Algorithm AS 197: A Fast Algorithm for the Exact Likelihood of Autoregressive-Moving Average Models

G. Melard

Applied Statistics, Vol. 33, No. 1 (1984), 104-114.

Stable URL:
http://links.jstor.org/sici?sici=0035-9254%281984%2933%3A1%3C104%3AAA1AFA%3E2.0.CO%3B2-3

Applied Statistics is currently published by Royal Statistical Society.

Algorithm AS 197

A Fast Algorithm for the Exact Likelihood of Autoregressive-moving Average Models

By G. Mélard

Université Libre de Bruxelles, Belgium

[Received July 1982. Revised June 1983]

Keywords: Maximum likelihood; Autoregressive-moving average model; Fast algorithm

Language

Fortran 66

Description and Purpose

This algorithm has the same purpose as Algorithm AS 154 of Gardner *et al.* (1980), namely to compute the exact likelihood function of a stationary autoregressive-moving average (*ARMA*) process of order (p, q). That algorithm appears to be slower, and requires more storage than is necessary, particularly for large p and q. The computer program described here is a combination of an improved version of an algorithm due to Pearlman (1980) with the quick recursion switching suggested by Gardner *et al.* (1980) and an algorithm of Wilson (1979). The program is extremely efficient both in terms of computing time and amount of storage.

Theory and Method

We want to compute the likelihood function of the *ARMA* (p, q) process, defined by the equation

$$w_t = \phi_1 w_{t-1} + \ldots + \phi_p w_{t-p} + a_t - \theta_1 a_{t-1} - \ldots - \theta_q a_{t-q} \tag{1}$$

associated to a time series $(w_t; t = 1, \ldots, n)$, under the assumptions that the a_t are normally and independently distributed with zero mean and constant variance σ^2 and that the process is stationary. The basic principle consists (Ansley, 1979; Harvey and Phillips, 1979) of computing the values taken by the innovations $\hat{a}_t$ of the stochastic process $(w_t;\ t = 1, 2, \ldots)$. The likelihood is then given by the expression

$$(2\pi)^{-n/2} \left(\prod_{t=1}^{n} \sigma_t \right)^{-1} \exp \left\{ -\frac{1}{2} \sum_{t=1}^{n} (\hat{a}_t/\sigma_t)^2 \right\}, \tag{2}$$

where $\sigma_t = h_t \sigma$ is the standard deviation of $\hat{a}_t$. Maximizing (2) with respect to the parameters included in ϕ and θ is equivalent (Ansley, 1979) to minimizing the sum of squares

$$\left(\prod_{t=1}^{n} h_t^2 \right)^{1/n} \sum_{t=1}^{n} \left(\frac{\hat{a}_t}{h_t} \right)^2. \tag{3}$$

The maximum likelihood estimate of σ^2 is then given by $n^{-1} \Sigma (\hat{a}_t/h_t)^2$, evaluated at the optimal parameter point. Subroutine *FLIKAM* can be used to compute (3) as the product *FACT*SUMSQ*.

One method of obtaining the $\hat{a}_t$ (Caines and Rissanen, 1974; Gardner *et al.*, 1980) is to use the Kalman filter recursions based on a state space representation of (1), e.g.

Present address: Institut de Statistique – C.P. 210, Université Libre de Bruxelles, Bruxelles, Belgium.

 0035–9254/84/33104 $2.00

$$w_t = HW_t,$$

$$W_t = FW_{t-1} + Ga_t, \tag{4}$$

where W_t is the $r \times 1$ state vector, $r = \max(p, q+1)$, $H = (1\ 0 \ldots 0)$, $G' = (1 - \theta_1 \ldots - \theta_{r-1})$, $F = (F_{i,j})$ is an $r \times r$ matrix such that $F_{i,1} = \phi_i$ and $F_{i,j} = \delta_{i,j+1}$ $(j = 2, \ldots, r)$ for $i = 1, 2, \ldots, r$, with the notations $\phi_i = 0$, $i > p$, $\theta_0 = -1$ and $\theta_i = 0$, $i > q$. Pearlman (1980) has suggested replacing the (matrix) Ricatti-type difference equation used in the Kalman filter by a (vector) Chandrasekhar-type difference equation, see Rissanen (1973), Lindquist (1974) and Morf *et al.* (1974). His first algorithm consists of the recursions

$$\hat{a}_t = w_t - H\hat{W}_t, \tag{5}$$

$$\hat{W}_{t+1} = F\hat{W}_t + K_t(\hat{a}_t/h_t^2), \tag{6}$$

$$K_{t+1} = K_t - \alpha_t FL_t, \tag{7}$$

$$L_{t+1} = FL_t - \alpha_t K_t, \tag{8}$$

$$h_{t+1}^2 = h_t^2(1 - \alpha_t^2), \tag{9}$$

where α_t is written for HL_t/h_t^2. The same idea was implicit in the paper of Caines and Rissanen (1974, p. 103, footnote).

