Algorithm AS 191

An Algorithm for Approximate Likelihood Calculation of ARMA and Seasonal ARMA Models

By A. I. McLEOD and P. R. HOLANDA SALES

University of Western Ontario, Canada

Eletrobrás, Rio de Janeiro, Brazil 20080

[Received November 1982]

Keywords: AUTOREGRESSIVE-MOVING AVERAGE MODEL; MAXIMUM LIKELIHOOD ESTIMATION; MODIFIED CHOLESKY DECOMPOSITION; MULTIPLICATIVE SEASONAL ARMA; TEST FOR STATIONARITY AND INVERTIBILITY

LANGUAGE

Fortran 66

DESCRIPTION AND PURPOSE

The algorithm *SARMAS* calculates an approximation to the likelihood function of the multiplicative seasonal autoregressive-moving average (*SARMA*) model (Box and Jenkins, 1976). The conditional, unconditional or iterated unconditional method of Box and Jenkins (1976) may be used in *SARMAS* in conjunction with an approximation to the determinant term (McLeod, 1977, 1982) to obtain an accurate and highly efficient algorithm. In fact, it may be pointed out that other algorithms, such as AS 154 (Gardner, Harvey and Phillips, 1980) and that of Ansley (1978, 1979) become computationally completely infeasible when the seasonal period s becomes much larger than 12 as in the case of half-monthly ($s = 24$), weekly ($s = 52$) or daily ($s = 365$) time series. Such models have been found useful in forecasting hydrological variables (McMichael and Hunter, 1972; McLeod, Hipel and Sales, 1982) and there are no doubt many other possible applications. *SARMAS* is usually more efficient for the regular non-seasonal *ARMA* model as well. Finally, another advantage of *SARMAS* is that residuals which estimate the actual innovation series are produced. These residuals are useful not only for model diagnostic checking (Box and Jenkins, 1976, Ch. 8) but also in the elegant and computationally efficient forecasting methods given in Box and Jenkins (1976, Chapter 5).

The subroutine *DTARMA* is used by *SARMAS* to calculate the approximation to the determinant term in the *ARMA* model likelihood given in McLeod (1977). This subroutine is also useful in checking for model stationarity and invertibility. Thus *DTARMA* could be used in conjunction with AS 154 to ensure the parameter values are inside the admissible region during the numerical maximization of the likelihood.

The subroutine *MCHOL*, used by *DTARMA*, determines the modified Cholesky decomposition of a positive-definite matrix A, given by

$$A = L D L' \tag{1}$$

where L is a lower triangular matrix with ones on the diagonal and D is a diagonal matrix. This form of the decomposition, which avoids the square-root computation in the standard decomposition (Healy, 1968), is slightly more accurate and efficient if only the determinant of A is required. *MCHOL* is also more convenient in other applications (Martin, Peters and Wilkinson, 1965; Pagano, 1972).

Present address: Department of Statistical and Actuarial Sciences, The University of Western Ontario London, Ontario, Canada N6A 5B9

 0035–9254/83/32211 $2.00

THEORY

The *SARMA* $(p, q)(p_s, q_s)_s$ model is defined by

$$\Phi(B^s)\,\phi(B)\,z_t = \Theta(B^s)\,\theta(B)\,a_t, \tag{2}$$

where

$$\phi(B) = 1 - \phi_1 B - \ldots - \phi_p B^p, \quad \theta(B) = 1 - \theta_1 B - \ldots - \theta_q B^q,$$

$$\Phi(B^s) = 1 - \Phi_1 B^s - \ldots - \Phi_{p_s} B^{sp_s}, \quad \Theta(B^s) = 1 - \Theta_1 B^s - \ldots - \Theta_{q_s} B^{sq_s},$$

B is the backshift operator, s the seasonal period and a_t a sequence of independent normal variables with mean 0 and variance σ^2. The a_t's, called the innovations, represent the one-step forecast errors when the model parameters, $\beta = (\phi_1, \ldots, \phi_p, \theta_1, \ldots, \theta_q, \Phi_1, \ldots, \Phi_{p_s} \Theta_1, \ldots, \Theta_{q_s})$, are known. Note that the *ARMA* (p, q) model is obtained by taking $p_s = q_s = 0$. The *SARMA* model is said to be stationary and invertible, respectively, if all roots of $\Phi(B)\,\phi(B) = 0$ and $\Theta(B)\,\theta(B) = 0$ are outside the unit circle. Although the *SARMA* model may be considered as a special case of the *ARMA* (p^*, q^*) model by taking $p^* = p + sp_s$, $q^* = q + sq_s$, $\phi^*(B) = \Phi(B^s)\,\phi(B)$ and $\theta^*(B) = \Theta(B^s)\,\theta(B)$, it will be shown how a more efficient estimation algorithm can be developed utilizing the multiplicative structure of the *SARMA* model.

