Abstract

Ansley (1978a, 1979) derived an algorithm to compute the likelihood function of data from an ARMA time series model. The algorithm reported here follows Ansley's basic idea but has some different features. While it cannot accommodate seasonal models, it estimates the mean, gives forecasts and their covariance matrix, and avoids computing square roots. A second algorithm is described which computes the necessary information for a structured Bayesian analysis.

A. Introduction

Ansley (1978a, 1979) derived an algorithm to compute the likelihood function of data from an ARMA time series model. The algorithm reported here follows Ansley's basic idea but has some different features. While it cannot accommodate seasonal models, it estimates the mean, gives forecasts and their covariance matrix, and avoids computing square roots. A second algorithm is described which computes the necessary information for a structured Bayesian analysis.

The great value of Ansley's algorithm and of those given here is a great reduction in the computational effort for analyzing ARMA time series models. For a series of length N, with the order of the model (p,q) such that $N >> \max(p,q)$, as is often the case, any straightforward method for evaluating the likelihood function exactly requires $O(N^3)$ operations. A special algorithm due to Trench (1964) reduces this by an order of magnitude to $O(N^2)$. But Trench's algorithm is sufficiently complicated and the cost $O(N^2)$ is still so high that only approximate or iterative methods have been used and the exact likelihood had been viewed only as an ideal. Ansley's algorithm reduces the computational effort another order of magnitude, to $O(N)$, making it competitive in time with approximations, making those approximations valueless.

For the research interests of this author, it is necessary to revise entirely Ansley's method to fit the computation needs of a structured Bayesian approach. The basis of this approach is described in detail in Monahan (1980). The complete computational requirements for the Bayesian approach are not met here; further work has been done and its exposition is in preparation.

The definitions and notation for this paper is given in Section B. The computational needs for the maximum likelihood and Bayesian approaches are given in Sections C and D, respectively. The main part of this paper is Section E, the description of algorithm ARMAML. The modifications for the Bayesian approach follow in Section F. Some details from Section E are found in the description of GETSET in Section G. Listings of the programs are given in the Appendix.

All the programs are written as Fortran subprograms, subroutines and a function, and are implemented in double precision. Changing to single precision requires changing all of the REAL*8 declarations to REAL and DABS to ABS (twice) in GESEPP. Single precision arithmetic on IBM machines is certainly inappropriate. However, the precision of single precision arithmetic of CDC machines (48 bit mantissa) may suffice. Any problems of portability, i.e., running these programs on machines other than the IBM 370 using Fortran G and H, should be minor.

The programs were checked in four ways. First, the covariance function coming out of CFARMA was compared to those obtainable from formulas. Second, both ARMAML and BARMAN were compared to a straightforward "long way" method. Then ARMAML was compared with Ansley's (1978b) test problems (although disagreeing on the known mean calculations). Finally, ARMAML and BARMAN were compared with special programs previously written for maximum likelihood and Bayesian analyses by the author over the last few years.

B. Notation and Definitions

Consider the autoregressive-moving average (ARMA) process $\{z_t\}$ of order (p,q) described by (B.1):

$$(B.1) \qquad (z_t-\mu) - \phi_1(z_{t-1}-\mu) - \dots - \phi_p(z_{t-p}-\mu) = e_t - \theta_1 e_{t-1} - \dots - \theta_q e_{t-q}$$

where the e_t's are iid normal random variables each with mean zero and variance σ_e^2. A finite segment of this process is observed: $z = (z_1, \dots, z_N)^T$ which has an N dimensional multivariate normal distribution with mean vector $\mu 1_N$ and covariance matrix $\sigma_e^2 A_N$. The matrix A_N is described by

$$(B.2) \qquad (A_N)_{ij} = \mathrm{Cov}(z_i, z_j) = \sigma(i-j) = \sigma(|i-j|)$$

where the covariance function σ characterizes this time series process. By taking variances of both sides of (B.1), the covariance function σ can be determined from

$$(B.3) \qquad \mathrm{Cov}\left(\sum_{i=0}^{p} \phi_i z_{t-i}, \sum_{j=0}^{p} \phi_j z_{t+s-j}\right) = \sum_{i=0}^{p}\sum_{j=0}^{p} \phi_i \phi_j \sigma(s+i-j)\sigma_e^2$$

$$= \sum_{j=0}^{q} \theta_j \theta_{j+s} \sigma_e^2$$

for any integer $s \geq 0$, where $\phi_0 = \theta_0 = -1$ and $\phi_i = \theta_j = 0$ for $i > p$ or $j > q$. (See Anderson (1971, p. 237); McLeod (1975) gives an algorithm for computing σ.)

Since forecasting is of primary interest, the distribution of the future n observations, that is, $z_F = (z_{N+1}, \dots, z_{N+n})^T$, conditional on the observed z, has a multivariate normal distribution with mean $\mu 1_n + A_{21}A_N^{-1}(z - \mu 1_N)$ and covariance matrix $\sigma_e^2 A_{nN}^*$ where

$$A_{N+n} = \begin{pmatrix} A_N & A_{12} \\ A_{21} & A_n \end{pmatrix} \begin{matrix} \} \, N \\ \} \, n \end{matrix}$$

and $A_{nN}^* = A_n - A_{21}A_N^{-1}A_{12}$. In the notation to be used here, this is written

(B.4) $$(z_F|\mu,\sigma_e^2,z) \sim N(\mu 1_n + A_{21}A_N^{-1}(z-\mu 1_N),\ \sigma_e^2 A_{nN}^*) \ .$$

Notice that A_{N+n} is a function of $\psi = (\phi_1,\ldots,\phi_p,\ \theta_1,\ldots,\theta_q)^T$.