Our implementation of this algorithm includes an improvement for the case where $p > q$. Indeed, (5-6) implies that

$$w_t - \sum_{j=1}^{p(t)} \phi_j w_{t-j} = \hat{a}_t - \sum_{j=1}^{p(t)} \phi_j \hat{a}_{t-j} + \sum_{j=1}^{q(t)} \frac{K_{t-j,j}}{h_{t-j}^2} \hat{a}_{t-j}, \tag{10}$$

where $p(t) = \min(t-2, p)$, $q(t) = \min(t-1, r)$ and $K_{t-j,j}$ is the jth element of K_{t-j}. Since $\hat{a}_t$ is the innovation at time t of the stochastic process $(w_t; t = 1, 2, \ldots)$, the right-hand side of (10) is, for $t \geqslant p+2$, the innovations representation (Cramér, 1961) of a non-stationary moving average process of order q which is known to be unique. Hence the terms for $j > q$ vanish, so that $K_{t-j,j}/h_{t-j}^2 = \phi_j$. Consequently, the jth elements of (K_t/h_t^2), $j = q+1, \ldots, r$, do not change when $t \geqslant p - q + 1$. Updating of these elements of K_t and L_t by (7)-(8) can be skipped over and the ratio $(K_{t,j}/h_t^2)$ must accordingly be replaced by ϕ_j in (6) for $j \geqslant p+1$. Note that Pearlman (1980) states a similar property for his second algorithm (erroneously for $t \geqslant q+1$ instead of $t \geqslant p - q + 1$). The present variant of his first algorithm, in terms of the total number of multiplications and divisions, is as fast as these two algorithms, uniformly in p and q.

The starting conditions for (5)-(9) are $\hat{W}_1 = 0$, $K_1 = L_1 = FPH'$, $h_1^2 = HPH'$, where P is the covariance matrix in the marginal distribution of W_1. Since

$$W_{t,i} = \sum_{j=i}^{r} (\phi_j w_{t-j+i-1} - \theta_{j-1} a_{t-j+i}),$$

we have $P_{1,1} = \gamma_0$ and $P_{i,1} = \mu_i$, where

$$\mu_i = \sum_{j=i}^{r} (\phi_j \gamma_{j-i+1} - \theta_{j-1} \lambda_{j-i}), \quad i = 1, \ldots, r$$

and

$$\sigma^2 \gamma_k = \operatorname{cov}(w_t, w_{t-k}), \quad \sigma^2 \lambda_k = \operatorname{cov}(w_t, a_{t-k}), \quad k = 0, 1, \ldots.$$

Hence

$$K_{1,i} = L_{1,i} = \phi_i \gamma_0 + \mu_{i+1}, \tag{11}$$

where we let $\mu_{r+1} = 0$. Note that Gardner *et al.* (1980) mention the suggestion from a referee that the autocovariances be used in the calculation of P.

The autocovariances γ_k $(k = 0, 1, \ldots, R)$, where $R = \max(p, q)$, are determined by an algorithm due to Wilson (1979). The covariances λ_k $(k = 0, 1, \ldots, q)$ are then given by the formula

$$\lambda_k = -\theta_k + \sum_{j=1}^{\min(p,\, k)} \phi_j \lambda_{q-j}.$$

The γ_k and λ_k are obtained by using subroutine *TWACF*.

We have retained the proposal of Gardner *et al.* (1980) to allow for a switching from the state space recursions (5)-(9) to the quick recursions

$$\hat{a}_t = w_t - \sum_{j=1}^{p} \phi_j w_{t-j} + \sum_{j=1}^{q} \theta_j \hat{a}_{t-j}, \tag{12}$$

$$h^{2*}_{t+1} = 1$$

as soon as $h^2_t < 1 + \delta$, where δ is a small positive real number. The switching has been delayed until $t \geqslant r + 1$ in order to avoid unnecessary complications. For the same reason, unnecessary updating of some elements of K_t and L_t has been maintained until $t \geqslant p - q + 1$. The switching always occurs at time $p + 1$ for pure autoregressive processes.

Note also that working storage is restricted to three vectors of length $r + 1$, which is really negligible by comparison with the n^2 memory cells required by the direct inversion of the covariance matrix. Our implementation of Wilson's algorithm uses the same three vectors as workspace without the need to reconstruct some coefficients (compare with Wilson, 1979, p. 303).