Given observations $z_t (t = 1, \ldots, n)$ the exact log-likelihood function maximized over σ^2 may be written, apart from an arbitrary constant, as

$$\log L(\beta) = -n \log (S_m/n)/2, \tag{3}$$

where S_m, the modified sum of squares, is

$$S_m = S[M_n(p, q, p_s, q_s, s)]^{-1/n}. \tag{4}$$

S represents the unconditional sum of squares of Box and Jenkins (1976) defined by

$$S = \sum_{t=-\infty}^{n} [a_t]^2, \tag{5}$$

where $[.]$ denotes expectation given $z_1, \ldots, z_n$.

The evaluation of S by the iterative unconditional sum of squares method may involve two types of truncation error. First, the infinite sum in (5) is replaced by

$$S = \sum_{t=1-Q}^{n} [a_t]^2 \tag{6}$$

for suitably large Q. Theoretically, Q should be chosen so that

$$\gamma_0/\sigma^2 - \sum_{i=0}^{Q} \psi_i^2 < e_{tol}, \tag{7}$$

where $\gamma_0 = \text{var}(z_t)$, ψ_i is the coefficient of a_{t-i} in the infinite moving average representation of (2) and e_{tol} is an error tolerance. Thus if the model contains an autoregressive factor with roots near the unit circle, a fairly large Q might be necessary. In practice,

$$Q = q + sq_s + 20\,(p + sp_s) \tag{8}$$

is often sufficient. The other truncation error involves terminating the iterative procedure used to calculate $[a_t]$. Several iterations may be required to obtain convergence when the model contains a moving average factor with roots near the unit circle. However, sufficient accuracy is usually obtained on the first evaluation.

McLeod (1977, 1982) suggested that the term $M_n(p,q,p_s,q_s,s)$ be replaced by $m(p,q,p_s,q_s,s)$, given by

$$m(p,q,p_s,q_s,s) = M(p,q)\,[M(p_s,q_s)]^s, \tag{9}$$

where $M(p,q)$ is defined for any *ARMA* (p,q) model as

$$M(p,q) = M_p^2 M_q^2 / M_{p+q} \tag{10}$$

where the terms M_p, M_q and M_{p+q} are defined in terms of the auxiliary autoregressions, $\phi(B)v_t = a_t$ and $\theta(B)u_t = a_t$ and the left-adjoint autoregression $\phi(B)\,\theta(B)y_t = a_t$. For the autoregression, $\phi(B)\,v_t = a_t, M_p$ is the determinant of the $p \times p$ matrix with (i,j) entry

$$\sum_{k=1}^{\min(i,j)} \phi_{i-k}\,\phi_{j-k} - \phi_{p+k-i}\,\phi_{p+k-j} \tag{11}$$

and similarly for the other autoregressions. The $p \times p$ matrix defined by (11) is called the Schur matrix of $\phi(B)$. Pagano (1973) has shown that a necessary and sufficient condition for stationarity of an autoregression is that its Schur matrix be positive-definite. Thus calculation of $m(p,q,p_s,q_s,s)$ also provides a check on the stationarity and invertibility conditions and so during estimation the parameters may be constrained to the admissible region. The subroutine *DTARMA* evaluates $M(p,q)$ using the modified Cholesky decomposition subroutine *MCHOL*. The method of Martin, Peters and Wilkinson (1965, equations (6) to (10)) is implemented in *MCHOL*.

METHOD

This section describes how the backforecasting method of Box and Jenkins (1976, Ch. 7) for *ARMA* models can be efficiently adapted to *SARMA* models by making use of their multiplicative structure.

After taking conditional expectations in (2) the backward equation is

$$\Phi(B^s)\,\phi(B)\,[z_t] = \Theta(B^s)\,\theta(B)\,[a_t], \tag{12}$$

where $[a_t] = 0, t > n$. This may also be written

$$\phi(B)\,[z_t] = \theta(B)\,[x_t] \tag{13}$$

and

$$\Phi(B^s)\,[x_t] = \Theta(B^s)\,[a_t]. \tag{14}$$

The forward form of the model is

$$\Phi(F^s)\,\phi(F)\,z_t = \Theta(F^s)\,\theta(F)\,e_t, \tag{15}$$

where $F = B^{-1}$ and e_t is a sequence of independent normal random variables with mean 0 and variance σ^2. Thus the forward equations may be written,

$$\phi(F)\,[z_t] = \theta(F)\,[y_t] \tag{16}$$

and

$$\Phi(F^s)\,[y_t] = \Theta(F^s)\,[e_t], \tag{17}$$

where $[e_t] = 0, t < 1$.

The iterative unconditional sum of squares calculation proceeds through the following steps.

Step 0: Initialization. Set S' to -1. Select Q and choose the error tolerance, E_{tol}, for the convergence test in Step 7.

