ABSTRACT

In the analysis of autoregressive-moving average time series models, the parameter space becomes unwieldy when the order of the model exceeds two. An efficient method is given for producing points in this region either for a random search (for maximum likelihood estimation) or for numerical integration (for a Bayesian analysis).

A. INTRODUCTION

Consider the autoregressive-moving average (ARMA) time series model given by the relation

$$(A.1) \quad z_t + \phi_1 z_{t-1} + \ldots + \phi_p z_{t-p} = e_t + \theta_1 e_{t-1} + \ldots + \theta_q e_{t-q}$$

where the disturbances e_t are independent and identically distributed random variables with mean zero and variance σ^2. The ARMA (p,q) model given by (A.1) can be rewritten in terms of the backshift operator $B^k z_t = z_{t-k}$:

$$(A.2) \quad S_p(\phi,B)z = S_q(\theta,B)e$$

where $S_n(y,w) = 1 + y_1 w + y_2 w^2 + \ldots + y_n w^n = 1 + \sum_{i=1}^{n} y_i w^i$.
In order that the parameters (ϕ,θ,σ^2) of this process be identified (see Appendix F) the parameter vectors ϕ and θ are restricted respectively to the regions C_p and C_q where

$$(A.3) \quad C_n \equiv \{y: \text{ all the roots of } S_n(y,w) = 0 \text{ exceed one in absolute value}\}$$

Hence the parameter space for (ϕ,θ,σ^2) is $C_p \times C_q \times R_+^1$. Note that $C_1 = (-1, +1)$ and C_2 is the interior of the triangle with vertices $(\pm 2, +1)$ and $(0, -1)$.

Tests for determining whether a point $y \in R^n$ is in C_n were first put forth in the time series literature by Wise [19]. Using a formulation in terms of Schur polynomials, Pagano [15] expanded this work and gave a general algorithm for testing. A similar result was obtained by Anderson [2] . All of these tests avoid the computationally prohibitive task of computing the roots of $S_n(y,w) = 0$.

However, when n exceeds 3, C_n becomes quite unwieldy and the Pagano-Anderson-Wise (PAW) tests are not easily used in applications. This is discussed further in Section D. In this paper, a transformation G is presented which maps onto C_n. The analysis of G, especially its use in integrating over C_n is discussed in the next section. The algorithm TRACN derived from G is described in Section C. The motivation and applications of TRACN are discussed in Section D. Appendix F gives a short discussion of identifiability in ARMA time series models. Listings of programs and a test program are given in Appendix G.

B. THEORETICAL FOUNDATION

The transformation G is based on the well known theorem that any polynomial with real coefficients can be uniquely factored into linear factors and real quadratic factors with a negative discriminant (MacLane and Birkoff 13, p. 185). Then any point $y \in C_{2k}$ can be obtained from a point $x \in A_{2k}$ where

(B.1) $$A_{2k} = C_2 \times C_2 \times \ldots \times C_2 \quad \text{(K times)}$$

by equating the coefficients of the corresponding polynomials (this defines G):

(B.2) $$S_{2k}(y,w) = \prod_{i=1}^{k} (1 + x_{2i-1}\, w + x_{2i}\, w^2) \ .$$

Likewise for odd p or q, every $y \in C_{2k+1}$ corresponds to at least one point $x \in A_{2k+1} = C_2 \times C_2 \times \ldots \times C_2 \times C_1$. Correspondingly, it is apparent that $G(A_n) \subseteq C_n$, since every point of A(n) produces a polynomial all whose roots lie outside the unit circle; hence $G(A_n) = C_n$. Let $u_{i0} \equiv 1$, $u_{i1} = x_{2i-1}$, $u_{i2} = x_{2i}$, $i = 1,\ldots,\ell$ where $\ell = [(N+1)/2]$, and let $u_{\ell 2} = 0$ if $2\ell >$ n (odd n), then the mapping $G: A_n \to C_n$ can be defined more precisely by $y = G(x)$ where

(B.3) $$y_r = \sum_{j_1,\ldots j_\ell} \left(\prod_{i=1}^{\ell} u_{i,j_i} \right) \cdot I\left(\sum_{i=1}^{\ell} j_i = r \right)$$

where $i = 1,\ldots, \ell$; $j_i = 0,1,2,$ and I is an indicator function. Also (B.2) can be rewritten more generally ($x_i = 0$ if $i > n$):

(B.2*) $$S_n(y,w) = \prod_{i=1}^{\ell} (1 + x_{2i-1}\, w + x_{2i}\, w^2) \ .$$

To illustrate, let $n = 4$; (B.2*) then becomes

$$(B.4) \qquad (1+y_1w+y_2w^2+y_3w^3+y_4w^4) = (1+x_1w+x_2w^2)(1+x_3w+x_4w^2)$$

and

$$(B.5) \qquad G(x_1,x_2,x_3,x_4) = (x_1+x_3, x_2+x_4+x_1x_3, x_1x_4+x_2x_3, x_2x_4) \ .$$

Thus one problem, traversing all of C_n, is solved by traversing all of A_n, a much simpler task, and using the mapping G. The other half of the problem, integration over C_n, is much more difficult, *because G is not a one-to-one mapping* and the standard result (see e.g., Buck [6], Theorem 4, p. 304 or Fleming [10], Theorem 17, p. 175)

$$(B.6) \qquad \int_{A_n} g(G(x))\,|J_G(x)|\,dx = \int_{C_n} g(y)dy$$

does not apply. $J_G(x)$ is the determinant of Jacobian matrix of the transformation G at $\underline{x}$:

$$(B.7) \qquad J_G(x) \equiv \det\left(\frac{\partial G}{\partial x^T}\right) = \det\left(\left(\frac{\partial G_r(v)}{\partial v_s}\right)\Big|_{v=x}\right)$$

