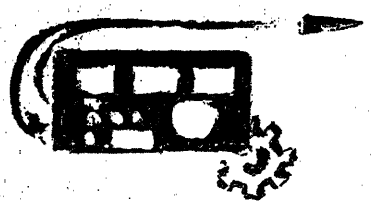

Scientific and Business Applications

D. TEICHROEW, Editor

FORTRAN Subroutines for Time Series Analysis

M. J. R. HEALY* AND B. P. BOGERT
Bell Telephone Laboratories, Inc.
Murray Hill, New Jersey

The authors have recently been concerned in a time-series study that constituted a fairly typical piece of applied statistical research, involving extensive computations on a moderately large quantity of data. We have found that the many different numerical processes that were required could be built up almost completely from a small number of basic operations, and a set of FORTRAN subroutines has been written to perform these. The main purpose of this note is to describe these subroutines, but since the question of general statistical programs is topical [1], we include some general remarks.

The desirability of a "building-block" type of approach, made possible by the FORTRAN subroutine facility, is a matter for argument. This approach amounts to the use of an interpretive programming technique, as opposed to an assembler or compiler. A subroutine call is analogous to a high-level instruction which is decoded into machine language each time it is encountered in a program. The chief advantage of interpreters lies in the fact that they are far simpler to write than compilers. This is overwhelmingly so in the present application in which the vast bulk of the programming work was taken care of by FORTRAN. It is also a matter of great practical importance; the set of subroutines here described evolved as a by-product of an actual statistical investigation in which the time available for programming was severely limited. In the construction of the subroutines and in their subsequent use, we have gotten through a very large amount of computation without imposing a load on scarce programming resources.

Interpreters are usually held to be inefficient because of the individual handling of each high-level instruction and the unproductive time spent in interpretation. The importance of these considerations depends to a large extent upon the size of the building blocks. FORTRAN has a certain capability for tackling a programming problem as a whole rather than piece-meal, and this can be given full play within each subroutine. Equally, the importance of interpretation time depends upon the proportion which this bears to computing time. Both these considerations make it desirable for each subroutine to carry out a fairly substantial piece of computation. The opposite limitation is felt when the subprograms become so complex that they do not readily fit together in different combinations.

The subroutine approach has a further important advantage. The interpretive use of FORTRAN in no way precludes its being used in the normal manner. The scheme is not a closed one and the user is free to specify any operations he pleases in ordinary FORTRAN terms. This in turn means that the scheme itself can be easily added to or modified and that it is quite unnecessary to try to foresee and provide for all future applications, a futile task in a rapidly developing area of research.

So far, we have discussed the set of subroutines effectively as a set of useful abbreviations for frequently required sequences of FORTRAN statements. One further function that they can perform, which has been forcibly brought to our notice, is that of labelling. With modern computers, the amount of paper arriving on the statistician's desk tends to increase rapidly and keeping track of results becomes a major problem. As a partial solution, we have insisted that each data deck or tape should carry with it its format and identification, the latter being carried forward onto any set of derived quantities, whether on paper, cards, tape or microfilm. We have also found it useful for each subroutine, on completing its task, to print out a statement of what it has done with the values of any parameters used. This provides an explanation of the results produced as well as giving a check on the correct loading of the problem and has occasionally detected machine malfunction.

The subroutines described below were not written with the above considerations explicitly in mind—rather, it was their construction for use that caused us to make these considerations explicit to ourselves. Intending users should acquaint themselves with alternative systems, such as the BLODI compiler (Kelly [2]) and the BOMM system, developed by Bullard and his collaborators at LaJolla and Cambridge, of which a description is to be published.

It is clearly desirable to have some uniformity in the calls for the different subroutines. The basic items to be

* On leave from Rothamsted Experimental Station, Harpenden, Herts, England.

handled are sequences of numbers of different lengths. Where possible, the name of a sequence is followed by an integer variable giving its length, and input variables precede output variables. Thus

CALL FILTER (*A*, *NA*, *F*, *NF*, *B*, *NB*)

can be read

To the sequence *A* of length *NA*, apply the filter coefficients *F*, *NF* in number, to produce the sequence *B* of length *NB*.

For explanation of technical terms in the field of time-series analysis, the reader is referred to Blackman & Tukey [3].

The Subroutines

1. READIN (*A*, *NA*, *ANAME*)

Reads in *NA* values of the sequence *A* from punch cards. *NA* is read from columns 1–6 of the first card; the BCD characters in columns 7–80 are stored in the vector *ANAME* for future reference. This vector must be given the dimension 13 in the main program. The second card of the deck carries the format of the data cards in ordinary FORTRAN form, omitting the word *FORMAT* but including the parentheses.

2. OUTPUT (*A*, *NA*, *ANAME*, *INDIC*)

Prints *NA* values of the sequence *A* in floating form (without starting a new page). The printout is preceded by the name held in BCD characters in the vector *ANAME*, which must be given the dimension 13 in the main program. If *INDIC* is negative, the series will be punched out five to a card with the two preceding cards required by READIN; if *INDIC* is a positive number, the series will be written on the corresponding tape as three binary records, the first containing *NA*, the second *ANAME*.