Step 1: Calculate $[y_t]$ $(t = n + Q, \ldots, 1)$ using (16). On the 0th iteration set $[y_t] = 0, t \geqslant n-p$.

Step 2: Calculate $[e_t]$ $(t = n + Q, \ldots, 1)$ using (17). On the 0th iteration, set $[e_t] = 0$, $t \geqslant n - p - sp_s$.

Step 3: Backforecast y_t $(t = 0, -1, \ldots, 1 - Q)$ using (17).

Step 4: Backforecast z_t $(t = 0, -1, \ldots, 1 - Q)$ using (16).

Step 5: Calculate $[x_t]$ $(t = 1 - Q, \ldots, n)$ using (13).

Step 6: Calculate $[a_t]$ $(t = 1 - Q, \ldots, n)$ using (14).

Step 7: Test for convergence. Calculate S. If $|S - S'|/S < E_{tol}$, terminate. Otherwise set $S' = S$ and proceed to Step 8.

Step 8: Forecast x_t $(t = n + 1, \ldots, n + Q)$ using (14).

Step 9: Forecast z_t $(t = n + 1, \ldots, n + Q)$. Return to Step 1.

S may also be calculated after the e_t's are calculated in Step 2 and tested with the previous value obtained in Step 7. However, the method given instead is preferred since the a_t's are usually required.

The unconditional method without iteration terminates after Step 6 while the conditional method uses only Steps 5 and 6.

STRUCTURE

SUBROUTINE SARMAS(Z, NZ, N, BETA, NBETA, IP, IQ, IPS, IQS, ISEA, IQAP, MAXIT, A, S, SM, W, NW, IFAULT)

Formal parameters

Z	Real array (*NZ*)	input:	$Z(1) \ldots Z(N)$ should contain the time series in reverse chronological order, $z_n, z_{n-1}, \ldots, z_1$
		output:	locations $n+1, \ldots, n+Q$ contain the backforecast values of $z_0, z_{-1}, \ldots, z_{1-Q}$ respectively while the first n locations are not changed
NZ	Integer	input:	$n + Q$
N	Integer	input:	n, the number of observations
BETA	Real array (*NBETA*)	input:	$\phi_1, \ldots, \phi_p$, $\theta_1, \ldots, \theta_q$, $\Phi_1, \ldots \Phi_{p_s}$, $\Theta_1, \ldots, \Theta_{q_s}$
NBETA	Integer	input:	$\max(1, p + q + p_s + q_s)$
IP	Integer	input:	p
IQ	Integer	input:	q
IPS	Integer	input:	p_s
IQS	Integer	input:	q_s
ISEA	Integer	input:	s, the seasonal period
IQAP	Integer	input:	Q, maximum number of backforecasts used. If $Q > 0$, the unconditional sum of squares is calculated. Otherwise, if $Q = 0$, the conditional sum of squares is calculated
MAXIT	Integer	input:	maximum number of iterations in the unconditional sum of squares calculation. If *IQAP* = 0 or *IQ* = *IQS* = 0, the *MAXIT* parameter is ignored and the algorithm terminates after Step 6 (see METHOD)
A	Real array (*NZ*)	output:	contains the residuals, $[a_n], [a_{n-1}], \ldots, [a_1], [a_0], \ldots, [a_{1-Q}]$
S	Real	output:	the unconditional or conditional sum of squares (depending on *IQAP*). Note that, S/N is an estimate of the residual variance

SM	Real	output:	the modified sum of squares, S_m
W	Real array (*NW*)	workspace:	
NW	Integer	input:	$\max(n+Q,\ (p+q)(p+q+1)/2,\ (p_s+q_s)(p_s+q_s+1)/2)$
IFAULT	Integer	output:	a fault indicator, equal to 1 if convergence not obtained in the iterative unconditional sum of squares calculation 2 if the model is non-stationary 3 if the model is non-invertible 4 if the setting of *NZ*, *NBETA* or *NW* is invalid 5 if $n \leqslant \max(p + sp_s, q + sq_s)$ 6 if one of *IP, IQ, IPS, IQS, ISEA, IQAP* or *MAXIT* is negative 7 if $Q < \max(p + sp_s, q + sq_s)$ when *MAXIT* and *IQAP* are positive 0 otherwise

AUXILIARY ALGORITHMS

Two auxiliary routines are included as indicated in the THEORY section: *DTARMA* to calculate $M(p, q)$ of equation (10), and *MCHOL* to perform the modified Cholesky decomposition.