THEOREM. Let $N_G(y)$ be the number[1] of points in $G^{-1}(\{y\})$, then for any finite valued Lebesgue measurable function $g: C_n \to R^1$,

$$(B.8) \qquad \int_{A_n} g(G(x))\,|J_G(x)|\,dx = \int_{C_n} g(y)N_G(y)dy \ .$$

---

[1] If the integration is to be taken over a (Lebesgue measurable) subset B of A_n, then $N_G(y)$ must be adjusted to count only points in $B \cap G^{-1}(\{y\})$.

PROOF. This is an application of Theorem 2.vi, p. 374 of Rado and Reichelderfer [16]. The condition that G be generalized Lipschitzian follows from Lemma 5, p. 375, and Lemma 1, p. 335, which requires each coordinate function of G to be Lipschitzian. This last condition is satisfied since each coordinate function is a finite sum of multinomials of bounded variables.

The remaining problem is to compute $N_G(G(x))$, the number of points in A_n which yield the same polynomial, $S_n(G(x),w)$. Different points can be obtained either by permutating the index of the quadratic factors,

$$(B.9) \qquad \prod_{i=1}^{k} (1+x_{2i-1}w+x_{2i}w^2) = \prod_{i=1}^{k} (1+x_{2\pi(i)-1}w+x_{2\pi(i)}w^2)$$

where π is a permutation of $i = 1,\ldots,k = [N/2]$, or by rearranging the pairing of real roots[2]

$$(B.10) \qquad \begin{aligned}(1+aw)(1+bw)(1+cw) &= (1+(a+b)w+abw^2)(1+cw) \\ &= (1+(a+c)w+acw^2)(1+bw) \\ &= (1+(b+c)w+bcw^2)(1+aw) \ .\end{aligned}$$

Several examples are given in Table 1. Notice that since $S_n(y,w)$ is real, all of the complex roots must be paired with conjugates together.

Given that $S_n(G(x),w) = 0$ has j pairs of complex roots, a counting argument[3] leads to

---

[2]Polynomials without distinct roots have Lebesgue measure zero and are ignored here.

[3]There are $k!$ permutations of the quadratic factors as in (B.9). There are $(n-2j)!/[2^{k-j}(k-j)!]$ ways of making $k - j$ pairs from $n - 2j$ real roots.

$$N_G(G(x)) = M_n(j) = \frac{k!\ (n-2j)!}{2^{k-j}(k-j)!} \tag{B.11}$$

Table 3 lists $M_n(j)$ for $n = 1,\ldots,10$. The number of pairs of complex roots can be found by counting the number of times the discriminant $(x_{2i-1}^2 - 4x_{2i})$ is negative, $i = 1,\ldots,k = [N/2]$.

Table 1. Examples of $G^{-1}(\{\underline{y}\})$

Case 1: $n = 3$, $S_n(y,w) = (1 + 5/6 + 3/8\ w^2 + 1/2\ w^3)$

Roots are $w = -2, -3, -4$. $M_3(0) = 3$

$$G^{-1}(\{y\}) = \begin{Bmatrix} (5/6,\ 1/6,\ 1/4) \\ (3/4,\ 1/8,\ 1/3) \\ (7/12,\ 1/12,\ 1/2) \end{Bmatrix}$$

Case 2: $n = 3$, $S_n(y,w) = 1 + 1/2\ w + 0w^2 - 1/4\ w^3$

Roots are $w = \pm\ i - 1,2$ $M_3(1) = 1$

$$G^{-1}(\{y\}) = \{(1,\ 1/2,\ -1/2)\}$$

$$S_n(y,w) = (1 + w + 1/2\ w^2)(1 - 1/2\ w)$$

Case 3: $n = 4$, $S_n(y,w) = (1 + 0w + 5/36\ w^2 + 0w^3 + 1/36\ w^4)$

Roots are $w = \pm\ 2i, \pm\ 3$ $M_4(1) = 2$

$$G^{-1}(\{y\}) = \begin{Bmatrix} (0,\ 1/4,\ 0,\ -1/4) \\ (0,\ -1/9,\ 0,\ 1/4) \end{Bmatrix}$$

Case 4: $n = 4$, $S_n(y,w) = (1 + 0w - 13/36\ 2^2 + 0w^3 + 1/36\ w^4)$

Roots are $w = \pm\ 2, \pm\ 3$ $M_4(0) = 6$

$$G^{-1}(\{y\}) = \begin{Bmatrix} (0,\ -1/4,\ 0,\ -1/9) \\ (0,\ -1/9,\ 0,\ -1/4) \\ (-5/6,\ 1/6,\ 5/6,\ 1/6) \\ (-1/6,\ -1/6,\ 1/6,\ -1/6) \\ (1/6,\ -1/6,\ -1/6,\ -1/6) \\ (5/6,\ 1/6,\ -5/6,\ 1/6) \end{Bmatrix}$$

Table 2

$$M_n(j) = \frac{k!(n-2j)!}{2^{k-j}(k-j)!}$$

k	n \ j	0	1	2	3	4	5
0	1	1					
1	2	1	1				
1	3	3	1				
2	4	6	2	2			
2	5	30	6	2			
3	6	90	18	6	6		
3	7	630	90	18	6		
4	8	2520	360	72	24	24	
4	9	22680	2520	360	72	24	
5	10	113400	12600	1800	360	120	120

$k = [n/2]$

C. ALGORITHM TRACN

Algorithm TRACN takes a point x from A_n and renders $y = G(x) \in C_n$, $N_G(G^{-1}(\underline{x}))$, and $|J_G(x)|$. The algorithm is described in pseudo-Algol in Table 3 and a FORTRAN listing is given in Appendix G. There are basically three steps to the operation.