C. Statistics for Maximum Likelihood Analysis

The probability model for the observation vector z given in Section B can be written as

(C.1) $$(z|\mu,\sigma_e^2,\psi) \sim N(\mu,\sigma_e^2 A_N(\psi))$$

Hence the log-likelihood function can be written as

(C.2) $$\ell(\mu,\sigma_e^2,\psi) = \text{constant} - \frac{N}{2}\ell n\sigma_e^2 - \tfrac{1}{2}\ell n|A_N| - \tfrac{1}{2}(z-\mu 1_N)^T A_N^{-1}(z-\mu 1_N) .$$

The function ℓ above can be easily maximized with respect to μ and σ_e^2 (for given ψ) by setting

$$\mu = \hat{\mu} = \frac{1_N^T A_N^{-1} z}{1_N A^{-1} 1_N}$$

$$\sigma_e^2 = \hat{\sigma}_e^2 = \frac{1}{N} Q(\psi)$$

where $$Q(\psi) = (z-\hat{\mu}1_N)^T A_N^{-1}(z-\hat{\mu}1_N) .$$

The resultant concentrated log-likelihood function is

$$\ell_3(\psi) = -\frac{N}{2}\ell n\, Q(\psi) - \tfrac{1}{2}\ell n|A_N|$$

which differs from $\ell(\hat{\mu},\hat{\sigma}_e^2,\psi)$ by a constant. It should now be apparent that the statistics needed to do maximum likelihood estimation are the following:

(C.3a) $$= 1_N^T A_N^{-1} z / 1_N^T A_N^{-1} 1_N = \text{MUHAT}$$

(C.3b) $$Q(\psi) = \text{QPSI}$$

(C.3c) $$|A_N| = \text{DETAN}*2^{\text{IXPDET}}$$

where MUHAT, QPSI, DETAN and IXPDET are the variable names used in the program ARMAML to carry this information. Also desired for the maximum likelihood analysis is the forecast function and its associated covariance matrix (from B.4).

(C.4a) $$\hat{z}_F = \hat{\mu} + A_{21}A_N^{-1}(z-\hat{\mu}1_N) = \text{ZFHAT(J),J = 1,NS}$$

(C.4b) $$A^*_{nN} = A_n - A_{21}A_N^{-1}A_{12} = \text{ANNSTR(I,J),I,J = 1,NS}$$

where NS = n .

D. Statistics for Bayesian Analysis

The computations needed for a Bayesian arise from two sources: the unconditional distribution of the observations z and the distribution of the forecasts. From Monahan (1980, D.4d), the density of z is

$$\text{(D.1)} \qquad p(z) = \frac{\Gamma(N/2+\alpha)\,|A_N + \tau^{-1} 1_N 1_N|^{-\frac{1}{2}} \left(\frac{\alpha}{\beta}\right)^{N/2}}{\Gamma(\alpha)(2\pi\alpha)^{N/2}\,[1 + \frac{1}{2\beta}(z-\gamma 1_N)^T (A + \tau^{-1} 1_N 1_N^T)^{-1}(z-\gamma 1_N)]^{\alpha+N/2}}$$

Notice $|A_N + \tau^{-1} 1_N 1_N^T| = |A_N|(\tau^*/\tau)$ where $\tau^* = \tau + 1_N^T A_N^{-1} 1_N$. Two necessary computations are obviously

$$\text{(D.2a)} \qquad |A_N| = \text{DETAN*2**IXPDET}$$

and

$$(z-\gamma 1_N)^T (A_N + \tau^{-1} 1_N 1_N^T)^{-1} (z-\gamma 1_N)$$

$$= (z-\gamma 1_N)^T (A_N^{-1} - A_N^{-1} 1_N 1_N^T A_N^{-1}/\tau^*)(z-\gamma 1_N)$$

$$= z^T A_N^{-1} z - 2\gamma 1_N^T A_N^{-1} z + \gamma^2 1_N^T A_N^{-1} 1_N$$

$$- (1_N^T A_N^{-1} z)^2/\tau^* + 2\gamma (1_N^T A_N^{-1} z)(1_N^T A_N^{-1} 1_N)/\tau^* - (1_N^T A_N^{-1} 1_N)^2/\tau^*$$

$$= z^T A_N^{-1} z - 2\gamma 1_N^T A_N^{-1} z + \gamma^2 1_N^T A_N^{-1} 1_N - (1_N^T A_N^{-1} z - \gamma 1_N^T A_N^{-1} 1_N)^2/\tau^* = \text{QF}$$

The posterior distribution of $(\mu|r,z)$ has a mean of

$$\text{(D.2c)} \qquad \gamma^* = (\tau\gamma + 1_N^T A_N^{-1} z)/\tau^* = \text{GAMST}$$

and variance $(r\tau^*)^{-1}$, where

$$\text{(D.3d)} \qquad \tau^* = \tau + 1_N^T A_N^{-1} 1 = \text{TAUST}$$

The posterior distribution of the forecasts is

(D.3) $$(z_F|z) \sim t_{n,2\alpha+N}\left(\gamma^* a+b, \left(\frac{2\alpha+N}{2\beta^*}\right)(A^*_{nN} + aa^T/\tau^*)^{-1}\right)$$

where $a = 1_n - A_{21}A_N^{-1}1_N$, $b = A_{21}A_N^{-1}z$ and $\beta^* = \beta + \frac{1}{2}QF$, with QF given above. Hence the mean of this posterior is needed,

(D.4a) $$\gamma^* a + b = ZFM(J),\ J = 1,NS$$

as well as the following covariance matrix:

(D.4b) $$A^{**}_{nN} = A^*_{nN} + aa^T/\tau^* = ANNST2(I,J), I,J = 1,NS$$

where, again, NS = n. Notice also that $(r|z) \sim \text{gamma}(\alpha+N/2,\ \beta^*)$ with β^* as given above is available through QF. The evaluation of the gamma functions in (D.1) require only α and N and can be done externally.

E. Algorithm ARMAML

The algorithm to be described below is basically a modification of Ansley's (1978a, 1979). Consider the $(N+n)$ by $(N+n)$ matrix B^m_{N+n} partitioned as

(E.1) $$B^m_{N+n} = \begin{pmatrix} B_{m,N} & 0 \\ B_1 & B_n \end{pmatrix}$$

where $B_{m,N}$ is N by N and each row of $(B_1 B_n)$ includes $(-\phi_p -\phi_{p-1} \cdots -\phi_1 \; 1)$, with zeroes elsewhere, staggered such that B_n is unit lower triangular. Now $B_{m,N}$ is also partitioned (with $m = \max(p,q)$)

(E.2) $$B_{m,N} = \begin{pmatrix} I_m & 0 \\ B_3 & B_{N-m} \end{pmatrix}$$

so that $(B_3 B_{N-m})$ has the same structure as $(B_1 B_n)$. Note B_n and B_{N-m} have the same structure: unit lower triangular and banded. If $p = 0$ then $B^m_{N+n} = I_{N+n}$.