The algorithm can still be improved in the case of a seasonal moving average process, defined by the equation

$$w_t = \theta(B)\, \Theta(B^s)\, a_t, \tag{13}$$

where B is the backshift operator, $\theta(B)$ is a polynomial in B of degree q', $\Theta(B^s)$ is a polynomial in B^s of degree q'' and s is the length of the seasonal cycle, such that $q' < s$. Let $\sigma^2 \Omega$ be the covariance matrix of $w = (w_1, \ldots, w_n)'$ and $\Omega = TT'$, the Cholesky factorization of Ω, where T is a lower triangular matrix. The elements T_{ij} of T are related to the elements $K_{t,j}$ of vectors K_t by $T_{t,t-j} = K_{t-j,j}/h_{t-j}$. Consequently by Theorem 4.1 of Ansley (1979), we have $K_{t,j} = 0$ for $t = h's + k - j$ and $j = h''s + l$ with $h', h'' = 0, 1, \ldots, k = 1, \ldots, s - q'$, and $l = q' + 1, \ldots, s - 1$. There is no simple generalization of this property when $p > 0$, except when the autoregressive operator $1 - \phi_1 B - \ldots - \phi_p B^p$ is a polynomial in B^s. A transformation like the one suggested by Ansley (1979, p. 64) can be used for mixed models but, since the algorithm is restricted to a time-invariant state space representation (4), the Kalman filter algorithm would become necessary. These refinements for seasonal models are not implemented in subroutine *FLIKAM*.

Structure

SUBROUTINE FLIKAM(P, MP, Q, MQ, W, E, N, SUMSQ, FACT, VW, VL, MRP1, VK, MR, TOLER, IFAULT)

Formal parameters

P	Real array (*MP*)	input: the value of ϕ in the first p locations
MP	Integer	input: the value of p

Q	Real array (*MQ*)	input:	the value of θ in the first q locations
MQ	Integer	input:	the value of q
W	Real array (*N*)	input:	the observations, w_t
E	Real array (*N*)	output:	the corresponding residuals $\hat{a}_t/\sigma_t$
N	Integer	input:	n, the number of observations
SUMSQ	Real	output:	the value of $\Sigma\,(\hat{a}_t/h_t)^2$
FACT	Real	output:	the value of $(\Pi\, h_t^2)^{1/n}$
VW	Real array (*MRP*1)	workspace:	used to store the state vector W_t
VL	Real array (*MRP*1)	workspace:	used to store vector L_t
*MRP*1	Integer	input:	the value of max $(p, q + 1) + 1$
VK	Real array (*MR*)	workspace:	used to store vector K_t
MR	Integer	input:	the value of max $(p, q + 1)$
TOLER	Real	input:	the value of δ. It should be negative if the exact likelihood is desired. Otherwise, switching to approximate recursions occurs when $h_t^2 < 1 + \delta$
IFAULT	Integer	output:	a fault indicator, equal to
			1 to 5 indicates an error detected in subroutine *TWACF* (see below)
			6 if $MR \neq \max(MP, MQ + 1)$
			7 if $MRP1 \neq MR + 1$
			8 if $h_t^2 \leqslant 10^{-10}$ This indicates bad numerical behaviour. Check the coefficients ϕ and θ
			9 if $N \leqslant 0$
			$-m$ not a failure: indicates that quick recursions took place from $t = m$
			0 otherwise

Constant

The constant 10^{-10} used in the eighth failure test is identified by variable *EPSIL*1, which is *DATA*-initialized.

SUBROUTINE TWACF(*P, MP, Q, MQ, ACF, MA, CVLI, MXPQP*1, *ALPHA, MXPQ, IFAULT*)

Formal parameters

P	Real array (*MP*)	input:	the value of ϕ in the first p locations
MP	Integer	input:	the value of p
Q	Real array (*MQ*)	input:	the value of θ in the first q locations
MQ	Integer	input:	the value of q
ACF	Real array (*MA*)	output:	the autocovariances of order 0 to $MA - 1$
MA	Integer	input:	the maximum lag in the autocovariances plus 1
CVLI	Real array (*MXPQP*1)	output:	the covariances between w_t and a_{t-k} for $k = 0, 1, \ldots, MXPQP1 - 1$
*MXPQP*1	Integer	input:	the value of max $(p, q) + 1$
ALPHA	Real array (*MXPQ*)	workspace	
MXPQ	Integer	input:	the value of max (p, q)
IFAULT	Integer	output:	a fault indicator, equal to
			1 if $MP < 0$ or $MQ < 0$
			2 if $MXPQ \neq \max(MP, MQ)$
			3 if $MXPQP1 \neq MXPQ + 1$
			4 if $MA < MXPQP1$
			5 if, for some k, $(ALPHA(k+1))^2 \geqslant 1 - 10^{-10}$, indicating that the nonstationarity boundary is too close.
			0 otherwise

Constant

The constant 10^{-10} used in the fifth failure test is known as variable *EPSIL2*, and is *DATA*-initialized.