SUBROUTINE DTARMA (BETA, NBETA, IP, IQ, WS, NWS, DETM, IFAULT)

Formal parameters

BETA	Real array (*NBETA*)	input:	$\phi_1, \ldots, \phi_p, \theta_1, \ldots, \theta_q$
NBETA	Integer	input:	$\max(1, p+q)$
IP	Integer	input:	p
IQ	Integer	input:	q
WS	Real array (*NWS*)	workspace:	
NWS	Integer	input:	$1 + p + q + (p+q)(p+q+1)/2$
DETM	Real	output:	$M(p, q)$
IFAULT	Integer	output:	a fault indicator, equal to 1 if the model is non-stationary 2 if the model is non-invertible 3 if the setting of *NBETA* or *NWS* is invalid 0 otherwise

SUBROUTINE MCHOL (A, NA, N, DET, IFAULT)

Formal parameters

A	Real array (*NA*)	input:	the positive definite input matrix, stored in symmetric-storage mode a_{11}, a_{21}, $a_{22}, \ldots a_{mm}$
		output:	the modified Cholesky decomposition stored as a one-dimensional array in the sequence d_1, l_{21}, d_{22}, l_{31}, l_{32}, $d_{33}, \ldots, l_{m,m-1}, d_{mm}$
NA	Integer	input:	$m(m+1)/2$
N	Integer	input:	m, the order of the input matrix
DET	Real	output:	the determinant of *A*
IFAULT	Integer	output:	a fault indicator, equal to 1 if the setting of *NA* or *N* is invalid 2 if the input matrix is not positive-definite 0 otherwise

Underflow

A floating point underflow may occur during the backforecasting step. The result should be set to zero. This is usually done automatically but sometimes it may be necessary to call a system subroutine to do this.

PRECISION

For machines using fewer than 60 bits for real variables, the use of double precision is recommended. This may be implemented as follows.

(i) Declare all real variables in *SARMAS*, *DTARMA* and *MCHOL* to be double precision.

(ii) Change all the real constants in the data statements in *SARMAS*, *DTARMA* and *MCHOL* to their double precision value. The variable *ETA* in *MCHOL* should also be changed as indicated in the comment statement which precedes it.

(iii) Change *FLOAT* to *DFLOAT* in *SARMAS* and then insert the statement

$$DFLOAT(N) = DBLE(FLOAT(N))$$

immediately before the first executable statement in the subroutine. Declare *FLOAT* to be real.

(iv) Change *ABS* to *DABS* in *SARMAS* and *MCHOL*.

TIME AND ACCURACY

The amount of computer time depends on the length of the series and the type of *ARMA* model. If the iterative method is used, short series may require a number of iterations to reach convergence when the parameters are close to the admissible boundary and in some cases Ansley's subroutine *ARMA* (Ansley, 1978) or AS 154 may be faster. However, what is more important is the time required to obtain estimates. Also, for short series slight differences are not crucial. Illustrative times for one function evaluation are shown in Table 1 for *SARMAS*, AS 154 and *ARMA* (Ansley, 1978).

TABLE 1

CPU time required in milliseconds on the CYBER 170/835 for one function evaluation with n = 50 (first entry) and n = 200 (second entry)

Model	*Parameter Setting* †	*SARMAS MAXIT* = 0	*SARMAS MAXIT* = 20	*AS*154	*ARMA*
(0, 1)	0.5	3,11	6,22	7,29	8,27
	0.9	3,12	6,21	7,27	7,25
(1, 1)	0.5	7,16	16,35	8,29	10,31
	0.9	7,17	14,32	10,29	9,31
$(0, 1)\ (0, 1)_4$	0.5	5,15	10,35	17,82	18,70
	0.9	4,15	35,34	18,72	20,71
$(0, 1)\ (0, 1)_{12}$	0.5	7,15	17,36	70,255	29,115
	0.9	5,16	86,73	74,260	84,118
$(0, 1)\ (0, 1)_{52}$	0.5	–,19	–,68	–	–
	0.9	–,20	–,248	–	–

† All parameters were set to either 0.5 or 0.9

The accuracy of the algorithm *SARMAS* is best judged in terms of the accuracy of the estimates it may provide. Simulation work of Ansley and Newbold (1980) suggests that exact maximum likelihood estimators are preferable to unconditional or conditional sum of squares estimators. Although further work is needed experience to date suggests there is little difference between the exact and the proposed approximate likelihood estimator based on the unconditional sum of squares without iteration. The amount of computer time needed to obtain estimates using this approximate likelihood technique is generally much less than that required by any of the exact likelihood methods particularly with "long seasonal" series. For example, when a $(0, 1)\ (0, 1)_{12}$

model was fitted to the log differenced-seasonal differenced Airline Data (Series G, Box and Jenkins, 1976) using *SARMAS* and the subroutines of Ansley (1978) and of Gardner, Harvey and Phillips (1980) it was found that, at least to within an error tolerance of four significant digits, all methods converged to exactly the same estimates of θ_1 and Θ_1. But the central processor time required was respectively 0.98, 15.5 and 27.4 seconds on a CYBER-835(NOS) Computer. Double precision arithmetic and the conjugate direction algorithm of Powell (1964) was used in each optimization. The numerical values of the estimates were previously given by McLeod (1977, Table1).