3. TAPEIN (*A*, *NA*, *ANAME*, *NTAPE*)

Performs the same function as READIN, except that the sequence is read from a binary tape of the form written by OUTPUT.

4. DETRND (*A*, *NA*, *B*, *NDEG*)

The mean (*NDEG* = 0) or a least-squares linear trend (*NDEG* = 1) is subtracted from the *NA* values of the sequence *A* and the residuals stored as the sequence *B*.

5. TAPER (*A*, *NA*, *START*, *END*, *B*)

The ends of the sequence *A*, of total length *NA*, and multiplied by "raised cosine tapers" of the form $(1 - \cos k\pi)/2$, $0 \leqq k \leqq 1$, to form the sequence *B*. The extents of the tapers are expressed by *START* and *END* as fractions of the total length of the sequence. Thus, to taper the first 10 percent and the last 20 percent of the sequence *A*, the call would read

CALL TAPER (*A*, *NA*, 0.1, 0.2, *B*)

6. LOGTR (*A*, *NA*, *B*, *AO*, *THETA*)

Forms the sequence $B(I) = \log_{10} A(I)$, $I = 1, NA$. If $A(I) = 0$, $B(I) = THETA * \log A(J)$, where $A(J)$ is the nearest previous positive value of the sequence *A*; if $A(I) < 0$, $B(I) = THETA * \log A(J) + (1.0 - THETA) * \log(-A(I))$. In either case a comment is printed out. If $A(1) \leqq 0$, $B(1) = \log AO$. This rudimentary subroutine preceding the ordinary error procedure when the sequence of numbers used) is based on the fact that when unexpected numerical values appear it is often more profitable to press on to completion of the problem with appropriate comments than to abandon the computation.

7. POLAR (*X*, *Y*, *N*, *R*, *THETA*)

Transforms the sequences of Cartesian coordinates *X*, *Y* of length *N* to the corresponding sequences of polar coordinates *R*, *THETA*;

$$R = (X^2 + Y^2)^{\frac{1}{2}}$$

$$THETA = \tan^{-1}(Y/X)$$

THETA is given as a multiple of 2π and is adjusted to lie in the range 0 to 1. In time series applications, this subroutine can be used to change from the rectangular to the polar representation of a complex quantity such as the cross-spectrum.

8. FILTER (*A*, *NA*, *F*, *NF*, *B*, *NB*)

Applies the *NF* filter coefficients given by the sequence *F* to the sequence *A* of length *NA*, to produce the sequence *B* of length *NB*. Specifically,

$$B(I) = A(I) * F(NF) + A(I+1) * F(NF-1) + \cdots + A(I+NF-1) * F(1), \quad I = 1, NB$$

with $NB = NA - NF + 1$.

9. AUTCOV (*A*, *NA*, *B*, *L*)

Calculates the series *B* as the autocovariances of the series *A* (of length *NA*) from lags 0 to *L*. Owing to the nature of FORTRAN indexing, *B*(*J*) corresponds to lag (*J* − 1) and the sequence *B* is of length *L* + 1. The formula used is

$$B(J) = \left(\sum_{I=1}^{NA-J+1} A(I) * A(I+J-1) \right) \Big/ (NA - J + 1) - AVE^2, \quad J = 1, \ L+1$$

when *AVE* denotes the average value of the sequence *A*.

10. CRSCOV (*A*, *B*, *N*, *C*, *D*, *E*, *F*, *L*)

Calculates the auto- and cross-covariances of the sequences *A* and *B* (of length *N*), for logs 0 to *L*. The remarks concerning AUTCOV apply. Specifically, for J = 1, L + 1,

$$C(J) = \left(\sum_{I=1}^{NA-J+1} A(I) * A(I+J-1) \right) \Big/ (N - J + 1) - AVEA^2,$$

$$D(J) = \left(\sum_{I=1}^{NA-J+1} B(I) * B(I+J-1) \right) \Big/ (N - J + 1) - AVEB^2,$$

$$E(J) = \left(\sum_{I=1}^{NA-J+1} A(I) * B(I+J-1) \right) \Big/ (N-J+1) - AVEA * AVEB,$$

$$F(J) = \left(\sum_{I=1}^{NA-J+1} A(I+J-1) * B(I) \right) \Big/ (N-J+1) - AVEA * AVEB,$$

where *AVEA*, *AVEB* are the average values of the sequences *A* and *B*.