Precision

On machines with small word length, all the real variables should be replaced by double precision variables. Overflow or underflow will not occur in the calculation of $(\Pi h_t^2)^{1/n}$ because the product is stored in the form $a2^b$ (Martin and Wilkinson, 1965; Ansley, 1979).

Time

The number of time-consuming operations—multiplications and divisions—is given by the formula

$$N_n(p, q) = N_0(p, q) + N(n, p, q),$$

where

$$N_0(p, q) = p^2 + \frac{q^2}{2} + 2pR + qS + \frac{R^2}{2},$$

$$N(n, p, q) = n(p + 3q + S),$$

where $R = \max(p, q)$, and $S = \min(p, q)$. Terms of lower power have been omitted. Note that the use of McLeod's (1975) algorithm would have given a term $O(p^3/2)$ in $N_0(p, q)$. The approximate conditional and unconditional methods (Box and Jenkins, 1976) correspond respectively to

$$N'(n, p, q) = n(p + q),$$

$$N''(n, p, q) = \{2n + 2\nu + (k - 1)(2n + 4\nu)\} \quad (p + q),$$

where ν is the maximum leadtime for backforecasting or forecasting and k is the number of iterations of the backforecasting procedure. With $\nu = n/2$ and $k = 1$, we obtain $N''(n, p, q) = 3n(p + q)$, i.e. more than $N(n, p, q)$ for the exact method. Table 1 shows the average computation

TABLE 1

Average computation times in milliseconds required to evaluate the conditional sum of squares (by the code used by Gardner et al., and that in the comment in FLIKAM), and the exact likelihood (by a method of Ansley improved, see Mélard, 1982, by the Algorithm AS 154 of Gardner et al. and the present algorithm)

Model	*Conditional method*		*Exact method*		
(p, q)	AS 154	*Comment in FLIKAM*	*Ansley (improved)*	AS 154	AS 197
(1, 0)	1.1	0.7	0.7	4.1	0.6
(0, 1)	1.2	0.8	3.6	4.8	3.1
(2, 0)	1.2	0.8	0.9	5.4	0.8
(0, 2)	1.6	0.8	4.6	5.9	3.3
(1, 1)	1.5	0.9	4.1	5.4	3.3
(13, 0)	2.7	1.9	12	77	3.4
(12, 1)	3.1	1.9	13	61	5.6
(1, 12)	4.4	1.9	19	75	5.2
(0, 13)	4.6	1.9	21	37	5.0

times in milliseconds for execution on a CDC Cyber 170-750 computer using the FTN compiler with optimization option 2, for series of length $n = 100$. The ratio of the computing times of the

exact likelihood over the conditional approximation is not greater than 4 for most of the models which have been tried and is less than 3 for the high-order models. If the quick recursion switching is allowed for still better results can be obtained when $q > 0$. For a more complete discussion of this point, see Gardner *et al.* (1980). We have given the code used for the conditional method in the comment near the end of *SUBROUTINE FLIKAM*. The code given in the program of Gardner *et al.* (1980) was also considered. We recall that even ratios can be highly dependent on the computer and on the compiler, especially its level of optimization (see Mélard, 1982). Anyway, the code was written in order to possibly take advantage of compiler optimization, in accordance with the recommendations of, for example, Kernighan and Plauger (1978).

If computation of the ratio $\hat{a}_t/h_t$ is not necessary (e.g. when a general purpose optimization algorithm is used instead of a non-linear least-squares algorithm) the algorithm can be slightly modified in order to avoid all square roots except q of them when switching to quick recursions occurs.

Related Algorithms

The number of multiplications and divisions required by the original algorithms of Pearlman (1980) is always as large as ours. Pearlman (1980) pointed out that algorithms based on the Morf *et al.* recursions are not necessarily faster than the algorithm of Ansley (1979) based on a Cholesky factorization of a band matrix. Between the original Ansley's algorithm, the improved version (Mélard, 1982) and the Kalman filter algorithm of Gardner *et al.*, (1980), the second one is faster when $p > q$ whereas the third one requires about $p^2r^4/2$ operations for the determination of the starting matrix P. The storage requirements are respectively rn, r^2 and $r^4/8$. For *ARMA* models with time-dependent coefficients, an algorithm has been given by Mélard (1982).