ADDITIONAL COMMENTS

The function minimization algorithm of Powell (1964) has proved very effective in obtaining maximum likelihood estimates by minimizing the modified sum of squares calculated by *SARMAS*. By searching down conjugate directions this algorithm obtains the minimum of a quadratic function in a finite number of iterations and so is said to be quadratically convergent. A Fortran subroutine coded by M. J. D. Powell, which implements the technique of Powell (1964) is given in Kuester and Mize (1973). When using this unconstrained minimization algorithm, it is convenient to standardize the time series so that the total sum of squares when $\beta = 0$ is n. Then S_m is set to n when β is found to be inadmissible. This simple penalty function does not degrade the performance of the algorithm. Furthermore, this standardization is also useful if a maximum likelihood estimate of the mean of the time series is also desired.

ACKNOWLEDGEMENT

This research was supported by the National Sciences and Engineering Research Council of Canada and by Eletrobrás, Brazil. The authors are grateful to Dr G. Tunnicliffe Wilson for helpful discussions.

REFERENCES

Ansley, C. F. (1978) Subroutine *ARMA*: Exact likelihood for univariate *ARMA* processes. Technical Report, Graduate School of Business, University of Chicago.

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

——— and Newbold, P. (1980) Finite sample properties of estimators for autoregressive moving average models *J. Econometrics*, **13**, 159–183.

Box, G. E. P. and Jenkins, G. M. (1976) *Time Series Analysis*: *Forecasting and Control* (*2nd ed.*). San Francisco: Holden-Day.

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.

Healy, M. J. R. (1968) Algorithm AS 6. Triangular decomposition of a symmetric matrix. *Appl. Statist.*, **17**, 19–21.

Kuester, J. L. and Mize, J. H. (1973) *Optimization Techniques with Fortran*. New York: McGraw-Hill.

Martin, R. S., Peters, G. and Wilkinson, J. H. (1965) Symmetric decomposition of a positive definite matrix. *Numerische Mathematik*, **7**, 362–383.

McLeod, A. I. (1977) Improved Box–Jenkins estimators. *Biometrika*, **64**, 531–534.

——— (1982) Duality and other properties of multiplicative seasonal autoregressive-moving average models. Submitted to *Biometrika*.

——— Hipel, K. W. and Holanda Sales, P. R. (1982) Modelling and forecasting weekly riverflow time series. In preparation.

McMichael, F. C. and Hunter, J. S. (1972) Stochastic modeling of temperature and flow in rivers. *Water Resources Res.*, **8**, 87–98.

Pagano, M. (1972) An algorithm for fitting autoregressive schemes. *Appl. Statist.*, **21**, 174–281.

——— (1973) When is an autoregressive process stationary? *Commun. Statist.*, **1**, 533–544.