(E.3) $$\left(B^m_{N+n}\right)_{k\ell} = \begin{cases} 0 & \text{if } k < \ell \\ 1 & \text{if } k = \ell \\ 1 & \text{if } k \le m \text{ and } k \ne \ell \\ -\phi_{k-\ell} & \text{if } k > m \text{ and } p \ge k-\ell > 0 \\ 0 & \text{if } k > m \text{ and } k-\ell > p \end{cases}$$

Now note the following product:

$$\text{(E.4)} \quad B^m_{N+n}A_{N+n}(B^m_{N+n})^T = \begin{bmatrix} B_{m,N}A_NB^T_{m,N} & B_{m,N}A_NB^T_1 + B_{m,N}A_{12}B^T_n \\ B_1A_NB^T_{m,N} + B_nA_{21}B^T_{m,N} & \begin{matrix} B_1A_NB^T_1 + B_1A_{12}B^T_n \\ + B_nA_{22}B^T_n + B_nA_{21}B^T_1 \end{matrix} \end{bmatrix}$$

But this product can be partitioned:

$$\text{(E.5)} \qquad B^m_{N+n}A_{N+n}(B^m_{N+n})^T = \begin{bmatrix} A_m & D^T_1 & 0 \\ D_1 & C_1 & E^T \\ 0 & E & C_2 \end{bmatrix} \begin{matrix} m \\ N-m \\ n \end{matrix}$$

$$\qquad\qquad\qquad\qquad\qquad\qquad \begin{matrix} m & N-m & n \end{matrix} .$$

where A_m is the covariance matrix of ARMA process of length m and C_1(N-m by N-m), C_2, and the last (N+n-m) rows and columns, are the covariance matrices of a MA process of appropriate length with the same parameters (q and θ) as always. Hence all of the matrices are banded and can be mostly zeroes. See Section G: Algorithm GETSET for the derivations of A_m, D_1, E, C_1, C_2, and 0.

The crux of the algorithm is that the matrix $B_{m,N}A_NB^T_{m,N}$ is a symmetric, positive definite banded matrix, with bandwidth not exceeding m, and, hence, has a factorization of the form LDL^T where L is unit lower triangular and banded and D is diagonal. This factorization is slightly different than that used by Ansley (1979) which can be found (in Algol) in Martin and Wilkinson (1971). The LDL^T factorization avoids computing square roots. Note also, since $|B_{m,N}| = 1$, that $|A_N| = |D|$, which is easily computed.

To compute the bilinear forms in 1_N, z, and A_N^{-1}, say, compute $L^{-1}B_{m,N}1_N$, $L^{-1}B_{m,N}z$, then

$$(L^{-1}B_{m,N}z)^T D^{-1}(L^{-1}B_{m,N}1_N) = z^T B_{m,N}^T (LDL^T)^{-1} B_{m,N}1_N$$

(E.6)
$$= z^T B_{m,N}^T (B_{m,N}A_N B_{m,N}^T)^{-1} B_{m,N}1_N$$

$$= z^T A_N^{-1} 1_N$$

and similarly obtain $1_n^T A_N^{-1} 1_N$ and $z^T A_N^{-1} z$.

To compute the forecasts, note that for an arbitrary vector d,

$$(0\ E^T)L^{-T}D^{-1}(L^{-1}B_{m,N}d) = (0\ E^T)(LDL^T)^{-1}B_{m,N}d$$

(E.7)
$$= (B_1 A_N B_{m,N}^T + B_n A_{21} B_{m,N}^T)(B_{m,N}A_N B_{m,N}^T)^{-1} B_{m,N}d .$$

$$= B_1 d + B_n A_{21} A_N^{-1} d .$$

(E.8)
$$A_{21}A_N^{-1}d = B_n^{-1}[(0\ E^T)L^{-T}D^{-1}(L^{-1}B_{m,N}d) - B_1 d]$$

Since the necessary expression is $1_n + A_{21}A_n^{-1}z - A_{21}A_N^{-1}1_N$, substituting 1_N and z for d in (E.8) leaves the B_n^{-1} and $B_1 d$ as the only extra computation (see below for $(0\ E^T)L^T$).

Now to compute the covariance matrix for the forecasts: $A_{22} - A_{21}A_N^{-1}A_{12}$, $A_{22} - A_{21}A_N^{-1}A_{12}$, note the following expressions:

$$C_2 - (0\ E^T)(LDL^T)^{-1}\begin{pmatrix}0\\E\end{pmatrix} = B_1 A_N B_1^T + B_2 A_{21} B_1^T + B_1 A_{12} B_2^T + B_2 A_{22} B_2^T$$

$$-(B_1 A_n B_{m,N}^T + B_2 A_{21} B_{m,N}^T)(B_{m,N}A_N B_{m,N})^{-1}(B_{m,N}A_N B_1^T + B_{m,N}A_{12}B_n^T)$$

(E.9)
$$= B_n A_{22} B_n^T - B_n A_{21} A_N^{-1} A_{12} B_n^T$$

Hence the needed computation is merely

$$B_n^{-1}\left[C_2 - \left\{L^{-1}\begin{pmatrix}0\\E^T\end{pmatrix}\right\}^T D^{-1}\left\{L^{-1}\begin{pmatrix}0\\E^T\end{pmatrix}\right\}\right] B_n^{-T} \tag{E.10}$$

Note that the expression in braces above is the one needed in (E.8).

F. Algorithm BARMAN

Algorithm BARMAN follows ARMAML closely, since the computational needs of the Bayesian and maximum likelihood analyses are so similar. For example, the determinant of A_N is needed by both. Likewise, the bilinear forms in z, 1_N and A_N^{-1} computed in ARMAML suffice for the computation of GAMST, TAUST, and QF.

Their computational demands differ in the need to explicitly compute $a = 1_n - A_{21}A_N^{-1}1_N$. In ARMAML, the forecasting vector is $\hat{\mu}\, a + b$, where $b = A_{21}A_N^{-1}z$, and this is computed implicitly using (E.8). In BARMAN, a is needed in the covariance matrix computations, therefore both a and b are computed explicitly, stored in arrays AS and BS, also following (E.8), with 1_N and z replacing d. The mean of the forecast distribution is then easily formed as $\gamma^* a + b$. The covariance matrix for the Bayesian forecasts can be rewritten as

$$A_{nN}^{**} = A_{nN}^{*} + aa^T/\tau^*.$$

With a and τ^* already computed, A_{nN}^{**} is then formed by following ARMAML to compute A_{nN}^{*} and adding on aa^T/τ^*.