Execution times of the programs on series of length 100 in the experiments reported in Table 1 seem to confirm that the program of Gardner *et al.* (1980) should not be used. The algorithm of Ansley improved (with the algorithm of McLeod, to tell the truth) is sometimes nearly as fast as the present algorithm except for high-order models, confirming the conclusions of Pearlman (1980).

The possible improvements for seasonal moving average processes (13) would reduce the approximate number of multiplications and divisions at each time from $3(q''s + q')$ to $3q''(1 + 2q')$.

Acknowledgements

This paper was written while at the "Université de Montréal" under the sponsorship of the international co-operation between Belgium and the Province of Quebec.

We thank the Algorithm Editor and the referee, especially for suggesting the examination of models with sparse coefficients.

References

Ansley, C. F. (1979) An algorithm for the exact likelihood of a mixed autoregressive-moving average process. *Biometrika*, **66**, 59–65.

Box, G. E. P. and Jenkins, G. M. (1976) *Time Series Analysis, Forecasting and Control* (revised edition). San Francisco: Holden Day.

Caines, P. E. and Rissanen, J. (1974) Maximum likelihood estimation of parameters in multivariate Gaussian stochastic processes. *IEEE Trans. Inf. Theory*, **IT-20**, 102–104.

Cramér, H. (1961) On some classes of nonstationary stochastic processes. In *Proc. 4th Berkeley Symp. Math. Statist. and Prob.*, Vol. 2, pp. 57–78. Berkeley and Los Angeles: University of California Press.

Gardner, G., Harvey, A. C. and Phillips, G. D. A. (1980) Algorithm AS 154. An algorithm for exact maximum likelihood estimation of autoregressive-moving average models by means of Kalman filtering. *Appl. Statist.*, **29**, 311–322.

Harvey, A. C. and Phillips, G. D. A. (1979) Maximum likelihood estimation of regression models with autoregressive-moving average disturbances. *Biometrika*, **66**, 49–58.

Kernighan, B. W. and Plauger, P. J. (1978) *The Elements of Programming Style*. New York: McGraw-Hill.

Lindquist, A. (1974) A new algorithm for optimal filtering of discrete-time stationary processes. *Siam J. Control*, **12**, 736–746.

McLeod, I. (1975) Derivation of the theoretical autocovariance function of autoregressive-moving average time series. *Appl. Statist.*, **24**, 255–256.

Martin, R. S. and Wilkinson, J. H. (1965) Symmetric decomposition of positive definite band matrices. *Num. Math.*, 7, 355–361.

Mélard, G. (1982) The likelihood function of a time-dependent *ARMA* model. In *Applied Time Series Analysis, Proceeding of the International Conference held at Houston, Texas, August, 1981* (O. D. Anderson and M. R. Perryman, eds). Amsterdam: North-Holland, to appear.

Morf, M., Sidhu, G. S. and Kailath, T. (1974) Some new algorithms for recursive estimation on constant, linear, discrete-time systems. *IEEE Trans. Auto. Control*, **AC-19**, 315–323.

Pearlman, J. G. (1980) An algorithm for the exact likelihood of a high-order autoregressive-moving average process. *Biometrika*, **67**, 232–233.

Rissanen, J. (1973) Algorithms for triangular decomposition of block Hankel and Toeplitz matrices with application to factoring positive matrix polynomials. *Math. Comp.*, **27**, 147–154.