1) Obtain x's and count the pairs of complex roots,

2) Compute $G(x)$,

3) Compute $J_G(x)$.

The first part is rather straightforward. In the FORTRAN subroutine TRACN, external references are made to RUIR1 and RUIR2 which furnish points from C_1 and C_2, respectively. The problem of computing $G(x)$ is more formidable. In order to multiply the ℓ quadratic polynomials together, a nesting of loops for indices $j_1,\ldots,j_\ell$ was simulated and the formulation of x in terms of u_{ij} and (B.3) was used. Every combination of 0,1,2 in j_1 through j_ℓ was obtained, r calculated and y incremented by $p = \Pi u_{ij_i}$. The values for $r = 0$ were computed and forgotten.

The final part of the problem, that of computing $J_G(x)$, now follows simply. A matrix is formed to hold $(\partial y/\partial u)$ from which the relevant columns are chosen for $(\partial y/\partial x)$. Now

$$\frac{\partial y_r}{\partial u_{i^*,j_{i^*}}} = \sum_{j_i,\ldots,j_\ell} \left(\prod_{\substack{i=1 \\ i\neq i^*}}^{\ell} u_{i,j_i} \right) \cdot I\left(\sum_{i=1}^{\ell} j_i = r \right)$$

hence the desired increment to $(\partial y_r/\partial u)$ is $p/u_{i^*j_{i^*}}$. Once the nested looping ceases and all combinations 0,1,2 appear for the indices $j_1,\ldots,j_\ell$, $(\partial y/\partial x)$ can be found by deleting the columns

corresponding to u_{io} and, if $2\ell > n$, $u_{\ell 2}$. In the subroutine TRACN, $(\partial y/\partial u)$ is stored transposed in PGPX; the determinant of the Jacobian matrix $(\partial y/\partial x)$, placed in array A, is computed by GESEPP, a general linear equations solver.

Table 3: Algorithm TRACN

1. $\ell \leftarrow [(n+1)/2]$; $j \leftarrow 0$;
2. Initialize GX and PGPX to 0;
3. (Obtain x's and count pairs of complex roots)

   for $i = 1,\ldots,\ell$ do

   begin $u_{i0} = 1$;

   if $(2*i \leq n)$ then (choose (u_{i1},u_{i2}) from C_2)

   else begin $u_{\ell 2} \leftarrow 0$;

   (choose u_{i1} from C_1);

   end;

   if $(u_{i1}^2 > 4*u_{i2})$ then (increment j by 1);

   end;
4. (Find G and $(\partial y/\partial u)$ = PGPX)

   for $i = 1,\ldots,\ell$ do

   for index (i) = 0,1,2 do

   begin

   Compute $r = \Sigma$ index (i);

   Compute $p = \Pi u_{i,\text{index (i)}}$;

   GX(r) = GX(r) + p;

   Comment $(\partial y/\partial u) = p/u_{i,\text{index (i)}}$;

   PGPX(i, index (i),r) = PGPX (i, index (i),r) + $p/u_{i,\text{index (i)}}$

   end;
5. Load A = jacobian matrix with relevant parts of PGPX;
6. Compute the determinant of A and return $|\det A|$;

D. MOTIVATION AND APPLICATIONS

To explain the value of TRACN, consider that two statistical methodologies can be applied to the majority of nonstandard statistical problems, maximum likelihood and Bayesian methods. In maximum likelihood (see, e.g., Theil [18]), the likelihood function, here a function of ϕ and θ for a given set of observations, must be maximized[4] over the parameter space $C_p \times C_q$. In most situations, this maximization must be done numerically. In a Bayesian analysis[5], since analytic expressions for the posterior distributions do not exist, it is necessary to integrate numerically over $C_p \times C_q$.

Consider now how the PAW tests apply to these two computational problems, beginning with integration over $C_p \times C_q$.

Either a mechanical quadrature procudt rule [17] or a Monte Carlo method [11] must be employed to produce abscissas over a simple, usually rectangular, region containing C_n. An indicator function, based on the PAW tests and vanishing if the abscissa is outside C_n, must be computed for each point. Both methods of numerical integration will be inefficient since C_n does not fit snugly inside an n-dimensional box. The poor fit is displayed in Table 4, giving estimates of the volumes of C_n and an enclosing box.

For the problem of optimizing over $C_p \times C_q$, the tests, being a "black-box" type of function do not help in locating the boundary of the

---

[4] The optimizing σ^2 can be found analytically as a (complicated) function of (ϕ,θ).