G. Algorithm GETSET

Algorithm GRETA provides ARMAML/BARMAN with the elements of

$$B^m_{N+n} A_{N+n} (B^m_{N+n})^T = \begin{pmatrix} A_m & D_1^T & 0 \\ D_1 & C_1 & E^T \\ 0 & E & C_2 \end{pmatrix} \tag{G.1}$$

quickly, retrieving the values stored in a common area named XARMAX, in variables, M, IQ1, AM, CC, and D1. The first two, M and IQ1, are integers, with $M = m \equiv \max(p,q)$ and $IQ1 = q + 1$. The last three are vectors of length 10 and will be described below.

Since the efficiency of these algorithms depend on the patterns of the matrix in (G.1), the nature of A_m, D_1, C_1, C_2, E and 0 must be discussed. The patterns in these submatrices arise from (B.3) which will be rewritten as

$$\sum_{i=0}^{p} \sum_{j=0}^{p} \phi_i \phi_j \sigma(s+i+j) = \sum_{j=0}^{q} \theta_j \theta_{j+s} \equiv c(s) \quad , \tag{G.2}$$

thus defining $c(s)$. A simplification can be obtained for $s > m$, which leaves $c(s) = 0$:

$$\mathrm{Cov}(z_t, z_{t+s} - \phi_1 z_{t+s-1} \cdots - \phi_p z_{t+s-p}) = \mathrm{Cov}(z_t, e_{t+s} - \theta_1 e_{t+s-1} \cdots - \theta_q e_{t+s-q})$$

$$= \sum_{j=0}^{p} \phi_j \sigma(s-j) = 0 \quad \text{if } s > m \equiv \max(p,q), \tag{G.3}$$

since z_t is uncorrelated with future disturbances.

Consider now the cases for the (k,ℓ)th elements of the matrix given in (G.1):

Case 1 $1 \le k \le m,\ 1 \le \ell \le m$

$$(B^m_{N+n} A_{N+n} (B^m_{N+n})^T)_{k\ell} = \sum_{i=1}^{N+n} \sum_{j=1}^{N+n} (B^m_{N+n})_{ki} (B^m_{N+n})_{\ell j} \sigma(i-j)$$

$$= \sigma(k-\ell) \ .$$

Case 2 $m < k \le N + n,\ m < \ell \le N + n$

$$(B^m_{N+n} A_{N+n} (B^m_{N+n})^T_{k-\ell} = \sum_{i=1}^{N+n} \sum_{j=1}^{N+n} (B^m_{N+n})_{ki} (B^m_{N+n})_{\ell j} \sigma(i-j)$$

$$= \sum_{i=k-p}^{k} \sum_{j=\ell-p}^{\ell} \phi_{k-i} \phi_{\ell-j} \sigma(i-j$$

$$\left.\begin{array}{l} i^* = k - i \\ j^* = \ell - j \end{array}\right\} \longrightarrow$$

$$= \sum_{i^*=0}^{p} \sum_{j^*=0}^{p} \phi_{i^*} \phi_{j^*} \sigma(k-i^*-\ell+j^*)$$

$$= \sum_{j=0}^{q} \theta_j \theta_{j+(k-\ell)} = c(k-\ell)$$

Case 3 $1 \le \ell \le m,\ m < k \le N + n$

$$(B^m_{N+n} A_{N+n} (B^m_{N+n})^T)_{k\ell} = \sum_{i=1}^{N+n} \sum_{j=1}^{N+n} (B^m_{N+n})_{ki} (B^m_{N+n})_{\ell j} \sigma(i-j)$$

$$= \sum_{i=1}^{N+n} (B^m_{N+n})_{ki}\ \sigma(i-\ell)$$

$$= \sum_{i=k-p}^{k} \phi_{k-i} \sigma(i-\ell)$$

$$= \sum_{i^*=0}^{k} \phi_{i^*}\ \sigma((k-\ell)-i^*)$$

From Case 1 it is apparent that A_m is the covariance matrix from the same ARMA (p,q) process, but of length m. Hence the vector AM holds $\sigma(0)$ through $\sigma(m)$ in AM(1) through AM(M+1). From Case 2, it can be seen that the submatrix formed by C_1, C_2 and E is the covariance matrix of a MA(q) series of length $N + n - m$, hence this submatrix is Toeplitz and has a band width of $q + 1$. Here the vector CC holds c(0) through c(q+1) in CC(1) through CC(IQ1). From Case 3, the (k,ℓ)th element,

$$\sum_{i=0}^{p} \phi_i \sigma((k-\ell)-i)$$

can be found in $D1((k-\ell))$. The first one is D1(1); the last is D1(M). If $(k-\ell)$ exceeds m, (G.3) applies, and the remainder of the region D_1 is found to be zero, as well as the corner regions denoted by 0.

The vectors CC and D1 are formed in GETSET. For the vector AM, CFARMA is called, which uses McLeod's (1975) algorithm to find $\sigma(0)$ through $\sigma(m)$.

H. Other Algorithms

Algorithms ARMAML, BARMAN, GRETA and GETSET have already been described. The remaining three can be described simply.

Algorithm CFARMA computes the covariance function $\sigma(0)$ through $\sigma(m)$, to be stored in the vector AM. CFARMA is called by GETSET; CFARMA calls GESEPP. The computations in CFARMA are nearly identical to those prescribed by McLeod (1975).

Algorithm GESEPP is a general linear equations solver. The method used is Gaussian elimination with partial pivoting.

Algorithm ADJUST implements a device for avoiding overflow in the computation of the determinant of A_N by storing it in the form $\text{DETAN} \cdot 2^{\text{IXPDET}}$. In general, the input D,I are modified by multiplying or dividing D by 16, and, simultaneously, subtracting or adding 4 to I, a sufficient number of times such that $1.0 \leq D \leq 16.0$ and $D \cdot 2^I$ does not change.

I. References

Anderson, T. W. (1970). *The Statistical Analysis of Time Series*, Wiley, New York.

Ansley, Craig F. (1978a). "Computation of the Exact Likelihood for an ARMA Process," *Proc. Computer Science and Statistics: Eleventh Symp. on the Interface*, pp. 71-78.

Ansley, Craig F. (1978b). Subroutine ARMA -- Exact Likelihood for Univariate ARMA Processes. Unpublished program documentation.