Tunnicliffe, Wilson, G. (1979) Some efficient computational procedures for high order *ARMA* models. *J. Statist. Comp. Simul.*, **8**, 301–309.

```
      SUBROUTINE FLIKAM(P, MP, Q, MQ, W, E, N, SUMSQ, FACT, VW, VL,
     *  MRP1, VK, MR, TOLER, IFAULT)
C
C        ALGORITHM AS 197  APPL. STATIST. (1984) VOL.33, NO.1
C
C        COMPUTES THE LIKELIHOOD FUNCTION OF AN AUTOREGRESSIVE-
C        MOVING AVERAGE PROCESS, EXPRESSED AS FACT*SUMSQ.
C
      REAL P(MP), Q(MQ), W(N), E(N), VW(MRP1), VL(MRP1), VK(MR)
C
      REAL FACT, SUMSQ, TOLER, EPSIL1, ZERO, P0625, ONE, TWO, FOUR,
     *  SIXTEN, A, ALF, AOR, DETCAR, DETMAN, FLJ, FN, R, VL1, VW1
C
      REAL ABS, SQRT
C
      DATA EPSIL1 /1.0E-10/
      DATA ZERO, P0625, ONE, TWO, FOUR, SIXTEN /0.0, 0.0625, 1.0, 2.0,
     *  4.0, 16.0/
C
      FACT = ZERO
      DETMAN = ONE
      DETCAR = ZERO
      SUMSQ = ZERO
      MXPQ = MAXO(MP, MQ)
      MXPQP1 = MXPQ + 1
      MQP1 = MQ + 1
      MPP1 = MP + 1
C
C        CALCULATION OF THE AUTOCOVARIANCE FUNCTION OF A PROCESS WITH
C        UNIT INNOVATION VARIANCE (VW) AND THE COVARIANCES BETWEEN THE
C        VARIABLE AND THE LAGGED INNOVATIONS (VL).
C
      CALL TWACF(P, MP, Q, MQ, VW, MXPQP1, VL, MXPQP1, VK, MXPQ, IFAULT)
      IF (MR .NE. MAXO(MP, MQP1)) IFAULT = 6
      IF (MRP1 .NE. MR + 1) IFAULT = 7
      IF (IFAULT .GT. 0) RETURN
C
C        COMPUTATION OF THE FIRST COLUMN OF MATRIX P (VK)
C
      VK(1) = VW(1)
      IF (MR .EQ. 1) GOTO 150
      DO 140 K = 2, MR
      VK(K) = ZERO
      IF (K .GT. MP) GOTO 120
      DO 110 J = K, MP
      JP2MK = J + 2 - K
      VK(K) = VK(K) + P(J) * VW(JP2MK)
  110 CONTINUE
  120 IF (K .GT. MQP1) GOTO 140
      DO 130 J = K, MQP1
      JP1MK = J + 1 - K
      VK(K) = VK(K) - Q(J - 1) * VL(JP1MK)
  130 CONTINUE
  140 CONTINUE
```

```
C         COMPUTATION OF THE INITIAL VECTORS L AND K (VL,VK).
C
  150 R = VK(1)
      VL(MR) = ZERO
      DO 160 J = 1, MR
      VW(J) = ZERO
      IF (J .NE. MR) VL(J) = VK(J + 1)
      IF (J .LE. MP) VL(J) = VL(J) + P(J) * R
      VK(J) = VL(J)
  160 CONTINUE
C
C         INITIALIZATION
C
      LAST = MPP1 - MQ
      LOOP = MP
      JFROM = MPP1
      VW(MPP1) = ZERO
      VL(MXPQP1) = ZERO
C
C         EXIT IF NO OBSERVATION, OTHERWISE LOOP ON TIME.
C
      IF (N .LE. 0) GOTO 500
      DO 290 I = 1, N
C
C         TEST FOR SKIPPED UPDATING
C
      IF (I .NE. LAST) GOTO 170
      LOOP = MIN0(MP, MQ)
      JFROM = LOOP + 1
C
C         TEST FOR SWITCHING
C
      IF (MQ .LE. 0) GOTO 300
  170 IF (R .LE. EPSIL1) GOTO 400
      IF (ABS(R - ONE) .LT. TOLER .AND. I .GT. MXPQ) GOTO 300
C
C         UPDATING SCALARS
C
      DETMAN = DETMAN * R
  190 IF (ABS(DETMAN) .LT. ONE) GOTO 200
      DETMAN = DETMAN * P0625
      DETCAR = DETCAR + FOUR
      GOTO 190
  200 IF (ABS(DETMAN) .GE. P0625) GOTO 210
      DETMAN = DETMAN * SIXTEN
      DETCAR = DETCAR - FOUR
      GOTO 200
  210 VW1 = VW(1)
      A = W(I) - VW1
      E(I) = A / SQRT(R)
      AOR = A / R
      SUMSQ = SUMSQ + A * AOR