11. FOURTR (*A*, *L*, *B*, *INDIC*)

Form the sequence *B* as the Fourier transform of the sequence *A* of length *L*+1. For *INDIC* = 1,

$$B(K) = \left(A(1) + \sum_{I=2}^{L} A(I) * \cos \frac{(I-1)(K-1)}{L} \pi + (-1)^{K-1} A(L+1) \right) \Big/ L, \quad K = 1, L+1.$$

For *INDIC* = 2, the cosines are replaced by sines and the end terms omitted. *INDIC* = 3 or 4 provides these same transforms smoothed ("hanned") with coefficients ¼, ½, ¼, the end terms being found from symmetry considerations. Note that the sequences are of length $L+1$, to match AUTCOV and CRSCOV. N must not exceed 1500.

12. DEMOD (A, NA, *FREQ*, X, Y)

"Complex demodulates" the sequence A of length NA at angular frequency $FREQ * \pi$ to form the sequences

$$X(I) = A(I) * \cos((I-1) * FREQ * \pi),$$
$$Y(I) = A(I) * \sin((I-1) * FREQ * \pi), \qquad I = 1, NA.$$

FREQ expresses the demodulating frequency as a fraction of the folding frequency.

13. REMOD (X, Y, N, *FREQ*, A)

The inverse of DEMOD forms

$$A(I) = X(I) * \cos((I-1) * FREQ * \pi) + Y(I) * \sin((I-1) * FREQ * \pi), \quad I = 1, N.$$

14. CENTRE (A, NA, B, NB)

Places the sequence A of length NA in the centre of the sequence B of length NB and surrounds it with zeros. A check is made that $NB \geqq NA$. Note the spelling of CENTRE.

15. NORMAL (A, NA, B)

$$B(I) = A(I)/RMS, \quad I = 1, NA; \quad \text{with } RMS^2 = \sum_{I=1}^{NA} (A(I))^2$$

16. RANGE (A, NA, *BIG*, *SMALL*)

BIG and SMALL are evaluated as the largest and smallest algebraic values of the sequence A of length NA.

17. SQUASH (A, NA, B, *BIG*, *SMALL*, *AMP*)

Calculates *BIG* and *SMALL* as in RANGE, and then multiplies the sequence A of length NA by a suitable constant to form the sequence B, the maximum absolute value of whose terms is *AMP*. Notice that *AMP* is an input variable.

The space requirements of the subroutines are given in the table. The "interpretation" time could be reduced by placing some of the variables in common storage, but we find that (at least with the present version of FORTRAN) this adds considerable complication to the user's task. Space can be economized when necessary by overwriting—when the lengths of two sequences are the same, output and input variables can safely be identified either by Equivalence statements or by giving them the same name. The exceptions to this are: FILTER, AUTCOV, CRSCOV, FOURTR; in POLAR, R must not be identified with X or Y, and in DEMOD, X must not be identified with A.

In their present version, the subroutines are written for use with the operating system BE SYS4. Small changes mainly in the input-output statements may be needed to achieve compatibility with other systems. The source programs of these subroutines are to become available through the SHARE organization shortly.

We would like to acknowledge help with construction of these programs from Miss S. L. Reed and J. F. Ossanna.

Numbers of Locations Occupied

READIN	92_{10}	AUTCOV	135
TAPEIN	80	CRSCOV	221
OUTPUT	113	FOURTR	316 + 3000 COMMON
DETRND	172	DEMOD	101
TAPER	177	REMOD	102
LOGTR	152	CENTRE	80
POLAR	97	NORMAL	70
FILTER	108	RANGE	80
		SQUASH	137

REFERENCES

1. YATES, F., AND HEALY, M. J. R. General computer programmes in statistics. *Bull. Inst. Internat. Stat. 39*, in press
2. KELLY, JOHN L., JR., LOCHBAUM, CAROL, AND VYSSOTSKY, V. A block diagram compiler. *Bell System Tech. J. 40* (May 1961), 669–676.
3. BLACKMAN, R. B., AND TUKEY, J. W. *The Measurement of Power Spectra*. Dover Publications, Inc., New York, 1959.

Letters to the Editor

Historic Photographs Query

Dear Editor:

The Academic Computer Center at West Point is attempting to expand its collection of photographs of historic computers beyond the classic examples of Pascal's, Leibnitz's, Hollerith's and Babbage's machines to give a complete visual history of computer developments.

We would be particularly interested in obtaining information as to possible sources of photos of historic equipments, either mechanical or electronic, in the period 1880–1950.

(Major) WILLIAM F. LUEBBERT
United States Military Academy
West Point, New York

SHARE Programs Available to Nonmembers

Dear Editor:

I should like to emphasize the remarks of Dr. Bernard A Galler regarding the availability of SHARE programs to non-SHARE members. It has always been the SHARE intent and policy to make this material available to all who have a legitimate interest in the programs.

As stated by Dr. Galler such requests for programs may go directly to: Mr. D. C. Cashman, DP Program Information Department, IBM Corporation, 112 East Post Road, White Plains N. Y.

I am sure that I speak for the SHARE membership in indicating that any SHARE member would be willing to make available a list of SHARE programs to nonmembers. I know also that Mr Cashman of the SHARE Distribution Agency is most willing to cooperate in this matter.

GEORGE F. RYCKMAN
SHARE President