Ansley, Craig F. (1979). "An Algorithm for the Exact Likelihood of a Mixed Autoregressive-Moving Average Process," *Biometrika 66*, 1, pp. 59-65.

Martin, R. S. and J. H. Wilkinson. (1971). Symmetric Decomposition of Positive Definite Band Matrices, *Numerische Mathematik 7*, 355-61.

McLeod, Ian. (1975). Derivation of the Theoretical Autocovariance Function of Autoregressive-Moving Time Series, *Applied Statistics 24*, 2, pp. 255-6. Correction: *26*, p. 194.

Monahan, John F. (1980). A Structured Bayesian Approach to ARMA Time Series Models, Part I: Distributional Results, Institute of Statistics Mimeo Series No. 1297.

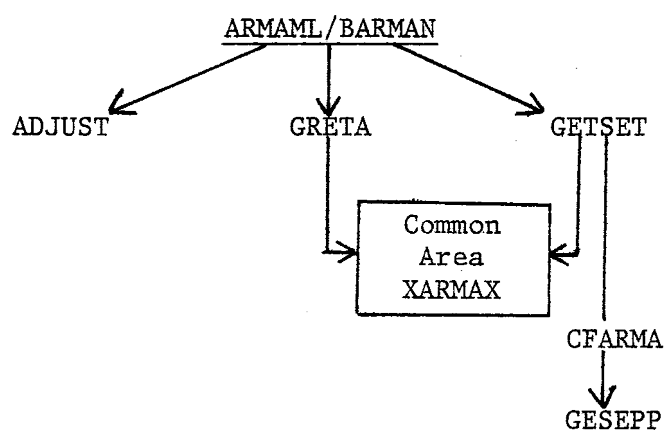
Trench, William F. (1964). An Algorithm for the Inversion of Finite Toeplitz Matrices, *J. Soc. Indust. Appl. Math.*, *12*, 3, pp. 515-522.

