-internet.fr/plafaye

DOCUMENTS SPÉCIAUX

Une disquette contenant les programmes informatiques est annexée à ce document.

RÉSUMÉ : On construit un test d'ajustement de la normalité pour les innovations d'un modèle ARMA(p, q) de tendance et moyenne connues, basé sur l'approche du test lisse dépendant des données et simple à appliquer. Une vaste étude de simulation est menée pour étudier ce test pour des tailles échantillonnales modérées. Notre approche est en général plus puissante que les tests existants. Le niveau est tenu sur la majeure partie de l'espace paramétrique. Cela est en accord avec les résultats théoriques montrant la supériorité de l'approche du test lisse dépendant des données dans des contextes similaires.
Un test d'indépendance (ou d'indépendance sérielle) semi-paramétrique entre des sous-vecteurs de loi normale est proposé, mais sans supposer la normalité jointe de ces marginales. La statistique de test est une fonctionnelle de type Cramér-von Mises d'un processus défini à partir de la fonction caractéristique empirique. Ce processus est défini de façon similaire à celui de Ghoudi et al. (2001) construit à partir de la fonction de répartition empirique et utilisé pour tester l'indépendance entre des marginales univariées. La statistique de test peut être représentée comme une V-statistique. Il est convergent pour détecter toute forme de dépendance. La convergence faible du processus est établie. La distribution asymptotique des fonctionnelles de Cramér-von Mises est approchée par la méthode de Cornish-Fisher au moyen d'une formule de récurrence pour les cumulants et par le calcul numérique des valeurs propres dans la formule d'inversion. La statistique de test est comparée avec celle de Wilks pour l'hypothèse paramétrique d'indépendance dans le modèle MANOVA à un facteur avec effets aléatoires.

MOTS-CLÉS : Processus ARMA, Bruit blanc Gaussien, Test d'ajustement, Normalité des résidus, Test lisse, Fonction caractéristique, Indépendance, Analyse multivariée, Indépendance sérielle, Processus stochastiques

TITLE : Independence tests in multivariate analysis and normality tests in ARMA models.

ABSTRACT : We build a goodness-of-fit test of normality for the innovations of an ARMA(p, q) model with known mean or trend. This test is based on the data driven smooth test approach and is simple to perform. An extensive simulation study is conducted to study the behavior of the test for moderate sample sizes. Our approach is generally more powerful than existing tests while holding its level throughout most of the parameter space. This agrees with theoretical results showing the superiority of the data driven smooth test approach in related contexts.
A semi-parametric test of independence (or serial independence) is proposed between marginal vectors each of which is normally distributed but without assuming the joint normality of these marginal vectors. The test statistic is a Cramér-von Mises functional of a process defined from the empirical characteristic function. This process is defined similarly as the process of Ghoudi et al. (2001) built from the empirical distribution function and used to test for independence between univariate marginal variables. The test statistic can be represented as a V statistic. It is consistent to detect any form of dependence. The weak convergence of the process is derived. The asymptotic distribution of the Cramér-von Mises functionals is approximated by the Cornish-Fisher expansion using a recursive formula for cumulants and by the numerical evaluations of the eigenvalues in the inversion formula. The test statistic is finally compared with Wilks' statistic for testing the parametric hypothesis of independence in the one-way MANOVA model with random effects.

KEYWORDS : ARMA process, Gaussian white noise, Normality of residuals, Smooth test, Characteristic function, Independence, Multivariate Analysis, Serial independence, Stochastic processes

DISCIPLINE : Mathématiques Appliquées : Statistique

LABORATOIRES : Laboratoire de probabilités et statistique - Département des sciences mathématiques - Université Montpellier II - Place Eugène Bataillon, 34095 Montpellier Cedex 5, France ET Département de mathématiques et de statistique - Université de Montréal - CP 6128 succ. Centre-Ville, Montréal QC H3C 3J7, Canada