Powell, M. J. D. (1964) An efficient method for finding the minimum of a function of several variables without calculating derivatives. *Computer J.*, **7**, 155–162.

```
      SUBROUTINE SARMAS(Z, NZ, N, BETA, NBETA, IP, IQ, IPS, IQS, ISEA,
     *  IQAP, MAXIT, A, S, SM, W, NW, IFAULT)
C
C        ALGORITHM AS 191  APPL. STATIST. (1983) VOL.32, NO.2
C
      DIMENSION Z(NZ), A(NZ), W(NW), BETA(NBETA)
C
      LOGICAL SWITCH
C
C        INITIALIZE NUMERICAL CONSTANTS
C
      DATA ZERO /0.0E0/, ONE /1.0E0/, ONENEG /-1.0E0/
C
C        ETOL - ERROR TOLERANCE IN CONVERGENCE CRITERION
C
      DATA ETOL /1.0E-8/
C
      ITER = 0
      SWITCH = .FALSE.
      SPREV = ZERO
      IPQ = IP + IQ
      IPQPS = IPQ + IPS
      IPSTS = IPS * ISEA
      IPSTS1 = IPSTS + 1
      IQSTS = IQS * ISEA
      IQSTS1 = IQSTS + 1
      IPSQS = IPS + IQS
      IQAP2 = IQAP
      IF (IP .EQ. 0 .AND. IPS .EQ. 0) IQAP2 = MIN0(IQAP, IQ + IQSTS)
      MAXIT2 = MAXIT
      IF (IQ .EQ. 0 .AND. IQS .EQ. 0) MAXIT2 = 0
      IF (MAXIT2 .EQ. 0) SWITCH = .TRUE.
      NBY2 = N / 2
C
C        INPUT VALIDATION
C
      IFAULT = 0
      IR = MAX0(IQ + IQSTS, IP + IPSTS)
      IF (IR .GE. N) IFAULT = 5
      IF (MAXIT .GT. 0 .AND. IR .GT. IQAP) IFAULT = 7
      IF (IPQ + IPSQS .GT. NBETA) IFAULT = 4
      IF (N + IQAP2 .GT. NZ) IFAULT = 4
      IF (NW .LT. MAX0(NZ, 1 + IPQ + IPQ * (IPQ + 1) / 2, 1 + IPSQS +
     *  IPSQS * (IPSQS + 1) / 2)) IFAULT = 4
      IF (MIN0(IP, IQ, IPS, IQS, ISEA, IQAP, MAXIT) .LT. 0) IFAULT = 6
      IF (IFAULT .GE. 1) RETURN
C
C        OBTAIN NECESSARY DETERMINANTS
C        CHECK FOR STATIONARITY/INVERTIBLITY
C
      DETM = ONE
      DETMS = ONE
      IER = 0
      IF (IPQ .NE. 0) CALL DTARMA(BETA, IPQ, IP, IQ, W, NW, DETM, IER)
      IF (IER .GT. 0) GOTO 340
      IF (IPSQS .EQ. 0) GOTO 20
      II = IPQ
      DO 10 I = 1, IPSQS
      II = II + 1
      A(I) = BETA(II)
   10 CONTINUE
      CALL DTARMA(A, IPSQS, IPS, IQS, W, NW, DETMS, IER)
      IF (IER .GT. 0) GOTO 340
C
C        IF IQAP2 IS 0, USE CONDITIONAL SUM OF SQUARES METHOD
C
   20 IF (IQAP2 .EQ. 0) GOTO 200
C
C        IF NO SEASONAL COMPONENT AND NO MOVING-AVERAGE COMPONENT,
C        PROCEED DIRECTLY TO BACKFORECASTING STEP
C        (Y AND E-SERIES NOT NEEDED)
```

```
      IF (IPSQS .EQ. 0 .AND. IQ .EQ. 0) GOTO 110
C
C        CALCULATE Y-SERIES, USE W-VECTOR
C
      DO 60 I = 1, N
      W(I) = ZERO
      IF (I .LE. IP) GOTO 60
      W(I) = Z(I)
      IF (IP .EQ. 0) GOTO 40
      DO 30 J = 1, IP
      III = I - J
      W(I) = W(I) - BETA(J) * Z(III)
   30 CONTINUE
   40 L = MIN0(IQ, I - 1)
      IF (L .EQ. 0) GOTO 60
      DO 50 J = 1, L
      JJ = IP + J
      III = I - J
      W(I) = W(I) + BETA(JJ) * W(III)
   50 CONTINUE
   60 CONTINUE
C
C        CALCULATE E-SERIES, USE A-VECTOR
C
      LQS = IQS
      DO 100 I = 1, N
      A(I) = ZERO
      IF (I .LE. IPSTS) GOTO 100
      A(I) = W(I)
      IF (IPS .EQ. 0) GOTO 80
      III = I
      JJ = IPQ
      DO 70 J = 1, IPS
      III = III - ISEA
      JJ = JJ + 1
      A(I) = A(I) - BETA(JJ) * W(III)
   70 CONTINUE
   80 IF (IQS .EQ. 0) GOTO 100
      IF (I .LE. IQSTS1) LQS = (I - 1) / ISEA
      IF (LQS .EQ. 0) GOTO 100
      III = I
      DO 90 J = 1, LQS
      III = III - ISEA
      JJ = IPQPS + J
      A(I) = A(I) + BETA(JJ) * A(III)
   90 CONTINUE
  100 CONTINUE
C
C        BACKFORECAST Y-SERIES, USE W(N+1), W(N+2), ...
C
  110 DO 150 I = 1, IQAP2
      NPI = N + I
      W(NPI) = ZERO
      A(NPI) = ZERO