[5] For the framework of Bayesian inference, see DeGroot [8] or Zellner [20]. The Bayesian analysis of ARMA time series models is the object of the author's current research [14].

constrained region. Moreover, the penalty method [1,12] of constrained optimization is not easily applied here. The Wise and Anderson results are just checks. Pagano's method of checking whether the Schur matrix (a function of y) is positive definite shows some hope but it is unclear how to measure how close to "not-positive-definite" the matrix is and hence how close y is to the boundary.

Algorithm TRACN cannot be used directly in maximum likelihood with any standard optimization method based on local properties since the mapping $G: A_n \to C_n$ is not one-to-one. However, neither local optima of the likelihood function nor other aberrant behavior can precluded from the problem of maximum likelihood, especially for large p or q. TRACN can then be useful in the problem of global optimization in two ways. First, a common method in global problems [9] is to employ a locally-based search method from several randomly chosen starting points, which TRACN can insure to be in C_n. More importantly current practice should be improved greatly by using TRACN to do a preliminary pure random search [4]. Since near its global maximum the likelihood function is nearly always well behaved, locally-based optimization algorithms armed with a coarse but reliable starting point should then quickly succeed.

Given the irregular shape of C_n as indicated in Table 4, TRACN provides the only practical hope for integrating over C_n. TRACN is best suited for Monte Carlo integration, which for n greater than 3, is the only logical approach [7]. The use of mechanical quadrature requires some warnings, however. First, a different set of abscissas must be used for each triangle C_2 making up A_n. If identical points are sampled simultaneously, $(\partial G(x)/\partial x)$ will be singular and the algorithm

will fail. Secondly, no sampled x should take the exact value 0, since the algorithm will treat this as the odd case $2\ell > n$ and J_G will be incorrect[6] ($G(x)$ will be fine). The easy way out is just to add 10^{-6} (or smaller) to all of the points. Finally, the potential user should be warned that $N_G(G(x))$ is a rather discontinuous function of x, hence the order of convergence for fixed integration rules will seldom hold up. Note that this is not a concern in Monte Carlo integration.

---

[6]There appears to be no efficient way to circumvent this.

Table 4

Approximate Volumes of C_n and Enclosing Box

n	Volume of C_n	Volume of Box	Ratio
3	5.3*	3.2×10^{1}	.17
4	7.1	3.2×10^{2}	.022
5	7.7	1.9×10^{3}	.004
6	8.3	2.6×10^{5}	3×10^{-5}
7	7.0	6.4×10^{5}	1×10^{-5}
8	8.0	1.0×10^{7}	8×10^{-7}

*Exact value is 16/3.

E. REFERENCES

[1] Adby, P. R. and M. A. H. Dempster (1974). *Introduction to Optimization Methods*, Wiley, New York.

[2] Anderson, O. D. (1975). "The Recursive Nature of the Stationary and Invertibility Constraints on the Parameters of Mixed Auto-regressive-Moving Average Processes," *Biometrika 62*, pp. 704-706.

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

[4] Anderssen, R. S. (1972). "Global Optimization," in *Optimization*, ed. R. S. Anderssen, et al., U. of Queensland Press, pp. 26-48.

[5] Box, G. E. P. and G. M. Jenkins (1976). *Time Series Analysis: Forecasting and Control* (Rev. ed.), Holden Day, San Francisco.

[6] Buck, R. C. (1965). *Advanced Calculus* (2nd. ed.), McGraw-Hill, New York.

[7] Davis, P. J. and P. Rabinowitz (1975). *Numerical Integration*, Academic Press, New York.

[8] DeGroot, M. H. (1970). *Optimal Statistical Decisions*, McGraw-Hill, New York.

[9] Dixon, L. C. W. and G. P. Szego, eds. (1975). *Towards Global Optimization*, North Holland, Amsterdam.

[10] Fleming, W. H. (1965). *Functions of Several Variables*, Addison-Wesley, Reading, Mass.

[11] Hammersley, J. M. and D. C. Handscomb (1964). *Monte Carlo Methods*, Methuen, London.

[12] Luenberger, D. G. (1969). *Optimization by Vector Space Methods*, Wiley, New York.

[13] MacLane, S. and G. Birkhoff (1967). *Algebra*, MacMillan, New York.

[14] Monahan, J. F. (1980). "A Structured Bayesian Approach to ARMA Time Series Models," Institute of Statistics Mimeo Series No. 1297, N. Carolina State Univ.

[15] Pagano, M. (1973). "When is an Autoregressive Scheme Stationary?" *Comm. Stat. 1*, pp. 533-544.

[16] Rado, T. and P. V. Reichelderfer (1955). *Continuous Transformations in Analysis*, Springer-Verlag, Berlin.

[17] Stroud, A. H. (1971). *Approximate Calculation of Multiple Integrals*, Prentice-Hall, Englewood Cliffs, N.J.

[18] Theil, H. (1971). *Principles of Econometrics*, Wiley, New York.

[19] Wise, J. (1956). "Stationary Conditions for Stochastic Processes of the Autoregressive and Moving Average Type," *Biometrika 43*, pp. 215-219.

[20] Zellner, A. (1971). *An Introduction to Bayesian Inference in Econometrics*, Wiley, New York.

APPENDIX F. IDENTIFIABILITY

Problems concerning identifiability* arise in one of two ways: parameter redundancy or insufficient observations to distinguish between models. Only the first will be considered here and the number of observations of the stochastic process $\{z_t\}$ of (A.1) is considered infinite.