APPENDIX

Listing of Programs

ARMAML
BARMAN
GRETA
ADJUST
GETSET
CFARMA
GESEPP

The structure of the programs:



```
      SUBROUTINE ARMAML(Z,P,Q,N,PHI,THETA,MUHAT,DETAN,IXPDET,QPSI,ZFHAT,
     1ANNSTR,NS)
C   ARMAML    J F MONAHAN    CURRENT VERSION JULY 1980
C   ARMAML COMPUTES STATISTICS FOR MAXIMUM LIKELIHOOD ANALYSIS
C   OF ARMA TIME SERIES MODELS
C IN   Z,P,Q,PHI,THETA,N,NS
C OUT MUHAT,QPSI,ZFHAT,ANNSTR,DETAN,IXPDET
C IN  Z=SERIES OF LENGTH N
C IN  PHI(J),J=1,P   AUTOREGRESSIVE PARAMETERS
C IN  THETA(J),J=1,Q   MOVING AVERAGE PARAMETRS
C OUT MUHAT= ML ESTIMATE OF SERIES MEAN
C OUT QPSI= QUADRATIC FORM FOR LIKELIHOOD
C OUT DETAN*2**IXPDET= DETERMINANT OF A-SUB(N)
C OUT ZFHAT(J),J=1,NS   FORECASTS
C OUT ANNSTR(I,J),I,J=1,NS   COVARIANCE MATRIX
C *** RESTRICTIONS:
C *** O LE P,Q LE 9
C *** MAX(P,Q) LT N LE 500
C *** MAX(P,Q) LT NS LE 10
C *** NB: THE 9,500, AND 10 ARE  ARBITRARY; THEY JUST FIX ARRAY BOUNDS
      REAL*8 Z(500),PHI(10),THETA(10),ZFHAT(10),ANNSTR(10,10),MUHAT,
     1DETAN,QPSI,BZ(500),BO(500),E(10,10),L(100),ODO,ODZ,ZDZ,S,W,GRETA
      INTEGER P,Q
      LOC(I,J)=MP1*MOD(I-1,MP1)+J-I+MP1
C   D LIES ON DIAGONAL OF L
C   L STORED SLIDING, AND ONLY RECENT PART
      M=MAXO(P,Q)
      MP1=M+1
      CALL GETSET(PHI,THETA,P,Q)
C   GETSET PREPARES VALUES OF A,D_1,C_1,C_2 AND E
C   USES COMMON AREA NAMED XARMAX   ***************
      ODO=0.DO
      ODZ=0.DO
      ZDZ=0.DO
C   CREATE BZ  AND  B*ONE
      DO 2I=1,M
      BZ(I)=Z(I)
    2 BO(I)=1.DO
      S=1.DO
      IF (P.EQ.O) GO TO 6
      DO 4I=1,P
    4 S=S-PHI(I)
    6 DO 8I=MP1,N
      BZ(I)=Z(I)
      IF(P.EQ.O) GO TO 8
      DO 9J=1,P
    9 BZ(I)=BZ(I)-PHI(J)*Z(I-J)
    8 BO(I)=S
C   BZ HOLDS B_M,N * Z
C   BO HOLDS B_M,N * ONE
      DETAN=1.DO
      IXPDET=0
```

```
        DO 12I=1,N
        W=GRETA(I,I)
C   GRETA GETS VALUES OF B_M,N * A_N * (B_M,N)-TRANSPOSE
C   AKA   A_M, C_1, C_2,D_1, AND E
        IF (I.EQ.1) GO TO 14
        IMQ=1
        IF(I.LE.M)  GO TO 11
        IF(Q.EQ.O) GO TO  14
        IMQ=I-Q
   11   IM1=I-1
        DO 15J=IMQ,IM1
        S=GRETA(I,J)
        IF(IMQ.GE.J) GO TO 15
        JM1=J-1
        DO 16K=IMQ,JM1
   16   S=S-L(LOC(J,K))*L(LOC(I,K))
   15   L(LOC(I,J))=S
        DO 17J=IMQ,IM1
        S=L(LOC(I,J))
        L(LOC(I,J))=S/L(LOC(J,J))
        W=W-S*L(LOC(I,J))
        BZ(I)=BZ(I)-L(LOC(I,J))*BZ(J)
   17   BO(I)=BO(I)-L(LOC(I,J))*BO(J)
   14   CONTINUE
        L(LOC(I,I))=W
C   COMPUTE BILINEAR FORMS
        ODO=ODO+BO(I)*BO(I)/W
        ZDZ=ZDZ+BZ(I)*BZ(I)/W
        ODZ=ODZ+BO(I)*BZ(I)/W
        DETAN=DETAN*W
        CALL ADJUST(DETAN,IXPDET)
   12   CONTINUE
C   GET ESTIMATE OF MEAN
        MUHAT=ODZ/ODO
C  GET QUADRATIC FORM FOR LIKELIHOOD
        QPSI=ZDZ-ODZ*MUHAT
C  NOW START ON FORECASTS
        DO 30 I=1,NS
        ZFHAT(I)=0.DO
        DO 30 J=1,I
C  INITIALIZE COV MX FOR FORECASTS
   30   ANNSTR(I,J)=GRETA(N+I,N+J)
        IF (Q.EQ.O) GO TO 39
C  GET D-INV  L-INV  ( O  E )
        DO 32K=1,Q
        DO 33I=K,Q
        E(I,K)=GRETA(N+K,N-Q+I)
        IF(I.EQ.K) GO TO 33
        IM1=I-1
        DO 34J=K,IM1
   34   E(I,K)=E(I,K)-L(LOC(N-Q+I,N-Q+J))*E(J,K)
   33   CONTINUE
```

```
      DO 36I=K,Q
      S=E(I,K)
      KM1=K-1
      IF(KM1.EQ.0) GO TO 38
      DO 37J=1,KM1
   37 ANNSTR(K,J)=ANNSTR(K,J)-S*E(I,J)
   38 E(I,K)=E(I,K)/L(LOC(N-Q+I,N-Q+I))
      ANNSTR(K,K)=ANNSTR(K,K)-S*E(I,K)
   36 ZFHAT(K)=ZFHAT(K)+E(I,K)*(BZ(N-Q+I)-MUHAT*BO(N-Q+I))
   32 CONTINUE
   39 CONTINUE
C   SYMMETRIZE
      DO 51I=1,NS
      DO 51J=1,I
   51 ANNSTR(J,I)=ANNSTR(I,J)
      IF(P.EQ.0) GO TO 48
      DO 41I=1,P
      DO 41J=I,P
   41 ZFHAT(I)=ZFHAT(I)+PHI(J)*(Z(N+I-J)-MUHAT)
C   FINISH FORECASTS WITH B_NS-INV
      DO 42I=1,NS
      IF(I.EQ.1) GO TO 42
      K=MIN0(I-1,P)
      DO 43 J=1,K
   43 ZFHAT(I)=ZFHAT(I)+PHI(J)*ZFHAT(I-J)
   42 CONTINUE
C  FORECASTS NEAR DONE   NOW GET COV MX
C   B_NS-INV IN FRONT
      DO 52J=1,NS
      DO 52I=1,NS
      LL=MIN0(I-1,P)
      IF(I.EQ.1) GO TO 52
      DO 53K=1,LL
   53 ANNSTR(I,J)=ANNSTR(I,J)+PHI(K)*ANNSTR(I-K,J)
   52 CONTINUE
C   B_NS-INV  TRANSP IN BACK
      DO 56J=1,NS
      DO 56I=1,NS
      IF(I.EQ.1) GO TO 56
      LL=MIN0(I-1,P)
      DO 55K=1,LL
   55 ANNSTR(J,I)=ANNSTR(J,I)+PHI(K)*ANNSTR(J,I-K)
   56 CONTINUE
   48 CONTINUE
      DO 49I=1,NS
   49 ZFHAT(I)=ZFHAT(I)+MUHAT
      RETURN
      END
```

```