      VL1 = VL(1)
      ALF = VL1 / R
      R = R - ALF * VL1
      IF (LOOP .EQ. 0) GOTO 230
C
C         UPDATING VECTORS
C
      DO 220 J = 1, LOOP
      FLJ = VL(J + 1) + P(J) * VL1
      VW(J) = VW(J + 1) + P(J) * VW1 + AOR * VK(J)
      VL(J) = FLJ - ALF * VK(J)
      VK(J) = VK(J) - ALF * FLJ
  220 CONTINUE
  230 IF (JFROM .GT. MQ) GOTO 250
      DO 240 J = JFROM, MQ
      VW(J) = VW(J + 1) + AOR * VK(J)
      VL(J) = VL(J + 1) - ALF * VK(J)
      VK(J) = VK(J) - ALF * VL(J + 1)
  240 CONTINUE
  250 IF (JFROM .GT. MP) GOTO 270
```

```
      DO 260 J = JFROM, MP
  260 VW(J) = VW(J + 1) + P(J) * W(I)
  270 CONTINUE
  290 CONTINUE
      GOTO 390
C
C        QUICK RECURSIONS
C
  300 NEXTI = I
      IFAULT = -NEXTI
      DO 310 I = NEXTI, N
  310 E(I) = W(I)
      IF (MP .EQ. 0) GOTO 340
      DO 330 I = NEXTI, N
      DO 320 J = 1, MP
      IMJ = I - J
      E(I) = E(I) - P(J) * W(IMJ)
  320 CONTINUE
  330 CONTINUE
  340 IF (MQ .EQ. 0) GOTO 370
      DO 360 I = NEXTI, N
      DO 350 J = 1, MQ
      IMJ = I - J
      E(I) = E(I) + Q(J) * E(IMJ)
  350 CONTINUE
  360 CONTINUE
C
C        RETURN SUM OF SQUARES AND DETERMINANT
C
  370 DO 380 I = NEXTI, N
  380 SUMSQ = SUMSQ + E(I) * E(I)
C
C        CODE FOR CONDITIONAL SUM OF SQUARES
C        REPLACES ALL EXECUTABLE STATEMENTS UPTO AND
C        INCLUDING THAT LABELLED 380
C
C              FACT = ZERO
C              DETMAN = ONE
C              DETCAR = ZERO
C              SUMSQ = ZERO
C              MXPQ = MAXO(MP, MQ)
C              DO 380 I=MXPQ,N
C              E(I)=W(I)
C              IF (MP.LE.0) GOTO 340
C              DO 320 J=1,MP
C              IMJ=I-J
C              E(I)=E(I)-P(J)*W(IMJ)
C          320 CONTINUE
C          340 IF (MQ.LE.0) GOTO 380
C              DO 350 J=1,MQ
C              IMJ=I-J
C              E(I)=E(I)+Q(J)*E(IMJ)
C          350 CONTINUE
C          380 SUMSQ=SUMSQ+E(I)*E(I)
C
  390 FN = N
      FACT = DETMAN ** (ONE / FN) * TWO ** (DETCAR / FN)
      RETURN
C
C        EXECUTION ERRORS
C
  400 IFAULT = 8
      RETURN
  500 IFAULT = 9
      RETURN
      END
C
      SUBROUTINE TWACF(P, MP, Q, MQ, ACF, MA, CVLI, MXPQP1, ALPHA, MXPQ,
     *  IFAULT)
C
C        ALGORITHM AS 197.1  APPL. STATIST. (1984) VOL.33, NO.1
```

```
C         IMPLEMENTATION OF THE ALGORITHM OF G. TUNNICLIFFE WILSON
C         (J. STATIST. COMPUT. SIMUL. 8, 1979, 301-309) FOR THE
C         COMPUTATION OF THE AUTOCOVARIANCE FUNCTION (ACF) OF AN ARMA
C         PROCESS OF ORDER (MP,MQ) AND UNIT INNOVATION VARIANCE.
C         THE AUTOREGRESSIVE AND MOVING AVERAGE COEFFICIENTS ARE STORED
C         IN VECTORS P AND Q, USING BOX AND JENKINS NOTATION. ON OUTPUT
C         VECTOR CVLI CONTAINS THE COVARIANCES BETWEEN THE VARIABLE AND
C         THE (K-1)-LAGGED INNOVATION, FOR K=1,...,MQ+1.
C
      REAL P(MP), Q(MQ), ACF(MA), CVLI(MXPQP1), ALPHA(MXPQ)
C
      REAL EPSIL2, ZERO, HALF, ONE, TWO, DIV
C
      DATA EPSIL2 /1.0E-10/
      DATA ZERO, HALF, ONE, TWO /0.0, 0.5, 1.0, 2.0/