Given the observation vector x arising from a family of distributions $P(x|\psi)$, $\psi \in \Psi$, ψ is *identified* iff

$$P(x|\psi_1) = P(x|\psi_2) \text{ implies } \psi_1 = \psi_2 . \tag{F.1}$$

For a stationary Gaussian time series model, $P(x|\psi_1) = P(x|\psi_2)$ is equivalent to $f(\lambda|\psi_1) = f(\lambda|\psi_2)$ where $f(\lambda|\psi)$ is the spectral density of the process [3, especially Ch. 7], which for an ARMA process is

$$f(\lambda|\phi,\theta,\sigma^2) = \frac{\sigma^2 \left| \sum_{r=1}^{q} \theta_r e^{i\lambda r} \right|^2}{2\pi \left| \sum_{s=1} \phi_s e^{i\lambda s} \right|^2} \tag{F.2}$$

Parameter redundancies arise in the following manner. Let v_i, $i=1,\ldots,q$ be the roots of $S_q(\theta,w) = 0$, hence

$$\left| \sum_{r=1}^{q} \theta_r e^{i\lambda r} \right| = \prod_{r=1}^{q} |e^{i\lambda} - v_r| . \tag{F.3}$$

---

*Box and Jenkins [5] use the term "model multiplicity" to indicate lack of identification.

But notice that $|e^{i\lambda} - v_r| = |v_r| \cdot |e^{i\lambda} - 1/\bar{v}_r|$, hence if v_r is real, $f(\lambda|\phi,\theta,\sigma^2) = f(\lambda|\phi,\theta^*,\sigma^2 v_{r^*}^2)$ where

$$S_q(\theta^*,w) = \prod_{r \neq r^*} (w - v_r)\cdot(w - 1/v_{r^*}) \tag{F.4}$$

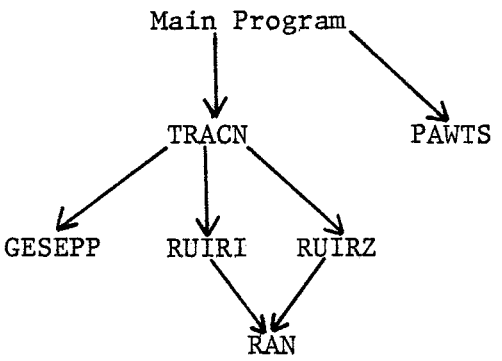
and $S_q(\theta^*,w)$ will be a polynomial with real coefficients. Hence, the parameter points (ϕ,θ,σ^2) and $(\phi,\theta^*,\sigma^2 v_{r^*}^2)$ are different but yield the same distribtion and (ϕ,θ,σ^2) is not identified. Notice any real root can be selected, perhaps many at a time, so that the resultant polynomial $S_q(\theta^*,w)$ has some of its roots flipped. Complex roots can be flipped also if chosen in conjugate pairs. Notice that the same changes can be done to the denominator of $f(\lambda)$, the autoregressive parameters. However, restricting ϕ to C_p and θ to C_q make these reparameterizations invalid, since the flipping would then produce a root *inside* the unit circle. Hence (ϕ,θ,σ^2) will be identified** if the parameter space Ψ is restricted to $C_p \times C_q \times R_+^1$.

**An additional constraint must be that $S_p(\phi,w)$ and $S_q(\theta,w)$ have no root in common.

APPENDIX G

The following are programs to test TRACN.

The Structure:



Note: These programs were implemented in double precision mainly to aid the performance of PAWTS. To change to single precision, change all of the REAL*8 references to REAL, change DABS to ABS (three times), and delete DBLE from RUIRZ.

```
00010 C  USE PAWTS TO CHECK IF TRACN GIVES ALL ROOTS OUTSIDE UNIT CIRCLE
00020       REAL*8 VEC(10),JACOB
00030       LOGICAL OUTSID,PAWTS
00040       JACOB=RAN(7372585)
00050       DO 1N=1,10
00060       DO 2I=1,3
00070       CALL TRACN(N,VEC,JACOB,M)
00080       OUTSID=PAWTS(VEC,N)
00090       WRITE(3,22) N,I,OUTSID,(VEC(J),J=1,N)
00100    2  CONTINUE
00110    1  CONTINUE
00120    22 FORMAT(1X,'N=',I3,I3,'  OUTSIDE? ',L1,'  VEC=',5F10.6/27X,5F10.6)
00130       STOP
00140       END
00150       LOGICAL FUNCTION PAWTS(V,N)
00160       REAL*8 V(10),ALPHA(11),A(10,10),T,S
00170       INTEGER N,IDET,I,J,K,IM1,JM1
00180 C  TEST TO SEE IF THE POLYNOMIAL Z**N+V(1)*Z**(N-1)+...+V(N-1)*Z+V(N)=0
00190 C  HAS ALL OF ITS ROOTS INSIDE THE UNIT CIRCLE
00200 C  PAGANO, COMM. STAT. V1 NO 6 (1973), PP.533-544.
00210 C  FIRST SET UP ALPHA'S 1 TO N+1  INSTEAD OF 0 TO N  (DAMN FORTRAN!)
00220       NP1=N+1
00230       ALPHA(1)=1.
00240       DO 2I=1,N
00250    2  ALPHA(I+1)=V(I)
00260 C  SET UP SCHUR'S MATRIX....BRUTE STRENGTH, SEE P.537
00270       DO 3J=1,N
00280       DO 3K=J,N
00290       A(J,K)=0.
00300       DO 4L=1,J
00310    4  A(J,K)=A(J,K)+ALPHA(J-L+1)*ALPHA(K-L+1)
00320      1-ALPHA(NP1+L-J)*ALPHA(NP1+L-K)
00330    3  A(K,J)=A(J,K)
00340 C  NOW CHECK TO SEE IF SCHUR'S MATRIX A IS POSITIVE DEFINTIE
00350 C  USING MODIFIED CHOLESKY FACTORIZATION  A=L*D*L-TRANSPOSE
00360       PAWTS=.FALSE.
00370       DO 6I=1,N
00380       T=A(I,I)
00390       IF(I.EQ.1) GO TO 5
00400       IM1=I-1
00410 C  SOLVE FIRST L(N-1)*X=AS(N)
00420       DO 7J=1,IM1
00430       S=A(I,J)
00440       IF(J.EQ.1) GO TO 7
00450       JM1=J-1
00460       DO 8K=1,JM1
00470    8  S=S-A(J,K)*A(I,K)
00480    7  A(I,J)=S
00490 C SOLVE NOW D(N-1)*LS(N)=X, GET D(N)
00500       DO 9J=1,IM1
00510       S=A(I,J)
00520       A(I,J)=A(I,J)/A(J,J)
00530    9  T=T-S*A(I,J)
00540    5  A(I,I)=T
00550       IF(T.LT.-1.D-6) RETURN
00560       IF(T.LE.0..AND.I.NE.N) RETURN
00570    6  CONTINUE
00580       PAWTS=.TRUE.
00590       RETURN
00600       END
```

```
00610       SUBROUTINE TRACN(N,VEC,JACOB,MNOJ)
00620 C  TRACN GENERATES THE RANDOM COEFFICIENTS OF POLYNOMIALS OF DEGREE N
00630 C  (N LE 10) ALL WHOSE ROOTS ARE GREATER THAN ONE IN ABSOLUTE VALUE
00640 C  TO INTEGRATE OVER THIS SPACE OF COEFFICIENTS, JACOB AND MNOJ NEEDED
00650 C  JACOB= DETERMINANT OF JACOBIAN OF TRANSFORMATION
00660 C  MNOJ= LOCAL MULTIPLICITY OF TRANSFORMATION TO THIS SPACE FROM THE
00670 C        ORIGINAL CARTESIAN PRODUCT OF TRIANGLES AND (ODD ONLY) INTERVAL
00680 C  IF FIXED INTEGRATION RULE IS PREFERRED, REPLACE RUIR2 AND RUIR1 WITH
00690 C  APPROPRIATE SUBPROGRAMS WITH EXTRANEOUS ACCESS TO WEIGTHS
00700 C  RUIR2 TAKES POINTS UNIFORMLY FROM TRIANGLE...WITH AREA 4
00710 C  RUIR1 TAKES POINTS UNIFORMLY FROM INTERVAL...WITH LENGTH 2
00720       REAL*8 VEC(10),JACOB,XX(5,3),GX(11),PGPX(15,11),A(10,10),X,Y,PROD
00730       INTEGER INDEX(5),MNOJP1(6,10),PT
00740       DATA MNOJP1/1,      0,      0,      0,      0,      0,
00750      2            1,      1,      0,      0,      0,      0,
00760      3            3,      1,      0,      0,      0,      0,
00770      4            6,      2,      2,      0,      0,      0,
00780      5           30,      6,      2,      0,      0,      0,
00790      6           90,     18,      6,      6,      0,      0,
00800      7          630,     90,     18,      6,      0,      0,
00810      8         2520,    360,     72,     24,     24,      0,
00820      9        22680,   2520,    360,     72,     24,      0,
00830      *       113400,  12600,   1800,    360,    120,    120/
00840 C  GET L FIRST
00850       L=(N+1)/2
00860 C  INITIALIZE
00870       J=0
00880       DO 19I=1,11
00890       GX(I)=0.
00900       DO 19K=1,15
00910    19 PGPX(K,I)=0.
00920 C GET RANDOM X'S