      IF (I .GT. IQSTS) GOTO 130
      III = NPI
      DO 120 J = 1, IQS
      III = III - ISEA
      JJ = IPQPS + J
      W(NPI) = W(NPI) - BETA(JJ) * A(III)
  120 CONTINUE
  130 IF (IPS .EQ. 0) GOTO 150
      III = NPI
      DO 140 J = 1, IPS
      III = III - ISEA
      JJ = IPQ + J
      W(NPI) = W(NPI) + BETA(JJ) * W(III)
  140 CONTINUE
  150 CONTINUE
C
C        BACKFORECAST Z-SERIES, USE Z(N+1), Z(N+2), ...
```

```
      DO 190 I = 1, IQAP2
      NPI = N + I
      Z(NPI) = W(NPI)
      IF (IQ .EQ. 0) GOTO 170
      DO 160 J = 1, IQ
      NPIMJ = NPI - J
      JJ = IP + J
      Z(NPI) = Z(NPI) - BETA(JJ) * W(NPIMJ)
  160 CONTINUE
  170 IF (IP .EQ. 0) GOTO 190
      DO 180 J = 1, IP
      NPIJ = NPI - J
      Z(NPI) = Z(NPI) + BETA(J) * Z(NPIJ)
  180 CONTINUE
  190 CONTINUE
C
C        CALCULATE X-SERIES, USE W-VECTOR
C
  200 NPQAP = N + IQAP2
      II = NPQAP + 1
      DO 240 I = 1, NPQAP
      II = II - 1
      W(II) = Z(II)
      IM1 = I - 1
      L = MIN0(IM1, IP)
      IF (L .EQ. 0) GOTO 220
      III = II
      DO 210 J = 1, L
      III = III + 1
      W(II) = W(II) - BETA(J) * Z(III)
  210 CONTINUE
  220 L = MIN0(IM1, IQ)
      IF (L .EQ. 0) GOTO 240
      III = II
      DO 230 J = 1, L
      III = III + 1
      JJ = IP + J
      W(II) = W(II) + BETA(JJ) * W(III)
  230 CONTINUE
  240 CONTINUE
C
C        CALCULATE A-SERIES, USE A-VECTOR
C
      II = NPQAP + 1
      DO 280 I = 1, NPQAP
      II = II - 1
      A(II) = W(II)
      IF (ISEA .EQ. 0) GOTO 280
      IF (I .LE. IPSTS1) LPS = (I - 1) / ISEA
      IF (LPS .EQ. 0) GOTO 260
      III = II
      DO 250 J = 1, LPS
      III = III + ISEA
      JJ = IPQ + J
      A(II) = A(II) - BETA(JJ) * W(III)
  250 CONTINUE
  260 IF (I .LE. IQSTS1) LQS = (I - 1) / ISEA
      IF (LQS .EQ. 0) GOTO 280
      III = II
      DO 270 J = 1, LQS
      III = III + ISEA
      JJ = IPQPS + J
      A(II) = A(II) + BETA(JJ) * A(III)
  270 CONTINUE
  280 CONTINUE
C
C        CALCULATE THE SUM OF SQUARES
C
      S = ZERO
      DO 300 I = 1, NPQAP
  300 S = S + A(I) * A(I)
```

```
C         TEST FOR CONVERGENCE
C
      IF (IQAP2 .EQ. 0) GOTO 330
      IF (SWITCH) GOTO 310
      IFAULT = 0
      RELERR = (S - SPREV) / S
      IF (ABS(RELERR) .LE. ETOL) GOTO 330
C
C         CONVERGENCE NOT OBTAINED.
C
  310 IFAULT = 1
      IF (ITER .GE. MAXIT2) GOTO 330
C
C         REVERSE THE SERIES AND PROCEED TO THE FORECASTING STEP.
C
      SPREV = S
      II = N
      DO 320 I = 1, NBY2
      TEMP = W(II)
      W(II) = W(I)
      W(I) = TEMP
      TEMP = A(II)
      A(II) = A(I)
      A(I) = TEMP
      TEMP = Z(II)
      Z(II) = Z(I)
      Z(I) = TEMP
      II = II - 1
  320 CONTINUE
      IF (SWITCH) ITER = ITER + 1
      SWITCH = .NOT.SWITCH
      GOTO 110
C
C         MODIFIED SUM OF SQUARES
C
  330 TEMP = ONENEG / FLOAT(N)
      SM = S * DETM ** TEMP * DETMS ** (FLOAT(ISEA) * TEMP)
      IF (MAXIT2 .EQ. 0) IFAULT = 0
      RETURN
C
C         MODEL IS NONSTATIONARY OR NONINVERTIBLE
C
  340 IFAULT = IER + 1
      RETURN
      END
C
      SUBROUTINE DTARMA(BETA, NBETA, IP, IQ, WS, NWS, DETM, IFAULT)
C
C         ALGORITHM AS 191.1  APPL. STATIST. (1983) VOL.32, NO.2
C
      DIMENSION BETA(NBETA), WS(NWS)
      DATA ZERO, ONE, ONENEG /0.0E0, 1.0E0, -1.0E0/
      IFAULT = 0
      IF (NBETA .LT. IP + IQ) GOTO 140
      NWCHEK = 1 + NBETA + NBETA * (NBETA + 1) / 2
      IF (NWCHEK .GT. NWS) GOTO 140
      DET = ONE
      DET1 = ONE
      IR = IP + IQ
      IRS = NWS - IR - 1