C
      IFAULT = 0
      IF (MP .LT. 0 .OR. MQ .LT. 0) IFAULT = 1
      IF (MXPQ .NE. MAX0(MP, MQ)) IFAULT = 2
      IF (MXPQP1 .NE. MXPQ + 1) IFAULT = 3
      IF (MA .LT. MXPQP1) IFAULT = 4
      IF (IFAULT .GT. 0) RETURN
C
C         INITIALIZATION AND RETURN IF MP=MQ=0
C
      ACF(1) = ONE
      CVLI(1) = ONE
      IF (MA .EQ. 1) RETURN
      DO 10 I = 2, MA
   10 ACF(I) = ZERO
      IF (MXPQP1 .EQ. 1) RETURN
      DO 20 I = 2, MXPQP1
   20 CVLI(I) = ZERO
      DO 90 K = 1, MXPQ
   90 ALPHA(K) = ZERO
C
C         COMPUTATION OF THE A.C.F. OF THE MOVING AVERAGE PART,
C         STORED IN ACF.
C
      IF (MQ .EQ. 0) GOTO 180
      DO 130 K = 1, MQ
      CVLI(K + 1) = -Q(K)
      ACF(K + 1) = -Q(K)
      KC = MQ - K
      IF (KC .EQ. 0) GOTO 120
      DO 110 J = 1, KC
      JPK = J + K
      ACF(K + 1) = ACF(K + 1) + Q(J) * Q(JPK)
  110 CONTINUE
  120 ACF(1) = ACF(1) + Q(K) * Q(K)
  130 CONTINUE
C
C         INITIALIZATION OF CVLI = T.W.-S PHI -- RETURN IF MP=0.
C
  180 IF (MP .EQ. 0) RETURN
      DO 190 K = 1, MP
      ALPHA(K) = P(K)
      CVLI(K) = P(K)
  190 CONTINUE
C
C         COMPUTATION OF T.W.-S ALPHA AND DELTA
C         (DELTA STORED IN ACF WHICH IS GRADUALLY OVERWRITTEN)
C
      DO 290 K = 1, MXPQ
      KC = MXPQ - K
      IF (KC .GE. MP) GOTO 240
      DIV = ONE - ALPHA(KC + 1) * ALPHA(KC + 1)
      IF (DIV .LE. EPSIL2) GOTO 700
      IF (KC .EQ. 0) GOTO 290
      DO 230 J = 1, KC
      KCP1MJ = KC + 1 - J
      ALPHA(J) = (CVLI(J) + ALPHA(KC + 1) * CVLI(KCP1MJ)) / DIV
```

```
  230 CONTINUE
  240 IF (KC .GE. MQ) GOTO 260
      J1 = MAXO(KC + 1 - MP, 1)
      DO 250 J = J1, KC
      KCP1MJ = KC + 1 - J
      ACF(J + 1) = ACF(J + 1) + ACF(KC + 2) * ALPHA(KCP1MJ)
  250 CONTINUE
  260 IF (KC .GE. MP) GOTO 290
      DO 270 J = 1, KC
  270 CVLI(J) = ALPHA(J)
  290 CONTINUE
C
C         COMPUTATION OF T.W.-S NU
C         (NU IS STORED IN CVLI, COPIED INTO ACF)
C
      ACF(1) = HALF * ACF(1)
      DO 330 K = 1, MXPQ
      IF (K .GT. MP) GOTO 330
      KP1 = K + 1
      DIV = ONE - ALPHA(K) * ALPHA(K)
      DO 310 J = 1, KP1
      KP2MJ = K + 2 - J
      CVLI(J) = (ACF(J) + ALPHA(K) * ACF(KP2MJ)) / DIV
  310 CONTINUE
      DO 320 J = 1, KP1
  320 ACF(J) = CVLI(J)
  330 CONTINUE
C
C         COMPUTATION OF ACF (ACF IS GRADUALLY OVERWRITTEN)
C
      DO 430 I = 1, MA
      MIIM1P = MINO(I - 1, MP)
      IF (MIIM1P .EQ. 0) GOTO 430
      DO 420 J = 1, MIIM1P
      IMJ = I - J
      ACF(I) = ACF(I) + P(J) * ACF(IMJ)
  420 CONTINUE
  430 CONTINUE
      ACF(1) = ACF(1) * TWO
C
C         COMPUTATION OF CVLI - RETURN WHEN MQ=0
C
      CVLI(1) = ONE
      IF (MQ .LE. 0) GOTO 600
      DO 530 K = 1, MQ
      CVLI(K + 1) = -Q(K)
      IF (MP .EQ. 0) GOTO 530
      MIKP = MINO(K, MP)
      DO 520 J = 1, MIKP
      KP1MJ = K + 1 - J
      CVLI(K + 1) = CVLI(K + 1) + P(J) * CVLI(KP1MJ)
  520 CONTINUE
  530 CONTINUE
C
  600 RETURN
C
C         EXECUTION ERROR DUE TO (NEAR) NON-STATIONARITY
C
  700 IFAULT = 5
      RETURN
      END
```