00930       DO 10I=1,L
00940       XX(I,1)=1.
00950       IF(2*I.LE.N) GO TO 9
00960 C TAKE CARE ODD N
00970       CALL RUIR1(X)
00980       Y=0.
00990       GO TO 8
01000    9  CALL RUIR2(X,Y)
01010    8  XX(I,2)=X
01020       XX(I,3)=Y
01030 C  CHECK IF COMPLEX PAIR HERE...INCREMENT J IF SO
01040       IF(X*X-4.*Y.LT.0.) J=J+1
01050    10 CONTINUE
01060 C  MNOJ=K!(N-2J)!/((K-J)!2**(K-J))  ...K=N/2
01070 C  J= NO OF PAIRS OF COMPLEX ROOTS
01080       MNOJ=MNOJP1(J+1,N)
```

```
01090 C  NOW TRANSFORM
01100 C  SIMULATE LOOPS TO GET ALL PRODUCTS
01110        DO 1I=1,L
01120    1   INDEX(I)=0
01130    2   PT=1
01140        INDEX(PT)=0
01150    3   CONTINUE
01160 C  PROCESS EACH PRODUCT: XX(1,?)*XX(2,?)*...*XX(L,?)
01170        PROD=1.0
01180        IEXPP1=1
01190        DO 11I=1,L
01200        INDY=INDEX(I)
01210        IEXPP1=IEXPP1+INDY
01220   11   PROD=PROD*XX(I,INDY+1)
01230 C  STASH PRODUCT OF TRANSFORMATION
01240        GX(IEXPP1)=GX(IEXPP1)+PROD
01250 C  NOW TAKE CARE OF MATRIX FOR JACOBIAN
01260        DO 12I=1,L
01270        INDYP1=INDEX(I)+1
01280        IF(XX(I,INDYP1).EQ.0.) GO TO 12
01290        IPT=3*(I-1)+INDYP1
01300        PGPX(IPT,IEXPP1)=PGPX(IPT,IEXPP1)+PROD/XX(I,INDYP1)
01310   12   CONTINUE
01320 C  FINISH SIMULATED LOOPING
01330    4   INDEX(PT)=INDEX(PT)+1
01340        IF(INDEX(PT).NE.3) GO TO (3,2,2,2,2),PT
01350        INDEX(PT)=0
01360        PT=PT+1
01370        IF(PT.LE.L) GO TO 4
01380 C  NOW GET WANTED ELEMENTS FOR JACOBIAN
01390        DO 13I=1,N
01400        VEC(I)=0.
01410        DO 13K=1,N
01420   13   A(I,K)=PGPX((3*I+1)/2,K+1)
01430 C  GESEPP IS GENERAL SIMULTANEOUS EQUATIONS SOLVER
01440        CALL GESEPP(A,N,JACOB,VEC)
01450 C  JACOB NOW HOLDS THE DETERMINANT OF THE JACOBIAN MATRIX
01460 C  DON'T FORGET TO TAKE THE ABSOLUTE VALUE
01470        JACOB=DABS(JACOB)
01480 C  NOW LOAD VEC WITH COEFFICIENTS
01490 C  POLY IS 1+VEC(1)*W+VEC(2)*W**2+...+VEC(N)*W**N
01500        DO 14I=1,N
01510   14   VEC(I)=GX(I+1)
01520        RETURN
01530        END
```

```
01540       SUBROUTINE RUIR1(X)
01550 C  RETURNED X IS UNIFORMLY DISTRIBUTED ON (-1,+1)
01560 C  RAN IS A UNIFORM(0,1) PSEUDORANDOM NUMBER GENERATOR
01570       REAL*8 X,S
01580       DATA S/-1./
01590       X=2.*RAN(1)
01600       IF(X.GT.1.) X=S*(X-1.)
01610       RETURN
01620       END
01630       SUBROUTINE RUIR2(X,Y)
01640 C RETURNED X,Y IS UNIFORMLY DISTRIBUTED ON TRIANGLE (0,-1),(-2,1),(2,1)
01650 C  RAN IS A UNIFORM(0,1) PSEUDORANDOM NUMBER GENERATOR
01660       REAL*8 X,Y,R,S,H
01670       R=DBLE(RAN(1))
01680       S=DBLE(RAN(2))
01690       IF(R.GT.S) GO TO 2
01700       H=R
01710       R=S
01720       S=H
01730    2  X=2.0*(R+S-1.)
01740       Y=2.*(S-R)+1.
01750       RETURN
01760       END
01770       REAL FUNCTION RAN(IXX)
01780 C  UNIFORM PSEUDORANDOM NUMBER GENERATOR
01790 C  FORTRAN VERSION OF LEWIS, GOODMAN, MILLER
01800 C  SCHRAGE,  ACM TOMS V.5 (1979) P132
01810 C  FIRST CALL SETS SEED TO IXX, LATER IXX IGNORED
01820       INTEGER A,P,IX,B15,B16,XHI,XALO,LEFTLO,FHI,K
01830       DATA A/16807/,B15/32768/,B16/65536/,P/2147483647/
01840       DATA IX/0/
01850       IF(IX.EQ.0) IX=IXX
01860       XHI=IX/B16
01870       XALO=(IX-XHI*B16)*A
01880       LEFTLO=XALO/B16
01890       FHI=XHI*A+LEFTLO
01900       K=FHI/B15
01910       IX=(((XALO-LEFTLO*B16)-P)+(FHI-K*B15)*B16)+K
01920       IF(IX.LT.0) IX=IX+P
01930       RAN=FLOAT(IX)*4.656612875E-10
01940       RETURN
01950       END
```

```
01960       SUBROUTINE GESEPP(A,N,DET,B)
01970 C GAUSSIAN ELIMINATION WITH PARTIAL PIVOTING
01980 C  SOLVES A*X=B,  A IS DESTROYED,   B IS OVERWRITTEN WITH SOLUTION X
01990 C  A IS N BY N, B IS N BY 1, 1 LE N LE 10, DIMENSION 10 IS ARBITRARY
02000 C  DET= DETERMINANT OF A
02010 C   J F MONAHAN     CURRENT VERSION JULY 1980
02020       REAL*8 A(10,10),B(10),DET,S,T
02030       DET=1.
02040       NM1=N-1
02050       IF(N.EQ.1) GO TO 7
02060       DO 6I=1,NM1
02070       IP1=I+1
02080       S=0.
02090       DO 1J=I,N
02100 C  LOOK FOR GOOD PIVOT IN I-TH COLUMN