      SUBROUTINE BARMAN(Z,P,Q,N,PHI,THETA,GAMST,DETAN,IXPDET,QF,ZFM,
     1ANNST2,NS,GAM,TAU,TAUST)
C   BARMAN    J F MONAHAN   CURRENT VERSION JULY 1980
C   BARMAN COMPUTES STATISTICS FOR A STRUCTURED BAYESIAN ANALYSIS
C   OF ARMA TIME SERIES MODELS
C IN  Z,P,Q,PHI,THETA,N,NS,GAM,TAU
C OUT GAMST,TAUST,QF,ZFM,ANNST2,DETAN,IXPDET
C *** MU=SERIES MEAN        R=DISTURBANCE PRECISION
C IN  PRIOR ON (MU ¦ R) IS NORMAL(GAM,1/(TAU*R))
C IN  Z=SERIES OF LENGTH N
C IN  PHI(J),J=1,P   AUTOREGRESSIVE PARAMETERS
C IN  THETA(J),J=1,Q   MOVING AVERAGE PARAMETRS
C OUT GAMST= MEAN OF POSTERIOR OF (MU¦R)
C OUT TAUST: 1/(TAUST*R)= VARIANCE OF POSTERIOR OF (MU¦R)
C OUT QF= QUADRATIC FORM FOR DENSITY OF (Z)
C OUT DETAN*2**IXPDET= DETERMINANT OF A-SUB(N)
C OUT ZFM(J),J=1,NS  MEAN VECTOR OF FORECAST DISTRIBUTION
C OUT ANNST2(I,J),I,J=1,NS   COVARIANCE MATRIX
C *** RESTRICTIONS:
C *** O LE P,Q LE 9
C *** MAX(P,Q) LT N LE 500
C *** MAX(P,Q) LT NS LE 10
C *** NB: THE 9,500, AND 10 ARE  ARBITRARY; THEY JUST FIX ARRAY BOUNDS
      REAL*8 Z(500),PHI(10),THETA(10),GAMST,DETAN,QF,ZFM(10),
     1ANNST2(10,10),GAM,TAU,TAUST,BZ(500),BO(500),E(10,10),AS(10),BS(10)
     2,GRETA,L(100),S,ODO,ODZ,ZDZ,W
      INTEGER P,Q
      LOC(I,J)=MP1*MOD(I-1,MP1)+J-I+MP1
C   D LIES ON DIAGONAL OF L
C   L STORED SLIDING, AND ONLY RECENT PART
      M=MAXO(P,Q)
      MP1=M+1
      CALL GETSET(PHI,THETA,P,Q)
C   GETSET PREPARES VALUES OF A,D_1,C_1,C_2 AND E
C   USES COMMON AREA NAMED XARMAX   ***************
      ODO=0.DO
      ODZ=0.DO
      ZDZ=0.DO
C   CREATE BZ  AND  B*ONE
      DO 2I=1,M
      BZ(I)=Z(I)
    2 BO(I)=1.DO
      S=1.DO
      IF (P.EQ.O) GO TO 6
      DO 4I=1,P
    4 S=S-PHI(I)
    6 DO 8I=MP1,N
      BZ(I)=Z(I)
      IF(P.EQ.O) GO TO 8
      DO 9J=1,P
    9 BZ(I)=BZ(I)-PHI(J)*Z(I-J)
    8 BO(I)=S
C   BZ HOLDS B_M,N * Z
C   BO HOLDS B_M,N * ONE
      DETAN=1.
      IXPDET=0
```

```
      DO 12I=1,N
      W=GRETA(I,I)
C   GRETA GETS VALUES OF B_M,N * A_N * (B_M,N)-TRANSPOSE
C   AKA   A_M, C_1, C_2,D_1, AND E
      IF (I.EQ.1) GO TO 14
      IMQ=1
      IF(I.LE.M)  GO TO 11
      IF(Q.EQ.0) GO TO  14
      IMQ=I-Q
 11   IM1=I-1
      DO 15J=IMQ,IM1
      S=GRETA(I,J)
      IF(IMQ.GE.J) GO TO 15
      JM1=J-1
      DO 16K=IMQ,JM1
 16   S=S-L(LOC(J,K))*L(LOC(I,K))
 15   L(LOC(I,J))=S
      DO 17J=IMQ,IM1
      S=L(LOC(I,J))
      L(LOC(I,J))=S/L(LOC(J,J))
      W=W-S*L(LOC(I,J))
      BZ(I)=BZ(I)-L(LOC(I,J))*BZ(J)
 17   BO(I)=BO(I)-L(LOC(I,J))*BO(J)
 14   CONTINUE
      L(LOC(I,I))=W
C   COMPUTE BILINEAR FORMS
      ODO=ODO+BO(I)*BO(I)/W
      ZDZ=ZDZ+BZ(I)*BZ(I)/W
      ODZ=ODZ+BO(I)*BZ(I)/W
      DETAN=DETAN*W
      CALL ADJUST(DETAN,IXPDET)
 12   CONTINUE
      TAUST=TAU+ODO
C   GET MEAN OF POSTERIOR OF MU**** E(MU GIVEN Z)
      GAMST=(GAM*TAU+ODZ)/TAUST
C  GET QUADRATIC FORM FOR DENSITY
      QF=ZDZ+GAM*GAM*TAU-GAMST*(GAM*TAU+ODZ)
C  NOW START ON FORECASTS
      DO 30 I=1,NS
      ZFM(I)=0.D0
      AS(I)=0.D0
      BS(I)=0.D0
      DO 30 J=1,I
C  INITIALIZE COV MX FOR FORECASTS
 30   ANNST2(I,J)=GRETA(N+I,N+J)
      IF (Q.EQ.0) GO TO 39
C  GET D-INV  L-INV  ( O  E )
      DO 32K=1,Q
      DO 33I=K,Q
      E(I,K)=GRETA(N+K,N-Q+I)
      IF(I.EQ.K) GO TO 33
      IM1=I-1
      DO 34J=K,IM1
 34   E(I,K)=E(I,K)-L(LOC(N-Q+I,N-Q+J))*E(J,K)
 33   CONTINUE
```

```
      DO 36I=K,Q
      S=E(I,K)
      KM1=K-1
      IF(KM1.EQ.0) GO TO 38
      DO 37J=1,KM1
37    ANNST2(K,J)=ANNST2(K,J)-S*E(I,J)
38    E(I,K)=E(I,K)/L(LOC(N-Q+I,N-Q+I))
      ANNST2(K,K)=ANNST2(K,K)-S*E(I,K)
      AS(K)=AS(K)+E(I,K)*BO(N-Q+I)
36    BS(K)=BS(K)+E(I,K)*BZ(N-Q+I)
32    CONTINUE
39    CONTINUE
C   SYMMETRIZE
      DO 51I=1,NS
      DO 51J=1,I
51    ANNST2(J,I)=ANNST2(I,J)
      IF(P.EQ.0) GO TO 48
      DO 41I=1,P
      DO 41J=I,P
      AS(I)=AS(I)+PHI(J)
41    BS(I)=BS(I)+PHI(J)*Z(N+I-J)
C   FINISH FORECASTS WITH B_NS-INV
      DO 42I=1,NS
      IF(I.EQ.1) GO TO 42
      K=MINO(I-1,P)
      DO 43 J=1,K
      AS(I)=AS(I)+PHI(J)*AS(I-J)
43    BS(I)=BS(I)+PHI(J)*BS(I-J)
42    CONTINUE
C  FORECASTS NEAR DONE   NOW GET COV MX
C   B_NS-INV IN FRONT
      DO 52J=1,NS
      DO 52I=1,NS
      LL=MINO(I-1,P)
      IF(I.EQ.1) GO TO 52
      DO 53K=1,LL
53    ANNST2(I,J)=ANNST2(I,J)+PHI(K)*ANNST2(I-K,J)
52    CONTINUE
C   B_NS-INV  TRANSP IN BACK
      DO 56J=1,NS
      DO 56I=1,NS
      IF(I.EQ.1) GO TO 56
      LL=MINO(I-1,P)
      DO 55K=1,LL