      IRSP1 = IRS + 1
      WS(IRSP1) = ONENEG
      ISW = 0
      IF (IP .EQ. 0) ISW = 1
      ILOOP = IP
   10 IF (ISW .EQ. 1) ILOOP = IQ
      IF (ILOOP .EQ. 0) GOTO 120
      IF (ISW .EQ. 2) GOTO 30
      DO 20 I = 1, ILOOP
      IRSPI = IRS + I + 1
      IPPI = ISW * IP + I
      WS(IRSPI) = BETA(IPPI)
```

```
   20 CONTINUE
      GOTO 60
   30 IF (IP .EQ. 0) GOTO 120
      ILOOP = IR
C
C        MULTIPLY THE AUTOREGRESSIVE AND MOVING AVERAGE OPERATORS TO
C        OBTAIN COEFFICIENTS IN THE LEFT-ADJOINT AR(IP+IQ) MODEL
C
      DO 50 I = 1, IR
      II = IRS + 1 + I
      WS(II) = ZERO
      IMIQ = I - IQ
      J1 = MAX0(0, IMIQ) + 1
      J2 = MIN0(I, IP) + 1
      DO 40 J = J1, J2
      JM1 = J - 1
      IF (J .EQ. 1) BJ = ONENEG
      IF (J .NE. 1) BJ = BETA(JM1)
      IMJ = I - J + 1
      IPPIMJ = IP + IMJ
      IF (IMJ .EQ. 0) BI = ONENEG
      IF (IMJ .NE. 0) BI = BETA(IPPIMJ)
      WS(II) = WS(II) - BI * BJ
   40 CONTINUE
   50 CONTINUE
C
C        FORM THE SCHUR MATRIX
C
   60 M = 0
      IEND = ILOOP + 1
      DO 90 I = 1, ILOOP
      DO 80 J = 1, I
      M = M + 1
      WS(M) = ZERO
      L = MIN0(I, J)
      DO 70 K = 1, L
      IRSI = IRS + I - K + 1
      IRSJ = IRS + J - K + 1
      IRSPI = IRS + IEND - I + K
      IRSPJ = IRS + IEND - J + K
      WS(M) = WS(M) + WS(IRSI) * WS(IRSJ)
      WS(M) = WS(M) - WS(IRSPI) * WS(IRSPJ)
   70 CONTINUE
   80 CONTINUE
   90 CONTINUE
C
C        CALCULATE THE DETERMINANT USING THE MODIFIED CHOLESKY DECOMP
C
      CALL MCHOL(WS, NWS, ILOOP, DET, IFAULT)
      IF (IFAULT .GT. 0) GOTO 130
C
      IF (ISW .GE. 1) GOTO 110
      ISW = 1
      DET1 = DET * DET
      GOTO 10
  110 IF (ISW .EQ. 2) GOTO 120
      ISW = 2
      DET1 = DET1 * DET * DET
      GOTO 10
  120 DETM = DET1 / DET
      RETURN
  130 IFAULT = ISW + 1
      RETURN
  140 IFAULT = 3
      RETURN
      END
C
      SUBROUTINE MCHOL(A, NA, N, DET, IFAULT)
C
C        ALGORITHM AS 191.2  APPL. STATIST. (1983) VOL.32, NO.2
C
      DIMENSION A(NA)
```

```
      DATA ONE /1.0E0/
C
C        ETA - LARGEST NUMBER SUCH THAT 1.0+ETA=1.0
C        (DEPENDS ON MACHINE PRECISION)
C
      DATA ETA /1.0E-15/
C
      IFAULT = 1
      DET = ONE
      IF (N .LE. 0) GOTO 70
      IF (NA .LT. N * (N + 1) / 2) GOTO 70
      IFAULT = 2
      J = 1
      K = 0
      DO 60 IROW = 1, N
      L = 0
      DO 20 ICOL = 1, IROW
      K = K + 1
      W = A(K)
      M = J
      IF (IROW .EQ. ICOL) GOTO 30
      DO 10 I = 1, ICOL
      L = L + 1
      IF (I .EQ. ICOL) GOTO 20
      W = W - A(L) * A(M)
      M = M + 1
   10 CONTINUE
   20 A(K) = W
   30 II = 0
      DO 40 I = 1, ICOL
      IF (I .EQ. ICOL) GOTO 50
      II = II + I
      T = A(M)
      TT = A(M) / A(II)
      W = W - T * TT
      A(M) = TT
      M = M + 1
   40 CONTINUE
   50 IF (W .LT. ETA * ABS(A(K))) GOTO 70
      A(K) = W
      DET = DET * W
      J = J + IROW
   60 CONTINUE
      IFAULT = 0
   70 RETURN
      END
```

Remark AS R47

A Remark on AS 177. Expected Normal Order Statistics (Exact and Approximate)

By W. KÖNIGER

Dorsch Consult Ingenieurgesellschaft mbH, 8 *München* 21, *Germany*

[Received November 1982]

In *SUBROUTINE NSCOR*1 the function *ALNFAC* is accessed unnecessarily prior to executing the *DO* 20 loop. This follows by noting in equation (1) that

$$\frac{n!}{(r-1)!\,(n-r)!} = r\binom{n}{r} = n\binom{n-1}{r-1}.$$

For increased efficiency the following changes should be made to *NSCOR*1:

 0035–9254/83/32223 $2.00