02110       IF(DABS(A(J,I)).LE.S) GO TO 3
02120       L=J
02130       S=DABS(A(J,I))
02140   1   CONTINUE
02150 C   DON'T SWITCH IF AVOIDABLE
02160       IF(I.EQ.L) GO TO 6
02170       DO 2K=I,N
02180       T=A(I,K)
02190       A(I,K)=A(L,K)
02200   2   A(L,K)=T
02210       T=B(I)
02220       B(I)=B(L)
02230       B(L)=T
02240       DET=-DET
02250 C   ELIMINATE
02260   3   DO 5J=IP1,N
02270       S=-A(J,I)/A(I,I)
02280       DO 4K=IP1,N
02290   4   A(J,K)=A(J,K)+S*A(I,K)
02300   5   B(J)=B(J)+S*B(I)
02310   6   DET=DET*A(I,I)
02320   7   DET=DET*A(N,N)
02330 C  ELIMINATION DONE----NOW BACKSOLVE
02340       B(N)=B(N)/A(N,N)
02350       IF(N.EQ.1) RETURN
02360       DO 9J=1,NM1
02370       I=N-J
02380       IP1=I+1
02390       DO 8K=IP1,N
02400   8   B(I)=B(I)-A(I,K)*B(K)
02410   9   B(I)=B(I)/A(I,I)
02420       RETURN
02430       END
```

```
watfiv tgrip5.fort
WATFIV COMPILATION
COMPILATION COMPLETE
N=  1  1  OUTSIDE? T  VEC= -0.047594
N=  1  2  OUTSIDE? T  VEC=  0.928670
N=  1  3  OUTSIDE? T  VEC=  0.165734
N=  2  1  OUTSIDE? T  VEC= -0.088458 -0.060604
N=  2  2  OUTSIDE? T  VEC=  0.520892  0.206442
N=  2  3  OUTSIDE? T  VEC=  0.632889  0.336010
N=  3  1  OUTSIDE? T  VEC= -0.257669  0.368785 -0.110866
N=  3  2  OUTSIDE? T  VEC=  0.144436  0.032676 -0.100029
N=  3  3  OUTSIDE? T  VEC=  1.779031  1.088213  0.286036
N=  4  1  OUTSIDE? T  VEC= -1.058543  0.316726  0.329597 -0.342188
N=  4  2  OUTSIDE? T  VEC= -0.710463 -0.278300  0.803767 -0.348730
N=  4  3  OUTSIDE? T  VEC= -0.580424  0.187076 -0.060817  0.003364
N=  5  1  OUTSIDE? T  VEC= -1.940971  1.198988  0.265220 -0.762863  0.245583

N=  5  2  OUTSIDE? T  VEC=  0.335461  1.677051  0.563330  0.874807  0.326085

N=  5  3  OUTSIDE? T  VEC= -1.234970  0.117741  0.561794 -0.594805  0.229232

N=  6  1  OUTSIDE? T  VEC= -0.612452  0.349930 -0.159932 -0.287869  0.253802
                           -0.278649
N=  6  2  OUTSIDE? T  VEC= -1.220780  1.119371 -0.983899  0.468332  0.006973
                            0.013335
N=  6  3  OUTSIDE? T  VEC=  0.545736  0.102358 -0.138618 -0.368020 -0.715616
                           -0.181465
N=  7  1  OUTSIDE? T  VEC= -0.765826 -0.574328  0.711938 -0.675878  0.059140
                            0.318127 -0.070376
N=  7  2  OUTSIDE? T  VEC=  1.318575  1.056819  0.261822 -0.193827 -0.056290
                           -0.100900 -0.005162
N=  7  3  OUTSIDE? T  VEC=  0.673042  0.052813 -0.018260 -0.462352 -0.072992
                            0.099433  0.009272
N=  8  1  OUTSIDE? T  VEC= -0.636126  0.348677 -0.775713  0.844114 -0.108362
                            0.044784 -0.013545  0.007095
N=  8  2  OUTSIDE? T  VEC=  0.279849 -0.724300  0.226236  0.170171  0.192251
                            0.045881 -0.238824 -0.027930
N=  8  3  OUTSIDE? T  VEC=  1.105256  1.862984  1.057405  0.743716  0.115142
                            0.072957 -0.029502  0.023494
N=  9  1  OUTSIDE? T  VEC= -0.744113  1.089365 -1.367273  0.858330 -0.543865
                            0.341173 -0.173896  0.055020 -0.006690
N=  9  2  OUTSIDE? T  VEC=  1.169027 -0.592276 -0.815448 -0.079767 -0.033202
                           -0.001561  0.003932 -0.000490  0.000016
N=  9  3  OUTSIDE? T  VEC= -1.581407  0.215439  0.072616  0.904987 -0.598849
                            0.008024 -0.113739  0.113855  0.001056
N= 10  1  OUTSIDE? T  VEC=  0.994891  0.940317  1.326104  1.240242  0.702571
                            0.473775  0.474974  0.106081  0.060867  0.034185
N= 10  2  OUTSIDE? T  VEC= -0.285989  1.384587 -0.118786  0.464443 -0.313202
                            0.142303 -0.594556  0.334711 -0.275455  0.121123
N= 10  3  OUTSIDE? T  VEC=  1.734054  1.844204  1.743714  1.065129  0.349692
                            0.101664  0.087318  0.023303  0.006772 -0.000753
STATEMENTS EXECUTED=  130779

READY
logoff
CPU(00:00:04)  EXCP(00:00:03)  IWT(  18,00:00:02)  OWT(  12,00:00:01)
    CONNECT(00:20:35)  MEMORY USED( 200K)  APPROX. COST($1.04)
USER07 LOGGED OFF TSO AT 16:22:16 ON AUGUST 10, 1980+
```