55    ANNST2(J,I)=ANNST2(J,I)+PHI(K)*ANNST2(J,I-K)
56    CONTINUE
48    CONTINUE
      DO 49I=1,NS
      AS(I)=1.DO-AS(I)
49    ZFM(I)=GAMST*AS(I)+BS(I)
      DO 58 I=1,NS
      DO 58 J=1,NS
58    ANNST2(I,J)=ANNST2(I,J)+AS(I)*AS(J)/TAUST
      RETURN
      END
```

```
      DOUBLE PRECISION FUNCTION GRETA(I,J)
C   GRETA GETS (QUICKLY) VALUES OF B_M,N * A_N * (B_M,N)-TRANSPOSE
      REAL*8 AM(10),D1(10),CC(10)
      COMMON /XARMAX/M,IQ1,AM,CC,D1
      K=I-J+1
C  MOST CALLS ARE HERE, DO THIS FIRST
      GRETA=0.D0
      IF(K.LE.IQ1) GRETA=CC(K)
C  REGIONS C1, C2, AND E NOW DONE
      IF(J.GT.M) RETURN
      IF(I.GT.M) GO TO 4
C  BOTH I AND J LE M ....RETURN FROM A-SUB(M)
      GRETA=AM(K)
      RETURN
C  J LE M , I GT M   ...RETURN FROM D-SUB(1)
    4 GRETA=D1(K-1)
      RETURN
      END
      SUBROUTINE ADJUST(D,I)
      REAL*8 D
C   ADJUST KEEPS DET FROM EXPLODING
      IF(D.LE.0.D0) GO TO 6
    3 IF(D.GE.1.D0) GO TO 4
      D=D*16.D0
      I=I-4
      GO TO 3
    4 IF(D.LE.16.D0) RETURN
      D=D/16.D0
      I=I+4
      GO TO 4
    6 I=-2147483644
      RETURN
      END
```

```
      SUBROUTINE GETSET(PHI,THETA,P,Q)
C  GETSET COMPUTES THE VALUES NEEDED BY ARMAML AND/OR BARMAN
C  GRETA CALLS THESE FOR B-M-SUB(N+NS)*A-SUB(M)*(B-M-SUB(N+NS))-TRANSPOS
      REAL*8 PHI(10),THETA(10),AM(10),CC(10),D1(10)
      INTEGER P,Q
      COMMON /XARMAX/M,IQ1,AM,CC,D1
      IQ1=Q+1
      M=MAXO(P,Q)
C  CFARMA FIND THE ESSENTIAL ELEMENTS OF AM, THE COVARIANCE FUNCTION
C  FOR ARMA(P,Q) PROCESS ... IN REGION A-SUB(M)...  M=MAX(P,Q)
      CALL CFARMA(PHI,THETA,P,Q,AM)
C  CC HOLDS THE COVARIANCE FUNCTION OF AN MA(Q) PROCESS
C  REGIONS C1,C2, AND E WILL USE CC
      CC(1)=1.
      IF(M.EQ.0) RETURN
      IF(Q.EQ.0) GO TO 6
      DO 2I=1,Q
  2   CC(1)=CC(1)+THETA(I)**2
      DO 4I=1,Q
      IP1=I+1
      CC(I+1)=-THETA(I)
      IF(I.EQ.Q) GO TO 4
      DO 5J=IP1,Q
  5   CC(I+1)=CC(IP1)+THETA(J)*THETA(J-I)
  4   CONTINUE
  6   CONTINUE
C  REGION D-SUB(1) USES D1
      DO 7I=1,M
      D1(I)=AM(I+1)
      IF(P.EQ.0) GO TO 7
      DO 8K=1,P
  8   D1(I)=D1(I)-PHI(K)*AM(IABS(I-K)+1)
  7   CONTINUE
      RETURN
      END
```

```
      SUBROUTINE CFARMA(PHI,THETA,P,Q,X)
C   COMPUTES COVARIANCE FUNCTION, SIGMA(O),...,SIGMA(M)   M=MAX(P,Q)
C FOR ARMA PROCESS USING MCLEOD'S ALGORITHM IN APPLIED STATISTICS (1975)
C  VOL 24 NO 2 PP255-256, CORRECTION VOL 26 P194
C     GESEPP IS LINEAR EQUATIONS SOLVER
      REAL*8 PHI(10),THETA(10),A(10,10),X(10),C(10),S
      INTEGER P,Q,R,RP1,QP1
      R=MAXO(P,Q)
      RP1=R+1
      C(1)=1.
      IF(Q.EQ.O) GO TO 4
      DO 2K=1,Q
      J=K+1
      C(J)=-THETA(K)
      IF(P.EQ.O) GO TO 2
      L=MINO(P,K)
      DO 3I=1,L
    3 C(J)=C(J)+PHI(I)*C(J-I)
    2 CONTINUE
    4 CONTINUE
      X(1)=C(1)
      DO 9I=2,10
    9 X(I)=O.
      IF(Q.EQ.O) GO TO 6
      DO 7I=1,Q
    7 X(1)=X(1)-THETA(I)*C(I+1)
      DO 8K=1,Q
      J=K+1
      DO 8I=K,Q
    8 X(J)=X(J)-THETA(I)*C(I-K+1)
    6 IF(P.EQ.O) RETURN
      DO 12I=1,RP1
      DO 12J=1,RP1
   12 A(I,J)=O.
C    NB  THIS IS NEG OF MACLEOD'S A
      DO 14I=1,RP1
      A(I,I)=1.
      DO 14J=1,P
      L=IABS(I-J-1)+1
   14 A(I,L)=A(I,L)-PHI(J)
      CALL GESEPP(A,RP1,S,X)
      RETURN
      END
```

```
      SUBROUTINE GESEPP(A,N,DET,B)
C GAUSSIAN ELIMINATION WITH PARTIAL PIVOTING
C  SOLVES A*X=B,  A IS DESTROYED,   B IS OVERWRITTEN WITH SOLUTION X
C  A IS N BY N, B IS N BY 1, 1 LE N LE 10, DIMENSION 10 IS ARBITRARY
C  DET= DETERMINANT OF A
C   J F MONAHAN     CURRENT VERSION JULY 1980
      REAL*8 A(10,10),B(10),DET,S,T
      DET=1.
      NM1=N-1
      IF(N.EQ.1) GO TO 7
      DO 6I=1,NM1
      IP1=I+1
      S=0.
      DO 1J=I,N
C  LOOK FOR GOOD PIVOT IN I-TH COLUMN
      IF(DABS(A(J,I)).LE.S) GO TO 3
      L=J
      S=DABS(A(J,I))
    1 CONTINUE
C   DON'T SWITCH IF AVOIDABLE
      IF(I.EQ.L) GO TO 6
      DO 2K=I,N
      T=A(I,K)
      A(I,K)=A(L,K)
    2 A(L,K)=T
      T=B(I)
      B(I)=B(L)
      B(L)=T
      DET=-DET
C   ELIMINATE
    3 DO 5J=IP1,N
      S=-A(J,I)/A(I,I)
      DO 4K=IP1,N
    4 A(J,K)=A(J,K)+S*A(I,K)
    5 B(J)=B(J)+S*B(I)
    6 DET=DET*A(I,I)
    7 DET=DET*A(N,N)
C  ELIMINATION DONE----NOW BACKSOLVE
      B(N)=B(N)/A(N,N)
      IF(N.EQ.1) RETURN
      DO 9J=1,NM1
      I=N-J
      IP1=I+1
      DO 8K=IP1,N
    8 B(I)=B(I)-A(I,K)*B(K)
    9 B(I)=B(I)/A(I,I)
      RETURN
      END
```